HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Paribus
Date: May 24th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Paribus.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type	Lending/borrowing platform
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://paribus.io
Timeline	20.04.2022 - 24.05.2022
Changelog	09.05.2022 - Initial Review 24.05.2022 - Second Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Paribus (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 smart contracts in the repository:

Initial review scope
Repository:
https://github.com/Paribus/paribus-protocol
Commit:
da9d3cec639fe1d1f328cd5f0a1a82f4291821be
Technical Documentation:
Type: Litepaper (partial functional requirements provided)
Link: https://paribus.io/documents/PARIBUS-Litepaper-V1.0.pdf

Type: Technical description
Link: https://github.com/Paribus/paribus-protocol/blob/develop/README.md

JS tests: Yes
Contracts:
File: ./contracts/CompoundLens.sol
SHA3: 0b5805b4d05adce8a01f06761ed64d8b6db15b63b39179ab3b6f56e3a6f40ae1

File: ./contracts/Comptroller.sol
SHA3: 20734a3749ed9b3fdba43341b0b8b604007631ed2b52a8ea8da67b2f718b80f7

File: ./contracts/ComptrollerStorage.sol
SHA3: fff9102cce60e36b443912a6ff2ad58586e2ea922aea6eaef4a0be53c3a9ab6f

File: ./contracts/ErrorReporter.sol
SHA3: 13d30ccb98cca9ea681d64befffc314a014d736b359db2537f1c6f5629ae9711

File: ./contracts/Governance/PBXToken.sol
SHA3: a6da41ee0e7f0215c31015747977620df7361781b3cc4265ce17bb007aded01c

File: ./contracts/InterestRateModels/BaseJumpRateModelV2.sol
SHA3: df651bd4569540666f543a637c7dfd8ecae6dbdbab68a86dba81919d4e36b7af

File: ./contracts/InterestRateModels/DAIInterestRateModelV3.sol
SHA3: be3d9841b7712ac24693ab1de3fa632ba7dc3cd9c5eb84439e5f75cf9dcfb867

File: ./contracts/InterestRateModels/InterestRateModel.sol
SHA3: 4e6b1609318b8fe64b5b54b033edfcaba7bd956ed2d167d233fc988536babcd9

File: ./contracts/InterestRateModels/JumpRateModel.sol
SHA3: 565e14cb2064ba24dc7f0656f5568175c0ec75e4debbbbeb6e92f192bd7b3ea4

File: ./contracts/InterestRateModels/JumpRateModelV2.sol
SHA3: d44314698f9164c7c9d6789d252a412ed33e5ce2b4ba0ac64b6d9aba798ad463

