HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Blueberry Foundation
Date: April 25, 2023

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 Blueberry Foundation
Approved By	Yevheniy Bezuhlyi | SC Audits Head at Hacken OU
Type	Lending & Leverage Farming Platform
Platform	EVM
Language	Solidity
Methodology	Link
Website	blueberry.garden
Changelog	07.04.2023 - Initial Review 25.04.2023 - 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 Blueberry Foundation (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 includes review and security analysis of the following smart contracts from the provided repository:

Initial review scope

Repository	https://github.com/Blueberryfi/blueberry-core
Commit	42671b9dc1f6b52c82efeef901f7de85646a7d09
Functional Requirements	Public GitBook General Architecture Overview
Technical Requirements	Essential Scripts
Contracts	File: ./contracts/BlueBerryBank.sol SHA3: fb289c23842e720cfedb169f5674b9cdeacdac66fcb4318fbf34e71098620e2d File: ./contracts/FeeManager.sol SHA3: a4c90e5e107fdb97de1af97e5c8f6e65c85a28a762d87950cb4537f28e3f2a6f File: ./contracts/ProtocolConfig.sol SHA3: d774188c545e4f4bcfa7989d0e6a2ad38d9487b11d9b6a97519bc7dfc8ce02a4 File: ./contracts/interfaces/IBank.sol SHA3: 777f084fc8e4401ade12391e7ef47f1f17b1547648fb9238d566988e41c1faed File: ./contracts/interfaces/IBaseOracle.sol SHA3: 1db54613a110c42336a1f00b14699f86dedf950eed8997f844be8e92d94a2ff0 File: ./contracts/interfaces/ICoreOracle.sol SHA3: 239ba394a5236bec9dd18891c973944c98593e603bf3922810839180e0191ebd File: ./contracts/interfaces/IERC20Wrapper.sol SHA3: c9479f002df21291440465a676dccf592408916d6be88a9ac4467b8cf58ed758 File: ./contracts/interfaces/IFeeManager.sol SHA3: 9d7dbcf3dd9a1ed6c54e40a6be5057d9f7889c85fcc4f5f6104ae95d2f171ac5 File: ./contracts/interfaces/IHardVault.sol SHA3: 3cb04de9cc8d88b32b3f938abe9a0f19f890f46e6f2a4b541344272709c1ee89 File: ./contracts/interfaces/IProtocolConfig.sol SHA3: 2cd0b9758c9f9f56ef9f8ad7579dc0cbae749b21b8de387790bb6ac9c25d4002 File: ./contracts/interfaces/ISoftVault.sol SHA3: fbbd551354d38f012f4623aaeef8957098af822f5c962d2d6d77a20b045a9ac4

	File: ./contracts/interfaces/IWERC20.sol SHA3: 654a86c582b5819881339ef13ce004b64733bff68326b137588be0ca17570bea File: ./contracts/interfaces/IWETH.sol SHA3: 446f3e23908913f35b2515fc297117b3f920b140a0014f0356db969c1d550c9f File: ./contracts/interfaces/IWIchiFarm.sol SHA3: 49bf251356eea49824b2e9065caee1989fca859d5747ef86d76a353dcb3d0da6 File: ./contracts/interfaces/balancer/IBalancerPool.sol SHA3: 1c53dd53251d7a22a9e32bc6fb25afba8c6223afd6c3c378ebf2255dacb12921 File: ./contracts/interfaces/band/IStdReference.sol SHA3: 0f0355c059c5c8a0d407b42a5aac60d25e807917123f9f35a6978732af34615e File: ./contracts/interfaces/chainlink/IFeedRegistry.sol SHA3: 8ca9ce016071ca31dcfd31e1834f636d7ffba517d5aa63e47739fbdd92150614 File: ./contracts/interfaces/compound/ICErc20.sol SHA3: fb71211d798642299d48034ed6d163c2ec5b308fd198d2f7554d8acec3786679 File: ./contracts/interfaces/compound/ICErc20_2.sol SHA3: a8c03e372a6a8ae4c2329e7a7d5fba5d520dd2593a091b19065c7318eb403ed2 File: ./contracts/interfaces/compound/ICEtherEx.sol SHA3: fdb106979e9699fa7a79a7a84a01f871e4433da75bf5109cb69be2baf648d8c0 File: ./contracts/interfaces/compound/IComptroller.sol SHA3: 8eafdd1adac94b32b1ad678b29be976320b764d19ccb8d14db456af3f8901b88 File: ./contracts/interfaces/curve/ICurvePool.sol SHA3: 58e0fd3e74a1963f2dfad95c1c47a2820a5ca58204d2d70b10659a72833e5028 File: ./contracts/interfaces/curve/ICurveRegistry.sol SHA3: 1379a844012d10da4da18414617dd080a448330bf3af2198013edc9b253f586a File: ./contracts/interfaces/curve/ILiquidityGauge.sol SHA3: 61d9e445d6c875e89acd4e3d75ef1289a43f2450926ac7cf34b06d7977d94563 File: ./contracts/interfaces/ichi/IIchiFarm.sol SHA3: f03dd038881f9608466857ae94890c309e82be916dcf09e332abe31536836e58 File: ./contracts/interfaces/ichi/IIchiV2.sol SHA3: 1b33b4387ec9f1a625cdbda62c2f7483275fe28f7fac424a026932e21d32d20f File: ./contracts/interfaces/ichi/IICHIVault.sol SHA3: 6e1ab915595d1e0f03cd382201d847eca802dc76f2e3ace869a57b2d5ad423e5 File: ./contracts/interfaces/ichi/IICHIVaultFactory.sol SHA3: 6e8e4faec91904f61ea6d4d691a32c79807b2de1e2c3a4c66b91cc91cd88329e File: ./contracts/interfaces/sushi/IMasterChef.sol SHA3: 7a664ef616ae5925fda6162ae82b72187f6682d693da9ac796a648980e39c96d File: ./contracts/libraries/BBMath.sol SHA3: c296d7517b2e45537edbb4453656d4fe350b783958a78fb6944c3db3d63b48d7 File: ./contracts/libraries/UniV3/UniV3WrappedLib.sol SHA3: db880c0271b9cec57045bc4676ded5b3045910d8e55f72cc2a16040f91dfd16d File: ./contracts/libraries/UniV3/UniV3WrappedLibMockup.sol SHA3: adb0165956b3f496ef16a2f0a2194007434baa6e52a529cb1cc752bad1fa3e1f

