SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Mimo Initiative ltd
Date: April 04, 2023

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Mimo Initiative
Approved By	Yevheniy Bezuhlyi | SC Audits Head at Hacken OU
Type	ERC20 rebase tokens wrapper;
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://mimo.capital/
Changelog	22.02.2023 - Initial Review 04.04.2023 - Second Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Mimo Initiative ltd (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is review and security analysis of smart contracts in the repository:

Initial review scope

Repository	https://github.com/mimo-capital/wrapped-rebase-tokens
Commit	ec404e0689f5e6900b1a521daca5801ab99f364f
Functional Requirements	https://github.com/mimo-capital/wrapped-rebase-tokens/blob/auditClarifications/docs/README.md
Contracts	File: ./src/interfaces/chainlink/AggregatorV3Interface.sol SHA3: c820cdec26eff0fff6154a5a8539c00554c88aadab00c2b120b8cd6f81b1122f File: ./src/interfaces/core/IAccessController.sol SHA3: c4eb1efb7b43a7c1258d871dffc2f296b3f2e01d55f3f145d2a0dbc16bd2e059 File: ./src/interfaces/core/IAddressProvider.sol SHA3: c04dc24f9fac71892e083d72e947cd38ce05f8fc98798a286764c16a96f75f85 File: ./src/interfaces/core/IBalancerPool.sol SHA3: a8e3307818a801d1ce5d21c2d38a114ad8d3d04a11f92dc4ad9f47ea0b2b0228 File: ./src/interfaces/core/IBalancerVault.sol SHA3: 075aed4653f3fcbe1d8e9229100751c754fec4c218f9814b958fea4d2da7ea4e File: ./src/interfaces/core/IConfigProvider.sol SHA3: aca92b5ecb85c4367492cc17d6f950b40a19b26cfc14ee6757f7a68931b23940 File: ./src/interfaces/core/IDebtNotifier.sol SHA3: 7d155e15406ab696b9368f6580730569c0f7841fdfe751ca2fa341232427b8c8 File: ./src/interfaces/core/IFeeCollector.sol SHA3: b44c3f6c4536f4bcfc0abb182d5bad93683ca4a6bc82d13edb6e9ce0b50d492c File: ./src/interfaces/core/IFeeDistributor.sol SHA3: b025aac02e18d9f4c2058161b52f71844776ce0cda7f2e34465862948fa17ef9 File: ./src/interfaces/core/IGovernanceAddressProvider.sol SHA3: c4dd048fae3181fa2b37c050376baf052c7afbbeaf0bdadaefc515a3a1fc6681 File: ./src/interfaces/core/IGUniPool.sol SHA3: e1976c5dd6d276502742b95bbf59ddfdc4711a0aa257867b55246cf6b5b0f9c3 File: ./src/interfaces/core/ILiquidationManager.sol SHA3: c187c7c2609b7612eb00fe5cfd604b6d70d7355158fead5962bbca34cc11c285