File: ./contracts/InterestRateModels/LegacyInterestRateModel.sol
SHA3: 9c8c9ba78a900c14c8ad6d95a65352d367b3e3ceb31e8f05c4ae1e15ad530b95

```
File: ./contracts/InterestRateModels/LegacyJumpRateModelV2.sol
SHA3: 3ca2ee9712be05a0e5d89b872b912f4493fe9609a88887ff91e4ff47eda78969

File: ./contracts/InterestRateModels/WhitePaperInterestRateModel.sol
SHA3: bf1ceb8b168929902930cfba83b8bd86d50116b37a35bdc19494fc77cfa71acc

File: ./contracts/Interfaces/ComptrollerInterface.sol
SHA3: 042419cffaac10e35a03e8c8aff3b0f64f729dab62c61b5fafb6b6c650a7ac41

File: ./contracts/Interfaces/EIP20Interface.sol
SHA3: 9f04d6854fb5ac2b37fadb8eed23264034cd2515f5fc0efbf4a942ec474e8506

File: ./contracts/Interfaces/EIP20NonStandardInterface.sol
SHA3: 83b3090f8985051b09b9b8bf131386957ae23819e0da190f503f7e69a250f654

File: ./contracts/Interfaces/IUniswapV2Factory.sol
SHA3: ffb9bbe0b89b584c04850a5607b1b2e9840ae827e61c4206558aec47e1289ea6

File: ./contracts/Interfaces/IUniswapV2Router.sol
SHA3: 96ba3e89c8c891062e894db392414a09b1de52353b73cd3f33dd8b44031c4cd2

File: ./contracts/Interfaces/PriceOracleInterface.sol
SHA3: 8e5326a843054239a7adbeab09fa5b881da5b23ce677f0d9c68158ba7c6303d3

File: ./contracts/Interfaces/PTokenInterfaces.sol
SHA3: 93e496f0ce65c4db4489de8b82ffc4e301363bf4673047461d9a49936f034d79

File: ./contracts/Liquidator.sol
SHA3: f03fd53e23a301ae5dc3bf76dfd06e5ab56d3dd70c6066d1f5c377ab0222f395

File: ./contracts/Maximillion.sol
SHA3: 2d56356fd27d67d61d7acbc6ded72a8dc6200d25cccde0fc274141a03c97c20e

File: ./contracts/PriceOracle/ArbitrumChainlinkPriceOracle.sol
SHA3: 0000f99d07ffe3872da09636193431ae8a94c5dee79b2b1a3dff794e3d66be5d

File: ./contracts/PriceOracle/ChainlinkPriceOracle.sol
SHA3: c8633124c9530dafa61daa42ce8db4dc6301b4678d128f0c824234f82dac7f6d

File: ./contracts/PriceOracle/RinkebyChainlinkPriceOracle.sol
SHA3: a63acf2f28248c85ee2e2f354f1929acd8b31259b397669c58be07a512beb1f2

File: ./contracts/PriceOracle/SimplePriceOracle.sol
SHA3: bde4c6ccdb78f0b1e05c79ad929d88845d1d8b3fe26dad6dd0a52f47045493ee

File: ./contracts/PToken/PErc20.sol
SHA3: 77e4acf4a2544813b25bea3498588eb88598fe517304409858d6b09bfec785be

File: ./contracts/PToken/PErc20Immutable.sol
SHA3: 3bbfb582f1b2136e8174412fe863bfb39bc88274ed9999964c70942c7e5f6aa5

File: ./contracts/PToken/PEther.sol
SHA3: 51e27702a1e335c0dea567e359f7bb0fb080740a8488135b297c29ccdf105f2c

File: ./contracts/PToken/PToken.sol
SHA3: cbcdc461a30e8d09935070a3e7d42c52983f9bdb51d1c74be54033e8a51e6e24

File: ./contracts/Reservoir.sol
SHA3: c0c2437e4641bd9989ac030288f39c7be782265880eadd70c5ead2bf2da28440

File: ./contracts/Unitroller.sol
SHA3: 11125a95df47f74401332db5bca0e5963e2d5ed87590aae3580a1d6dd9c9c121
```

File: ./contracts/Utils/CarefulMath.sol
SHA3: 6c8f42472882c2b2413befb5d0771032802d19fd1eba3eedac7576d4aecba33c

File: ./contracts/Utils/Exponential.sol
SHA3: 6f7dd16efb32233ceea4acb987ea067724f747e50ba187abbea236db623b786f

File: ./contracts/Utils/ExponentialNoError.sol
SHA3: 92df36a8865b7669e5d0125ca4b2910709a182c24a7bd82150485233370cda36

File: ./contracts/Utils/Ownable.sol
SHA3: 26174cc7c22780b3df5d44dd6b0015cc52779e02ad4bfb691d3eec075a494506

File: ./contracts/Utils/SafeMath.sol
SHA3: 60e6dc8f43c9ca59cf273c4691d1d1d7aac5ed724dfee53b0238f84edbf8e14f

File: ./contracts/Utils/Timelock.sol
SHA3: 3c46800318aa5ffe783d5477ab26f53097f6fdd3d37f185b738d7b0bf9c19097