	File: ./contracts/oracle/AggregatorOracle.sol SHA3: 15f655b2cb3ca12f33cf244eb55d85887b2626fe32fa42237ca1f81acdd175cc File: ./contracts/oracle/BandAdapterOracle.sol SHA3: 83ed3c63ed8c08569ca17cf731b4a68692aaa10c5b06852c65a4958ee90ab823 File: ./contracts/oracle/BaseAdapter.sol SHA3: 872a7d8c987cdfc18e4a4d780e87630d88aae50a7c3c5ff3aab799e98b3e9cc7 File: ./contracts/oracle/ChainlinkAdapterOracle.sol SHA3: c56c9d55f1bfb10c09d0a3159cc8095432ac6573006f9bd4828efd863d170e5f File: ./contracts/oracle/CoreOracle.sol SHA3: 4cc82f1a15f90ab67a855aa70a9e586c25368d00a6b7a557d9d8561870e659e8 File: ./contracts/oracle/IchiVaultOracle.sol SHA3: 5be6b6a2b2c74f67b33ee106a878049b077c4fe56bb8ab01a4db189c49566c34 File: ./contracts/oracle/UniswapV2Oracle.sol SHA3: 978e7e11df9103502a319d22535293120a59741c2e045c1e10568bc6903af588 File: ./contracts/oracle/UniswapV3AdapterOracle.sol SHA3: 556aa5c10bc11130f78b1cba19d720d9d564f3732bea57e94147b8820e085005 File: ./contracts/oracle/UsingBaseOracle.sol SHA3: ebee3c87d6657584bc3dde07b18292c56ceb5053021447d4de2fbc4e13e0a453 File: ./contracts/spell/BasicSpell.sol SHA3: 059b8ef3d0881987f904e1b4d8d31ed4543fb8d31e38bd54efb89389f03d70b5 File: ./contracts/spell/IchiSpell.sol SHA3: d7d65df47e9e24ab282a68f56a9d71f4a242140106fd2325b82bb92b667b9a4b File: ./contracts/utils/BlueBerryConst.sol SHA3: 367351804cd29f826de2027d5637a5bf5e7d7e0326f01959bba946392dc19eeb File: ./contracts/utils/BlueBerryErrors.sol SHA3: 18234418a1698ca8b81144cebe60f52897dc4490abb2ad8a3e15d52900721e88 File: ./contracts/utils/ERC1155NaiveReceiver.sol SHA3: c4c963d8f3aece96a64ce12d37be1025d667929a10889f35a263c5f010df7274 File: ./contracts/vault/HardVault.sol SHA3: 11b1f973c63763f6b0813dd23b2550391003b029d0e5eb9d3ccf3b98021f34d7 File: ./contracts/vault/SoftVault.sol SHA3: 1f5f86f874b33d69eaed4833d63d2a91b975bdb74b0ef27b2a7d6787843a0241 File: ./contracts/wrapper/WERC20.sol SHA3: 0f66a2769a0971937f6a4b3a15d8e7a4cbbe82813e5c225eb5178bc9743aea3e File: ./contracts/wrapper/WIchiFarm.sol SHA3: 78fc477e7f62b87f1b8e725072aeb7c117e75870fdf719a4e6b395c6a2231025