	File: ./src/interfaces/core/IMerkleDistributor.sol SHA3: 41129172d4a0905d71391dcaec74d94b62ceebce95754b8c11aa330d971a9098 File: ./src/interfaces/core/IPriceFeed.sol SHA3: 058bc1aa502a7d58aa05fc837f5914f6390d5fc81dcc0609e676304739ab7afb File: ./src/interfaces/core/IRatesManager.sol SHA3: d6178f8d94ee797217ce763f84fd5bbeac57761daeeda8b6ce5eb3623d268a1b File: ./src/interfaces/core/ISTABLEX.sol SHA3: c2c701bb0d1dd7ce0c64a5d949bb049108c7c1f14b9c155ae9a4fc84ac8687ba File: ./src/interfaces/core/ISupplyMiner.sol SHA3: 760a578215d6d365006ce40b467d28ffff93833e500db5c086d88dfe85985fd0 File: ./src/interfaces/core/IVaultsCore.sol SHA3: 0ff5ebc4984c6dc5c2830cdb3286f835ba0c9d22a0de744183d6b9405e23deea File: ./src/interfaces/core/IVaultsCoreState.sol SHA3: 1882538da81670c3aea15e51536747cd26a021ee54a226ecc45075767154e110 File: ./src/interfaces/core/IVaultsDataProvider.sol SHA3: 43ad861042e4578dc4653316aab7595752d1b4133e887652626543be10184381 File: ./src/interfaces/core/IWETH.sol SHA3: 55ea3190f076959c287aedd9711be4356a5b66b8357c6452583040aeba455f05 File: ./src/interfaces/core/v1/IAddressProviderV1.sol SHA3: daeb1da8c7751a0db42731868a5d7e4faae2fd6a3f252631aea623fd13dd2aeb File: ./src/interfaces/core/v1/IConfigProviderV1.sol SHA3: a0731b92e61f518969091573d1bdcaf3ec6b6b9f1a4f5b029ff13c6b2fb647f4 File: ./src/interfaces/core/v1/IFeeDistributorV1.sol SHA3: c0ba69a346dd3bab8bc3cb34a5f0090dd63943ec08d74eaeb2b7929060627fe2 File: ./src/interfaces/core/v1/ILiquidationManagerV1.sol SHA3: 87022e329e3d8a8d5b1d79869136062f11689b4755e37cb52a1e66bc5c56a807 File: ./src/interfaces/core/v1/IVaultsCoreV1.sol SHA3: b70e36cf6da19b9baff4a619a1a534c3e1f876ecd01dbc537a7062ce1a5acfbe File: ./src/interfaces/core/v1/IVaultsDataProviderV1.sol SHA3: f26eab57e91c10d88e82b307ece5ce602d3aef544f117c815e3e2f6d91964fb9 File: ./src/interfaces/IBlacklistable.sol SHA3: f969a6374183380e33ce7dbedbea6f92f0a8df547d6f97c611fc966bdf859dc1 File: ./src/interfaces/IMIBToken.sol SHA3: ec629c3103ecf575ccd72a7af33195b65e10c432adfcaddad76091a65eaf9914 File: ./src/interfaces/IWrappedRebaseToken.sol SHA3: 3b4e5f2034f2e5a6ec2dd5b267fd538bd3288b701cde56772499c28307d97416 File: ./src/interfaces/IWrappedRebaseTokenOralce.sol SHA3: 688985b75dccc53117ccf741501b4c76ec973ca37ae543862468ecf60ea336e4 File: ./src/libraries/Errors.sol SHA3: ad6772143ff25b64c489e03339385d990b5d008fefb8bc629361a2a991a26279

	File: ./src/libraries/Roles.sol SHA3: e72e0cec571bbb6416487a9720056fb1964ce63788fc4f43ad3d0e4730aecdc3 File: ./src/libraries/WadRayMath.sol SHA3: b9cafc4715fa55b3936f64d52cc31eed303692d17be139a05510da86e06e9064 File: ./src/libraries/WrappedRebaseTokenErrors.sol SHA3: ef413556caaa38e9bda3ab4cfec1686e16796b5b29382e5f7bec867326d91f39 File: ./src/WrappedRebaseToken.sol SHA3: 57e3b4a71ac99635887d3966dc3c8f28b0d5d31a3fd886b4a44544878ed5af90