Second review scope

Repository:
https://github.com/Paribus/paribus-protocol

Commit:
d6e83354692f6fd0e6b988ed06ae676c921d58a3

Technical Documentation:
Type: Litepaper (partial functional requirements provided)
Link: https://paribus.io/documents/PARIBUS-Litepaper-V1.0.pdf

Type: Technical description
Link: https://github.com/Paribus/paribus-protocol/blob/develop/README.md

JS tests: Yes

Contracts:
File: ./contracts/CompoundLens.sol
SHA3: 0b5805b4d05adce8a01f06761ed64d8b6db15b63b39179ab3b6f56e3a6f40ae1

File: ./contracts/Comptroller.sol
SHA3: 20734a3749ed9b3fdba43341b0b8b604007631ed2b52a8ea8da67b2f718b80f7

File: ./contracts/ComptrollerStorage.sol
SHA3: fff9102cce60e36b443912a6ff2ad58586e2ea922aea6eaef4a0be53c3a9ab6f

File: ./contracts/ErrorReporter.sol
SHA3: 13d30ccb98cca9ea681d64befffc314a014d736b359db2537f1c6f5629ae9711

File: ./contracts/Governance/PBXToken.sol
SHA3: b55ce22e5200ab67f9a400cd04cad4ca589312cca2ceab2235ed7945bc1cca88

File: ./contracts/InterestRateModels/BaseJumpRateModelV2.sol
SHA3: df651bd4569540666f543a637c7dfd8ecae6dbdbab68a86dba81919d4e36b7af

File: ./contracts/InterestRateModels/DAIInterestRateModelV3.sol
SHA3: be3d9841b7712ac24693ab1de3fa632ba7dc3cd9c5eb84439e5f75cf9dcfb867

File: ./contracts/InterestRateModels/InterestRateModel.sol
SHA3: 4e6b1609318b8fe64b5b54b033edfcaba7bd956ed2d167d233fc988536babcd9

File: ./contracts/InterestRateModels/JumpRateModel.sol
SHA3: 565e14cb2064ba24dc7f0656f5568175c0ec75e4debbbbeb6e92f192bd7b3ea4