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

Second review scope

Repository	https://github.com/Blueberryfi/blueberry-core
Commit	13040356e26a0405f26b67ecc4b63c1631411ebe
Functional Requirements	Public GitBook General Architecture Overview
Technical Requirements	Essential Scripts
Contracts	File: ./contracts/BlueBerryBank.sol SHA3: d8731a795c09346b9360884f7c043345d7799486d112ceada4fe79a3b1604bde File: ./contracts/FeeManager.sol SHA3: b2d3d42a42b271309a4d69750380569fa100f7dc40537368b85f04dd57b35954 File: ./contracts/ProtocolConfig.sol SHA3: d5cef5bdaec1277283d326be6ac2a0b019aea2d6c4535325a184a51d3ad730a7 File: ./contracts/interfaces/IBank.sol SHA3: 1a100542f0c5bec03b11b8ae0fcb5609ebd4335975aa22451306edc342163195 File: ./contracts/interfaces/IBaseOracle.sol SHA3: 1db54613a110c42336a1f00b14699f86dedf950eed8997f844be8e92d94a2ff0 File: ./contracts/interfaces/ICoreOracle.sol SHA3: b4e05f0b3dcf6e53e6c01112c199d10c2d05d0e7796708e1b54ee9d3a1b39edf File: ./contracts/interfaces/IERC20Wrapper.sol SHA3: c9479f002df21291440465a676dccf592408916d6be88a9ac4467b8cf58ed758 File: ./contracts/interfaces/IFeeManager.sol SHA3: 9d7dbcf3dd9a1ed6c54e40a6be5057d9f7889c85fcc4f5f6104ae95d2f171ac5 File: ./contracts/interfaces/IHardVault.sol SHA3: 3cb04de9cc8d88b32b3f938abe9a0f19f890f46e6f2a4b541344272709c1ee89 File: ./contracts/interfaces/IProtocolConfig.sol SHA3: 2cd0b9758c9f9f56ef9f8ad7579dc0cbae749b21b8de387790bb6ac9c25d4002 File: ./contracts/interfaces/ISoftVault.sol SHA3: 576afe720c25d4c5f8ab2836e5346f93c5ba215463c366b5525ed2b14eacaab6 File: ./contracts/interfaces/IWERC20.sol SHA3: 654a86c582b5819881339ef13ce004b64733bff68326b137588be0ca17570bea File: ./contracts/interfaces/IWETH.sol SHA3: 446f3e23908913f35b2515fc297117b3f920b140a0014f0356db969c1d550c9f File: ./contracts/interfaces/IWIchiFarm.sol SHA3: 49bf251356eea49824b2e9065caee1989fca859d5747ef86d76a353dcb3d0da6 File: ./contracts/interfaces/balancer/IBalancerPool.sol SHA3: 1c53dd53251d7a22a9e32bc6fb25afba8c6223afd6c3c378ebf2255dacb12921 File: ./contracts/interfaces/band/IStdReference.sol SHA3: 0f0355c059c5c8a0d407b42a5aac60d25e807917123f9f35a6978732af34615e File: ./contracts/interfaces/chainlink/IFeedRegistry.sol SHA3: 8ca9ce016071ca31dcfd31e1834f636d7ffba517d5aa63e47739fbdd92150614

File: ./contracts/interfaces/compound/ICErc20.sol
SHA3: fb71211d798642299d48034ed6d163c2ec5b308fd198d2f7554d8acec3786679

File: ./contracts/interfaces/compound/ICErc20_2.sol
SHA3: a8c03e372a6a8ae4c2329e7a7d5fba5d520dd2593a091b19065c7318eb403ed2

File: ./contracts/interfaces/compound/ICEtherEx.sol
SHA3: fdb106979e9699fa7a79a7a84a01f871e4433da75bf5109cb69be2baf648d8c0

File: ./contracts/interfaces/compound/IComptroller.sol
SHA3: c61a8a0a4eff0b41dce16ca828fe00c66d95ccb6f6137754224ea7d29c42b7a4

File: ./contracts/interfaces/curve/ICurvePool.sol
SHA3: 58e0fd3e74a1963f2dfad95c1c47a2820a5ca58204d2d70b10659a72833e5028

File: ./contracts/interfaces/curve/ICurveRegistry.sol
SHA3: 1379a844012d10da4da18414617dd080a448330bf3af2198013edc9b253f586a

File: ./contracts/interfaces/curve/ILiquidityGauge.sol
SHA3: 61d9e445d6c875e89acd4e3d75ef1289a43f2450926ac7cf34b06d7977d94563

File: ./contracts/interfaces/ichi/IIchiFarm.sol
SHA3: f03dd038881f9608466857ae94890c309e82be916dcf09e332abe31536836e58

File: ./contracts/interfaces/ichi/IIchiV2.sol
SHA3: 1b33b4387ec9f1a625cdbda62c2f7483275fe28f7fac424a026932e21d32d20f

File: ./contracts/interfaces/ichi/IICHIVault.sol
SHA3: 6e1ab915595d1e0f03cd382201d847eca802dc76f2e3ace869a57b2d5ad423e5

File: ./contracts/interfaces/ichi/IICHIVaultFactory.sol
SHA3: 6e8e4faec91904f61ea6d4d691a32c79807b2de1e2c3a4c66b91cc91cd88329e

File: ./contracts/interfaces/sushi/IMasterChef.sol
SHA3: 7a664ef616ae5925fda6162ae82b72187f6682d693da9ac796a648980e39c96d

File: ./contracts/interfaces/uniswap/ISwapRouter.sol
SHA3: 15c12024ccbb7a09a89ce7b538f066ce3147b79596b7651a044f366f4742600a