Second review scope

Repository	https://github.com/mimo-capital/wrapped-rebase-tokens
Commit	8c27ae16e234c26e899155aeb5fc061387524fed
Functional Requirements	https://github.com/mimo-capital/wrapped-rebase-tokens/blob/8c27ae16e234c26e899155aeb5fc061387524fed/docs/README.md
Contracts	File: ./src/interfaces/chainlink/AggregatorV3Interface.sol SHA3: c820cdec26eff0fff6154a5a8539c00554c88aadab00c2b120b8cd6f81b1122f File: ./src/interfaces/core/IAccessController.sol SHA3: c4eb1efb7b43a7c1258d871dffc2f296b3f2e01d55f3f145d2a0dbc16bd2e059 File: ./src/interfaces/core/IAddressProvider.sol SHA3: c04dc24f9fac71892e083d72e947cd38ce05f8fc98798a286764c16a96f75f85 File: ./src/interfaces/core/IBalancerPool.sol SHA3: a8e3307818a801d1ce5d21c2d38a114ad8d3d04a11f92dc4ad9f47ea0b2b0228 File: ./src/interfaces/core/IBalancerVault.sol SHA3: 075aed4653f3fcbe1d8e9229100751c754fec4c218f9814b958fea4d2da7ea4e File: ./src/interfaces/core/IConfigProvider.sol SHA3: aca92b5ecb85c4367492cc17d6f950b40a19b26cfc14ee6757f7a68931b23940 File: ./src/interfaces/core/IDebtNotifier.sol SHA3: 7d155e15406ab696b9368f6580730569c0f7841fdfe751ca2fa341232427b8c8 File: ./src/interfaces/core/IFeeCollector.sol SHA3: b44c3f6c4536f4bcfc0abb182d5bad93683ca4a6bc82d13edb6e9ce0b50d492c File: ./src/interfaces/core/IFeeDistributor.sol SHA3: b025aac02e18d9f4c2058161b52f71844776ce0cda7f2e34465862948fa17ef9 File: ./src/interfaces/core/IGovernanceAddressProvider.sol SHA3: c4dd048fae3181fa2b37c050376baf052c7afbbeaf0bdadaefc515a3a1fc6681 File: ./src/interfaces/core/IGUniPool.sol SHA3: e1976c5dd6d276502742b95bbf59ddfdc4711a0aa257867b55246cf6b5b0f9c3 File: ./src/interfaces/core/ILiquidationManager.sol SHA3: c187c7c2609b7612eb00fe5cfd604b6d70d7355158fead5962bbca34cc11c285

	File: ./src/interfaces/core/IMerkleDistributor.sol SHA3: 41129172d4a0905d71391dcaec74d94b62ceebce95754b8c11aa330d971a9098 File: ./src/interfaces/core/IPriceFeed.sol SHA3: 058bc1aa502a7d58aa05fc837f5914f6390d5fc81dcc0609e676304739ab7afb File: ./src/interfaces/core/IRatesManager.sol SHA3: d6178f8d94ee797217ce763f84fd5bbeac57761daeeda8b6ce5eb3623d268a1b File: ./src/interfaces/core/ISTABLEX.sol SHA3: c2c701bb0d1dd7ce0c64a5d949bb049108c7c1f14b9c155ae9a4fc84ac8687ba File: ./src/interfaces/core/ISupplyMiner.sol SHA3: 760a578215d6d365006ce40b467d28ffff93833e500db5c086d88dfe85985fd0 File: ./src/interfaces/core/IVaultsCore.sol SHA3: 0ff5ebc4984c6dc5c2830cdb3286f835ba0c9d22a0de744183d6b9405e23deea File: ./src/interfaces/core/IVaultsCoreState.sol SHA3: 1882538da81670c3aea15e51536747cd26a021ee54a226ecc45075767154e110 File: ./src/interfaces/core/IVaultsDataProvider.sol SHA3: 43ad861042e4578dc4653316aab7595752d1b4133e887652626543be10184381 File: ./src/interfaces/core/IWETH.sol SHA3: 55ea3190f076959c287aedd9711be4356a5b66b8357c6452583040aeba455f05 File: ./src/interfaces/core/v1/IAddressProviderV1.sol SHA3: daeb1da8c7751a0db42731868a5d7e4faae2fd6a3f252631aea623fd13dd2aeb File: ./src/interfaces/core/v1/IConfigProviderV1.sol SHA3: a0731b92e61f518969091573d1bdcaf3ec6b6b9f1a4f5b029ff13c6b2fb647f4 File: ./src/interfaces/core/v1/IFeeDistributorV1.sol SHA3: c0ba69a346dd3bab8bc3cb34a5f0090dd63943ec08d74eaeb2b7929060627fe2 File: ./src/interfaces/core/v1/ILiquidationManagerV1.sol SHA3: 87022e329e3d8a8d5b1d79869136062f11689b4755e37cb52a1e66bc5c56a807 File: ./src/interfaces/core/v1/IVaultsCoreV1.sol SHA3: b70e36cf6da19b9baff4a619a1a534c3e1f876ecd01dbc537a7062ce1a5acfbe File: ./src/interfaces/core/v1/IVaultsDataProviderV1.sol SHA3: f26eab57e91c10d88e82b307ece5ce602d3aef544f117c815e3e2f6d91964fb9 File: ./src/interfaces/IBlacklistable.sol SHA3: f969a6374183380e33ce7dbedbea6f92f0a8df547d6f97c611fc966bdf859dc1 File: ./src/interfaces/IMIBToken.sol SHA3: ec629c3103ecf575ccd72a7af33195b65e10c432adfcaddad76091a65eaf9914 File: ./src/interfaces/IWrappedRebaseToken.sol SHA3: 3b4e5f2034f2e5a6ec2dd5b267fd538bd3288b701cde56772499c28307d97416 File: ./src/interfaces/IWrappedRebaseTokenOralce.sol SHA3: 688985b75dccc53117ccf741501b4c76ec973ca37ae543862468ecf60ea336e4 File: ./src/libraries/Errors.sol SHA3: ad6772143ff25b64c489e03339385d990b5d008fefb8bc629361a2a991a26279