File: ./contracts/InterestRateModels/JumpRateModelV2.sol

```
SHA3: d44314698f9164c7c9d6789d252a412ed33e5ce2b4ba0ac64b6d9aba798ad463

File: ./contracts/InterestRateModels/LegacyInterestRateModel.sol
SHA3: 9c8c9ba78a900c14c8ad6d95a65352d367b3e3ceb31e8f05c4ae1e15ad530b95

File: ./contracts/InterestRateModels/LegacyJumpRateModelV2.sol
SHA3: 3ca2ee9712be05a0e5d89b872b912f4493fe9609a88887ff91e4ff47eda78969

File: ./contracts/InterestRateModels/WhitePaperInterestRateModel.sol
SHA3: bf1ceb8b168929902930cfba83b8bd86d50116b37a35bdc19494fc77cfa71acc

File: ./contracts/Interfaces/ComptrollerInterface.sol
SHA3: 042419cffaac10e35a03e8c8aff3b0f64f729dab62c61b5fafb6b6c650a7ac41

File: ./contracts/Interfaces/EIP20Interface.sol
SHA3: 9f04d6854fb5ac2b37fadb8eed23264034cd2515f5fc0efbf4a942ec474e8506

File: ./contracts/Interfaces/EIP20NonStandardInterface.sol
SHA3: 83b3090f8985051b09b9b8bf131386957ae23819e0da190f503f7e69a250f654

File: ./contracts/Interfaces/IUniswapV2Factory.sol
SHA3: ffb9bbe0b89b584c04850a5607b1b2e9840ae827e61c4206558aec47e1289ea6

File: ./contracts/Interfaces/IUniswapV2Router.sol
SHA3: 96ba3e89c8c891062e894db392414a09b1de52353b73cd3f33dd8b44031c4cd2

File: ./contracts/Interfaces/PriceOracleInterface.sol
SHA3: 8e5326a843054239a7adbeab09fa5b881da5b23ce677f0d9c68158ba7c6303d3

File: ./contracts/Interfaces/PTokenInterfaces.sol
SHA3: 93e496f0ce65c4db4489de8b82ffc4e301363bf4673047461d9a49936f034d79

File: ./contracts/Liquidator.sol
SHA3: 42ce4e2f932eeeba0ab1d3eafc092e83d7d7ace4d1a183855699e71b72649f1b

File: ./contracts/Maximillion.sol
SHA3: 2d56356fd27d67d61d7acbc6ded72a8dc6200d25cccde0fc274141a03c97c20e

File: ./contracts/PriceOracle/ArbitrumChainlinkPriceOracle.sol
SHA3: 0000f99d07ffe3872da09636193431ae8a94c5dee79b2b1a3dff794e3d66be5d

File: ./contracts/PriceOracle/ChainlinkPriceOracle.sol
SHA3: 90b19e2da49608aae5619208f136152066537845193f37fe74fc92c8dfaac694

File: ./contracts/PriceOracle/RinkebyChainlinkPriceOracle.sol
SHA3: 78fbee1b84489608a0ba27dbb82daaa84edae6d8a66e7f848fd2285d7cb03469

File: ./contracts/PriceOracle/SimplePriceOracle.sol
SHA3: bde4c6ccdb78f0b1e05c79ad929d88845d1d8b3fe26dad6dd0a52f47045493ee

File: ./contracts/PToken/PErc20.sol
SHA3: 77e4acf4a2544813b25bea3498588eb88598fe517304409858d6b09bfec785be

File: ./contracts/PToken/PErc20Delegate.sol
SHA3: 5ca251360c8bb7ba5dbacd87f79898d7edd9dc7cfb06c8547bb3ed5a90fa24d8

File: ./contracts/PToken/PErc20Delegator.sol
SHA3: 992820a8d539dd276cff5249507bb2a887fa15ce05fd625b0f572189a561a312

File: ./contracts/PToken/PErc20Immutable.sol
SHA3: 3bbfb582f1b2136e8174412fe863bfb39bc88274ed9999964c70942c7e5f6aa5
```

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

```
File: ./contracts/PToken/PEther.sol
SHA3: 51e27702a1e335c0dea567e359f7bb0fb080740a8488135b297c29ccdf105f2c

File: ./contracts/PToken/PToken.sol
SHA3: cbcdc461a30e8d09935070a3e7d42c52983f9bdb51d1c74be54033e8a51e6e24

File: ./contracts/Unitroller.sol
SHA3: 11125a95df47f74401332db5bca0e5963e2d5ed87590aae3580a1d6dd9c9c121

File: ./contracts/Utils/CarefulMath.sol
SHA3: 6c8f42472882c2b2413befb5d0771032802d19fd1eba3eedac7576d4aecba33c

File: ./contracts/Utils/Exponential.sol
SHA3: 6f7dd16efb32233ceea4acb987ea067724f747e50ba187abbea236db623b786f

File: ./contracts/Utils/ExponentialNoError.sol
SHA3: 92df36a8865b7669e5d0125ca4b2910709a182c24a7bd82150485233370cda36

File: ./contracts/Utils/Ownable.sol
SHA3: 26174cc7c22780b3df5d44dd6b0015cc52779e02ad4bfb691d3eec075a494506

File: ./contracts/Utils/SafeMath.sol
SHA3: 60e6dc8f43c9ca59cf273c4691d1d1d7aac5ed724dfee53b0238f84edbf8e14f

File: ./contracts/Utils/Timelock.sol
SHA3: 3c46800318aa5ffe783d5477ab26f53097f6fdd3d37f185b738d7b0bf9c19097
```

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

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

Documentation quality

The Customer provided superficial functional requirements and technical requirements. The total Documentation Quality score is **5** out of **10**.