File: ./contracts/interfaces/uniswap/IUniswapV2Router02.sol
SHA3: 0b62ead2d33459a571593e85f9171fe474c4db60bc1d5d2264dbb8cb4359c307

File: ./contracts/libraries/BBMath.sol
SHA3: c296d7517b2e45537edbb4453656d4fe350b783958a78fb6944c3db3d63b48d7

File: ./contracts/libraries/UniV3/UniV3WrappedLib.sol
SHA3: 12a31bb17192d9416840cd07a8c718b5e005735e126a5084d9be5b669c32d190

File: ./contracts/libraries/UniV3/UniV3WrappedLibMockup.sol
SHA3: d3431f97c177ae43e56807d37470cbf7c3ccf6ac7360227573a50d327dee86f6

File: ./contracts/oracle/AggregatorOracle.sol
SHA3: 1e1be29a36cdacb1ad794b4e487addd59830d374fdc581f2c54ddbe02f16b10e

File: ./contracts/oracle/BandAdapterOracle.sol
SHA3: 116e18e16a2f72a88522be8fbb16dcd87086fb6b3de82fb4a48f12172e770173

File: ./contracts/oracle/BaseAdapter.sol
SHA3: 341af6a666d8411ba0b9c03d469ee25469d96e23666d390dba07b87acc6e6c2c

File: ./contracts/oracle/BaseOracleExt.sol
SHA3: b49b8d1fe8b9155e0aff0e029ff58fc291d2534cea6e67674910bc2071433b0a

	File: ./contracts/oracle/ChainlinkAdapterOracle.sol SHA3: e5b456a892bd60eac7b1c25714799551362beb4cf34305ead5660eb490544b6a File: ./contracts/oracle/CoreOracle.sol SHA3: 93a314d9075bde5b0249151e1332e47f0d7999e215163cbd4db948e4f235ee91 File: ./contracts/oracle/IchiVaultOracle.sol SHA3: a8369048c4b64d29eb050a8f8be942d816e27c206c81d4600b1528e99ae4139a File: ./contracts/oracle/UniswapV2Oracle.sol SHA3: 055624d61fa91c12e0869c3b6f13f1987f4205421dd3bffffa6c12090dcab990 File: ./contracts/oracle/UniswapV3AdapterOracle.sol SHA3: eddfca88b8eea9a620f98d133a3136d492ecabc32156e4f7f567c5a0acd8d232 File: ./contracts/oracle/UsingBaseOracle.sol SHA3: ebee3c87d6657584bc3dde07b18292c56ceb5053021447d4de2fbc4e13e0a453 File: ./contracts/spell/BasicSpell.sol SHA3: 0e5628b3d5d8330fbc78cefe47ccda23b2a69208d8457866920c933a915f5439 File: ./contracts/spell/IchiSpell.sol SHA3: a586f6d11d7f2fd48265e0658590f2f53aa43a3b1881837757221e982a9693d4 File: ./contracts/utils/BlueBerryConst.sol SHA3: 86ffa312907d0b1edb3c4c8c3332848b80e6830b67aea735d1d1ac3dd0cdbb01 File: ./contracts/utils/BlueBerryErrors.sol SHA3: f5549a4221bce91882fb23f1065110c0fd39742529700dbb1f590f11bb71a3ba File: ./contracts/utils/EnsureApprove.sol SHA3: b17e1fe16b8bf22ebe851596bfe913bfe1731af277f7f34863ad7ff6027bc004 File: ./contracts/utils/ERC1155NaiveReceiver.sol SHA3: bc383cea203a41713365e01c26922f4a10129db6d1ba587e1659ed93c4cf0045 File: ./contracts/vault/HardVault.sol SHA3: 7eca66aaf4b072846baadcf6baeb2f705683f24b1a2501c0bf29c068d89bdb43 File: ./contracts/vault/SoftVault.sol SHA3: 5d57ae02bd2d24dea653b8b9d6cec84f8a217cb4ea20c066902c7b43c6dcecc7 File: ./contracts/wrapper/WERC20.sol SHA3: c6d978f4a98503073296505fa1d4ea4109d555f36aa2cb9074f94cc2c152d56b File: ./contracts/wrapper/WIchiFarm.sol SHA3: dfa2fdddc3bc5192bc3bca3b6de0e2286b0467da1bef54ab561fd43a789075b4

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 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.

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 scoring methodology.

Documentation quality

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

- Functional requirements are sufficient.
- Technical description is provided.

Code quality

The total Code Quality score is **8** out of **10**.

- The code is designed in a way to do recursive contract calls *BlueBerryBank -> Spell -> BlueBerryBank*.
- The code is missing initialization of *PausableUpgradeable*.
- The code is missing events for such actions as *Spell* whitelist or permitting interactions by smart contracts.
- The code performs calls to the *ERC20-approve* function without checking the return value.
- The code highly trusts vaults, tokens, etc. However, all the interacted systems are whitelisted by the system owners.
- The development environment is configured.

Test coverage

Code coverage of the project is **89.39%** (branch coverage).