	File: ./src/libraries/Roles.sol SHA3: e72e0cec571bbb6416487a9720056fb1964ce63788fc4f43ad3d0e4730aecdc3 File: ./src/libraries/WadRayMath.sol SHA3: b9cafc4715fa55b3936f64d52cc31eed303692d17be139a05510da86e06e9064 File: ./src/libraries/WrappedRebaseTokenErrors.sol SHA3: ef413556caaa38e9bda3ab4cfec1686e16796b5b29382e5f7bec867326d91f39 File: ./src/WrappedRebaseToken.sol SHA3: f9117f844d98a176b44374a2ae8119c8b6c8eebeef82dc413946ac1899c0ab2c

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor gas optimization. These issues won't have a significant impact on code execution but affect code quality

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **3** out of **10**.

- Functional requirements are missing.
- Technical description is not provided.

Code quality

The total Code Quality score is **9** out of **10**.

- The PEP 8 recommendation for the readability of the lines is not followed.
- The development environment is configured.

Test coverage

Code coverage of the project is **27.45%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative cases coverage is missing.
- Interactions with several users are not tested thoroughly.

Security score

As a result of the audit, the code contains **1** low severity issue. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.1**.

1
2
3
4
5
6
7
8
9
10
The final score

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
22 February 2023	2	1	0	0
04 April 2023	1	0	0	0

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Not Relevant
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Not Relevant
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Passed
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed

Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Not Relevant
Style Guide Violation	**Custom**	Style guides and best practices should be followed.	Failed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

System Overview

This Smart Contract enables the wrapping of rebase tokens to simultaneously maintain stable balances and offer users the full rewards they would have received if they had held the unwrapped tokens.

By abstracting the required accounting logic for accruing token balances, these contracts simplify the integration in DeFi applications of rebase tokens: unlike the original rebase tokens, a user's wrapped rebase token balance will not automatically accrue, remaining constant until they are minted, burned, or transferred.

Users can easily return to holding unwrapped tokens by unwrapping their rebase tokens and receiving the corresponding amount.

The only smart contract in this scope is WrappedRebaseToken.sol .

Privileged roles

There are no privileged roles in this Smart Contract.

Risks

- While described in the documentation, WrappedRebaseTokenOracle.sol is out of the scope of this audit.
- As stated in the documentation, the contract does not support deflationary tokens (e.g. fee on transfer), but there are no checks in the code to enforce this.
- The rebase token contracts must implement the SafeERC20 OpenZeppelin library, otherwise the deposit(s) will revert.

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

No high severity issues were found.

■■ Medium

M01. Requirements Violation

In the documentation the *WrappedRebaseToken* is described with the use of the *WrappedRebaseTokenOracle*, this is not reflected in the code.

Path: ./src/WrappedRebaseToken.sol;

Recommendation: Update the documentation.

Status: Fixed (Revised commit: 8c27ae1)

■ Low

L01. Solidity Style Guide

Keeping lines under the PEP 8 recommendation to a maximum of 79 (or 99) characters helps readers easily parse the code.

Path: ./src/WrappedRebaseToken.sol

Recommendation: Follow the official Solidity guidelines.

Status: Reported

L02. Redundant Code

There is a check for the *totalSupply == 0*, this check is redundant, because if the amount to be withdrawn is higher than 0 and the *balanceOf* the user withdrawing is higher or equal to the amount to be withdrawn, then the totalSupply must be higher than 0.

Path: ./src/WrappedRebaseToken.sol : withdraw()

Recommendation: Remove redundant code.

Status: Fixed (Revised commit: 8c27ae1)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.