Code quality

The total CodeQuality score is **7** out of **10**. Commented code. TODO comments. Unit tests were provided.

Architecture quality

The architecture quality score is **10** out of **10**. Follows best practices.

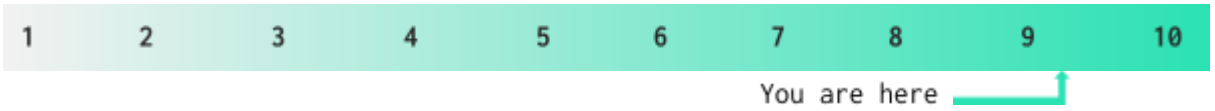
Security score

As a result of the audit, security engineers found **2** medium severity issues, **1** medium issue was fixed and **1** remained. 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: **9.2**.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Checked Items

We have audited provided smart contracts for commonly known and more specific vulnerabilities. Here are some of the items that are 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.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Failed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Failed
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 destroyed until it has funds belonging to users.	Passed
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Uninitialized Storage Pointer	SWC-109	Storage type should be set explicitly if the compiler version is < 0.5.0.	Not Relevant
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.	Passed
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless it is 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.	Passed
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Passed
Signature Unique Id	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	Passed
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.	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.	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.	Passed
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.	Passed

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
Repository Consistency	**Custom**	The repository should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be 100%, 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, that may be changed in the future.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

System Overview

The *Paribus Protocol* is an Ethereum smart contract for supplying or borrowing assets. Through the *pToken* contracts, accounts on the blockchain supply capital (Ether or ERC-20 tokens) to receive pTokens or borrow assets from the protocol (holding other assets as collateral). The *Paribus pToken* contracts track these balances and algorithmically set interest rates for borrowers.

The core contracts in the Paribus Protocol:

- *PToken, PErc20 and PEther* — the *Paribus pTokens*, self-contained borrowing and lending contracts. *PToken* contains the core logic, and *PErc20* and *PEther* add public interfaces for *Erc20* tokens and Ether, respectively. Each *PToken* is assigned an interest rate and risk model, and allows accounts to *mint* (supply capital), *redeem* (withdraw capital), *borrow* and *repay a borrow*. Each *PToken* is an *ERC-20* compliant token where balances represent ownership of the market.
- *Comptroller* — the risk model contract, which validates permissible user actions and disallows actions if they do not fit certain risk parameters. For instance, the *Comptroller* enforces that each borrowing user must maintain a sufficient collateral balance across all *pTokens*.
- *Paribus (PBX)* — the Paribus Governance Token.
- *InterestRateModel* — contracts which define interest rate models. These models algorithmically determine interest rates based on the current utilization of a given market (that is, how much of the supplied assets are liquid versus borrowed).
- *WhitePaperInterestRateModel* — initial interest rate model, as defined in the Whitepaper. This contract accepts a base rate and slope parameter in its constructor.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Findings

Critical

No critical severity issues were found.

High

No high severity issues were found.

Medium

1. **Unfinished code**

   TODO comments in the code.

   This indicates that the code is not yet complete.

   Contracts: Liquidator.sol, Comptroller.sol, Reservoir.sol, ChainlinkPriceOracle.sol

   Recommendation: complete the code to meet all the requirements and delete the TODO comments.

   Status: Reported

2. **The code does not consider all cases**

   The decimal normalization in the *getUnderlyingPrice* function works correctly only if *underlyingDecimals* is 18 and *priceDecimals* is less than or equal to 18.

   The function may not work properly in some cases.

   Contract: ChainlinkPriceOracle.sol

   Function: getUnderlyingPrice

   Recommendation: change decimals normalization to a more general one that works properly with any decimals values.

   Status: Fixed (Revised commit: d6e8335)

Low

No low severity issues were found.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date 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 audit makes no statements or warranties on the security of the code. It also cannot be considered a 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.

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 audit cannot guarantee the explicit security of the audited smart contracts.