Security score

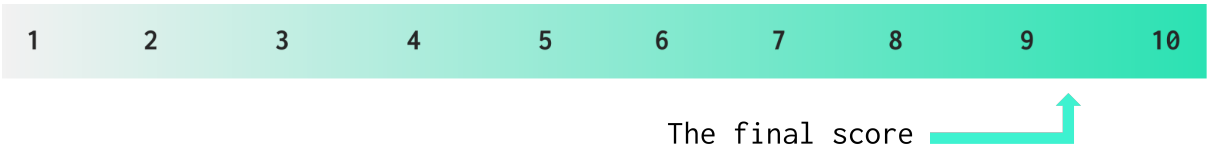
As a result of the audit, the code does not contain security issues. The score is **10** out of **10**.

All found issues are displayed in the Findings section of the report.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.2**.

The system users should acknowledge all the risks summed up in the Risks section of the report.



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

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
7 April 2023	19	12	2	0
25 April 2023	0	0	0	0

Risks

- The system highly relies on the *Blueberry Money Market* and *ICHI Farm* functionality which is out of the audit scope. The mentioned systems receive access to user funds.
- The system uses prices received from *Band Protocol*, *Chainlink*, *Uniswap* (at previous blocks), and possibly other sources. The data providers may affect user position states. Check if the sources are stable and do not have disclosed vulnerabilities.
- *IchiVaultOracle* highly relies on *IchiVault*. In case the *IchiVault* reserves could be manipulated, the resulting price may be affected and some positions become at risk of liquidation. It is recommended to check that the used *IchiVault* instances do not have disclosed vulnerabilities and could not be significantly affected by any third party.
- The *AggregatorOracle* contract receives data from several sources and checks if the deviation of the price is within bounds. However, in case only one source responds with a price, that value is used. In case some of the sources are unstable or vulnerable, an attacker may manipulate the resulting price. It is recommended to ensure that the contract relies on stable data providers.
- According to the documentation, only the *CoreOracle* contract should be used for providing prices to the target contracts (*BlueberryBank*, *IchiSpell*, etc.). In case another oracle is used, important price validations may be missed.
- The oracles highly depend on the owner. The owner is able to manipulate token prices received by the project.
- The *Admin* of *BlueBerryBank* may disable/enable actions for users at any time: lend, withdraw, repay, borrow.
- The *Admin* of *BlueBerryBank* may de-whitelist previously whitelisted tokens and spells.
- The system may be vulnerable to uncommon ERC20 tokens such as tokens with floating decimals (and 19+ decimals), fee-on-transfer tokens, or tokens with a non-failing *approve* function implemented (which in case of error returns *false* instead of reverting the transaction).
- The *IchiSpell* contract allows the *Admin* to add strategies with custom vault addresses. The pool address received from the vault contract is able to drain any allowances to the *IchiSpell* contract.

- In case repaying is not enabled on the contract, it is impossible to liquidate risky positions. Users and the platform may lose their assets or funds value.
- In case of high borrowing interest rates, it may be impossible to withdraw lent funds and collaterals until someone's debt is not repaid.
- The *Admin* of *BlueBerryBank* may change the maximum "loan to value" proportion for any collateral at any time.

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

System Overview

Blueberry Bank is a leveraged yield-farming platform. It consists of

- *BlueBerryBank* – the main system contract, which supports borrowing, lending, and repaying debts according to a selected strategy.
- *WERC20* – an ERC1155 wrapper of ERC20 tokens.
- *WIchiFarm* – a wrapper of the ICHI liquidity management protocol.
- *HardVault* – an ERC1155 contract which represents the deposits as an ERC1155 collection.
- *SoftVault* – an ERC20 token contract that represents the number of tokens lent to the Blueberry Money Market.
- *Spell contracts* – strategy contracts that support undersecure borrowed funds to be used for leverage farming.
- *Oracles* – contracts to manage trusted data sources and allow retrieval of token prices.

Privileged roles

FeeManager, HardVault, SoftVault, WERC20, WIchiFarm, BlueBerryBank, ProtocolConfig, IchiSpell, CoreOracle:

- Admin (owner) – can upgrade the contract logic.

BlueBerryBank:

- Admin (owner) – can
  - allow or disallow taking/repaying loans,
  - allow or disallow lending/withdrawing isolated collateral,
  - whitelist a spell contract,
  - whitelist tokens,
  - whitelist ERC1155 tokens,
  - allow or disallow contract calls.
- EOA or Whitelisted contract – can create a position and interact with the contract.

ProtocolConfig:

- Admin (owner) – can
  - set all the fee values,
  - set fee manager, treasury, pool, and vault addresses.

IchiSpell:

- Admin (owner) – can
  - add strategies,
  - set “max position size”,
  - set “max loan to value” for different strategies and tokens.

Oracle:

- Admin (owner) – can
  - pause and unpause CoreOracle,
  - set Oracle routes, refs, data feeds, and time gaps.

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

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.	Passed
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

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

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.	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.	Passed

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

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.	Passed
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.	Passed
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. The 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

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

H01. Upgradeability Issues – Missing Storage Gap

The contracts are abstract and upgradable but do not follow the upgradability best practices by not adding a gap in the contract storage.

This may lead to a child contract storage layout corruption during an upgrade.

Paths: ./contracts/spell/BasicSpell.sol: __gap()

Recommendation: add a gap to the contract storage to allow future upgradability.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

H02. Invalid Calculations – Collateral Value Corruption

underlyingVaultShare is multiplied by *bToken.exchangeRateStored* independently on the result of *_isSoftVault(underlyingToken)*.

For the case *_isSoftVault(underlyingToken)* to be *true*, the *underlyingVaultShare* equals the amount of *bToken* and the multiplication is needed.

However, in case of *_isSoftVault(underlyingToken)* being *false* during the lend process, the tokens are deposited to the *HardVault* contract and *underlyingVaultShare* equals the amount of *underlyingToken*. Thus, the value should not be multiplied by *bToken.exchangeRateStored* in the target function.

This may lead to the function returning incorrect value and the position cannot be liquidated in time or is liquidated too early.

Paths: ./contracts/BlueBerryBank.sol: getIsolatedCollateralValue()

Recommendation: check the *_isSoftVault(underlyingToken)* value and do not perform the multiplication if it is *false*.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

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

Medium

M01. Unscalable Functionality – Immutable Position Token

The function provides the ability to (lend / borrow) (isolated collateral / debt token) to the specified position. However, there is no possibility for a change of the position's token in the future.

This may lead to the user being unable to provide wanted (isolated collateral / debt token) to the position and creating a new position wasting Gas.

Path: ./contracts/BlueBerryBank.sol: lend(), borrow()

Recommendation: allow (isolated collateral / debt token) reset if no isolated collaterals are deposited (the way it is implemented in the *putCollateral* function).

Found in: 42671b9

Status: Mitigated (implemented logic matches the expected result)

M02. Best Practice Violation – Zero Price Allowed

It is not checked that the received price is not zero.

This may lead to a broken oracle being considered a valid one.

Path: ./contracts/oracle/AggregatorOracle.sol: getPrice()

Recommendation: provide the corresponding check.

Found in: 42671b9

Status: Mitigated (according to the project architecture, it is designed to interact with the oracles only through the CoreOracle contract)

M03. Contradiction – Unchecked Relyment

The contract logic relies on the fact that *token decimals of the original and remapped tokens are the same*. However, the corresponding check is not implemented.

This may lead to the wrong price being returned by the oracle.

Path: ./contracts/oracle/ChainlinkAdapterOracle.sol: setTokenRemappings()

Recommendation: provide the corresponding check.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

M04. Contradiction – Zero Price Allowed

The function does not fully check that the token is supported. The check against a zero price is missing.

This may lead to the wrong assumptions about the oracle's ability to provide the token price.

Path: ./contracts/oracle/CoreOracle.sol: isWrappedTokenSupported()

Recommendation: use the *isTokenSupported* function over the *uToken* value to be consistent with the check.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

M05. Contradiction – Requirement Violation

The function allows requesting prices of unsupported tokens. It does not check if the token is whitelisted. However, the contract should not process an unsupported token as a valid one.

This may lead to the wrong assumption of the token status.

Path: ./contracts/oracle/CoreOracle.sol: getPositionValue()

Recommendation: provide the corresponding check.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

M06. Contradiction – Unchecked Relyment

The function logic relies on the fact that *the underlying token is an LP token, and it always has 18 decimals*. However, the corresponding check is not implemented.

This may lead to the wrong price being returned by the oracle.

Path: ./contracts/oracle/CoreOracle.sol: getPositionValue()

Recommendation: use the *getTokenValue* function to calculate the correct price.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

M07. Contradiction – Unvalidated TWAP Period

The value received from *vault.twapPeriod()* is not validated, however, it is used as a parameter for the *UniV3WrappedLibMockup.consult* call.

This may lead to the received price being affected by an attacker.

Path: ./contracts/oracle/IchiVaultOracle.sol: twapPrice0InToken1()

Recommendation: make sure that the value is big enough that an attacker can not affect the result.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

M08. Contradiction – Unchecked Relyment

The function logic relies on the vault token having 18 decimals. However, this fact is not checked.

This may lead to the wrong price being returned by the oracle.

Path: ./contracts/oracle/IchiVaultOracle.sol: getPrice()

Recommendation: use *10 ** vault.decimals()* instead of *10e18* to calculate the correct price.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

M09. Contradiction – Unchecked Relyment

The contract highly relies on the *stablePools[token]* pool including the *token* as one of the assets pair. However, the function does not check this fact.

This may lead to the wrong price being returned by the oracle.

Path: ./contracts/oracle/UniswapV3AdapterOracle.sol: setStablePools()

Recommendation: provide the corresponding check.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

M10. Contradiction – Unvalidated TWAP Period

The *maxDelayTimes* functionality is implemented to prevent outdated prices received by users. However, in the function, it is used as a *secondsAgo* value for calculating *arithmeticMeanTick* which in turn is used to get a price from *OracleLibrary*.

secondsAgo is allowed to be 10 seconds which is a low value and could be influenced by an attacker.

This may lead to wrong assumptions on the *setMaxDelayTimes* function purpose, and the wrong price being returned by the oracle.

Path: ./contracts/oracle/UniswapV3AdapterOracle.sol: getPrice()

Recommendation: provide declarative namings and set a reasonable *secondsAgo* minimal value to prevent the price from being influenced.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

M11. Best Practice Violation – Denial of Service

There are some tokens that reject non-zero approves if the current allowance is non-zero. The functions perform approvals without a previous reset.

This may lead to a Denial of Service contract state during such tokens are interacted with.

There is an *_ensureApprove* mechanic implemented in different ways in different contracts. This makes it unclear if the approval reset is performed.

Paths:
./contracts/BlueBerryBank.sol: _ensureApprove()
./contracts/spell/BasicSpell.sol: _ensureApprove()
./contracts/vault/SoftVault.sol: _ensureApprove()
./contracts/wrapper/WIchiFarm.sol: _ensureApprove()

Recommendation: move the *_ensureApprove* functionality to a separate module, and implement an approve reset there.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

M12. Best Practice Violation – Unpausable Oracles

The price data feeds could not be paused if an emergency situation on the data provider happens.

This may lead to the inability to quickly pause a specified oracle to prevent total system harm.

Paths:
./contracts/oracle/IchiVaultOracle.sol: getPrice()
./contracts/oracle/UniswapV2Oracle.sol: getPrice()

Recommendation: implement an ability to pause the *getPrice* functionality of the oracles.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

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

Low

L01. Mixing Whitelists

The whitelist functionality is provided in the *BlueBerryBank* contract. However, the function uses a whitelist set up on the *CoreOracle* contract.

This may lead to the wrong assumption on which tokens are whitelisted and allowed to be used as collaterals or isolated collaterals.

Paths:
./contracts/BlueBerryBank.sol: putCollateral()
./contracts/oracle/CoreOracle.sol: setWhitelistERC1155()

Recommendation: implement whitelists on one contract and provide corresponding namings which represent the group of tokens managed by the whitelist.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L02. Redundant Statements

The variables are never used and could be removed.

Path: ./contracts/libraries/UniV3/UniV3WrappedLib.sol: MIN_SQRT_RATIO, MAX_SQRT_RATIO

Recommendation: remove the redundant statements.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L02-1. Redundant Statements

The variables are never used and could be removed.

Path: ./contracts/libraries/UniV3/UniV3WrappedLibMockup.sol: MIN_SQRT_RATIO, MAX_SQRT_RATIO

Recommendation: remove the redundant statements.

Found in: 42671b9

Status: Mitigated (the values are needed for project tests and are native UniswapV3 constants)

L03. Parameters Order Dependence

The function result value depends on the order of the *price0* and *price1* values processed.

It may happen that *price0* is always used as the denominator.

Example: for *price0 = 90* & *price1 = 100* & *maxDeviation = 10%* the result is negative, however, the result is positive for *price0 = 100* & *price1 = 90* & *maxDeviation = 10%*.

Paths:
./contracts/oracle/AggregatorOracle.sol: _isValidPrices()
./contracts/oracle/IchiVaultOracle.sol: _isValidPrices()

Recommendation: use the smallest/biggest value as the denominator to make the function result independent of the order of the values.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L04. Unscalable Functionality

The contract allows setting remapped tokens. However, it does not allow clearing the remapped token addresses.

Although it is possible to set the token to be remapped to itself, it uses contract storage and makes the remapping management process unclear.

Path: ./contracts/oracle/ChainlinkAdapterOracle.sol: setTokenRemappings()

Recommendation: remove the check preventing a remapped token from being *0x0*.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L05. Redundant Using Statement

The using statement is redundant as the library is not used directly for the specified type.

Path: ./contracts/oracle/UniswapV2Oracle.sol
- using BBMath for uint256

Recommendation: remove the redundant statement.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L06. Unscalable Functionality

The contract allows setting routes to data providers for each of the supported tokens. However, it does not allow clearing the route addresses.

Although it is possible to set a route to a nonexistent address, it uses contract storage and makes the route management process unclear.

Path: ./contracts/oracle/CoreOracle.sol: setRoutes()

Recommendation: remove the check preventing a route from being *0x0*.

Found in: 42671b9

Status: Mitigated (according to the client such logic matches the expected behavior)

L07. Legacy Check

answeredInRound is a legacy variable and should not be used for price freshness validation.

Path: ./contracts/oracle/ChainlinkAdapterOracle.sol: getPrice()

Recommendation: remove the legacy variable usage.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L08. Mixing Functionality Purposes

The liquidation threshold management functionality is unrelated to the oracling of any data. It could be moved to a special configuration contract to clarify the contract's purpose.

Path: ./contracts/oracle/CoreOracle.sol: setLiqThresholds()

Recommendation: split the contract to clarify functionality purposes.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L09. Best Practice Violation

The system of the contracts has the *PRICE_PRECISION* constant value, but the constant is not widely used across the project.

Paths:
./contracts/oracle/ChainlinkAdapterOracle.sol: getPrice()
./contracts/oracle/CoreOracle.sol: getPositionValue()
./contracts/oracle/IchiVaultOracle.sol: getPrice()
./contracts/wrapper/WIchiFarm.sol: pendingRewards()
./contracts/BlueBerryBank.sol: getIsolatedCollateralValue()

Recommendation: replace the price precision *1e18* value with a constant.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L10. Code Duplication

The functionality of price deviation in bounds is implemented twice with the same code and purpose.

Paths:
./contracts/oracle/AggregatorOracle.sol: _isValidPrices(), _setPrimarySources()
./contracts/oracle/IchiVaultOracle.sol: _isValidPrices(), setPriceDeviation()

Recommendation: move the functionality to a separate module, avoid code duplications.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L11. Missing Implementation Initialization

According to the upgradable contracts pattern documentation, it is recommended to disable the possibility of initialization on the logic contract.

It may be done by adding the constructor to the target code.

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() {
    _disableInitializers();
}
```

Paths:
./contracts/ProtocolConfig.sol
./contracts/BlueBerryBank.sol
./contracts/wrapper/WIchiFarm.sol
./contracts/wrapper/WERC20.sol
./contracts/vault/SoftVault.sol
./contracts/vault/HardVault.sol
./contracts/spell/IchiSpell.sol
./contracts/oracle/CoreOracle.sol

Recommendation: add a *constructor* to the target code to prevent logic contracts from being overtaken.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L12. Misleading Names

The function is named *getPositionValue*, however, it does not correspond to the contract's purpose. It could be renamed to *getWrappedTokenValue*.

The return parameters are named *positionValue* and *debtValue*, however, they do not correspond to the function's purposes. They could be named *wrappedTokenValue* and *tokenValue* correspondingly.

Path: ./contracts/oracle/CoreOracle.sol: getPositionValue(), getTokenValue()

Recommendation: provide correct naming representing the object's purpose.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L13. Redundant Imports

The import statements are redundant as the imported objects are unused.

Paths:
./contracts/oracle/IchiVaultOracle.sol
- @uniswap/v3-periphery/contracts/interfaces/ISwapRouter.sol

./contracts/oracle/UniswapV2Oracle.sol
- @uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol

./contracts/oracle/UniswapV3AdapterOracle.sol
- @openzeppelin/contracts/access/Ownable.sol

./contracts/wrapper/WIchiFarm.sol
- @openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol

./contracts/vault/SoftVault.sol
- ../utils/BlueBerryConst.sol

./contracts/vault/HardVault.sol
- ../utils/BlueBerryConst.sol

./contracts/spell/BasicSpell.sol
- ../interfaces/IWETH.sol

Recommendation: remove the redundant statements.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L14. Redundant Actions

int24 twapTick received from *UniV3WrappedLibMockup* is implicitly upcasted to *int256* during an assignment and then is explicitly downcasted back to *int24*.

The *bank.getPositionInfo(bank.POSITION_ID())* call is performed twice, however, it is not necessary.

Paths:
./contracts/oracle/IchiVaultOracle.sol: twapPrice0InToken1()
./contracts/spell/IchiSpell.sol: reducePosition()

Recommendation: remove the redundant actions.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L15. Missing Zero Address Validation

Address parameters are being used without checking against the possibility of *0x0*.

Paths:
./contracts/spell/IchiSpell.sol: initialize()
./contracts/spell/BasicSpell.sol: __BasicSpell_init()
./contracts/wrapper/WIchiFarm.sol: initialize()

Recommendation: provide a check against *0x0*.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L16. Duplicated Hardcoded Values

The contracts system has constant values; it is widely used across some contracts.

This may lead to typos during further development, which may affect the project's functionality.

Path: ./contracts/ProtocolConfig.sol: setDepositFee(), setWithdrawFee(), setMaxSlippageOfClose(), setRewardFee()

Recommendation: move the commonly used values to the *BlueBerryConst* file.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L17. Code Duplication

ERC20 being wrapped to an ERC1155 functionality is implemented twice.

Paths:
./contracts/vault/HardVault.sol: _encodeTokenId(), _decodeTokenId(), balanceOfERC20(), getUnderlyingToken()
./contracts/wrapper/WERC20.sol: _encodeTokenId(), _decodeTokenId(), balanceOfERC20(), getUnderlyingToken()

Recommendation: move the functionality to a separate module, avoid code duplications.

Found in: 42671b9

Status: Mitigated (different implementations of the mentioned functions would be introduced in the future, as a result, such functionality presented in a separate module may be confusing)

L18. Default Visibility

Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.

Path: ./contracts/FeeManager.sol: config

Recommendation: provide visibility levels consciously.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

L19. Best Practice Violation

Same code style over the project is broken. The project mixes default `approve` and custom `_ensureApprove` functions.

Paths:
./contracts/BlueBerryBank.sol: withdrawLend()
./contracts/spell/IchiSpell.sol: openPositionFarm()
./contracts/vault/HardVault.sol: withdraw()
./contracts/wrapper/WIchiFarm.sol: _ensureApprove()

Recommendation: use the universal `_ensureApprove` functionality to increase token transfer allowance.

Found in: 42671b9

Status: Fixed (Revised commit: 1304035)

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

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.