HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Blueberry Foundation
Date: January 17th, 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	Evgeniy Bezuglyi | SC Audits Department Head at Hacken OU
Type	DeFi lending protocol
Platform	EVM
Language	Solidity
Methodology	Link
Changelog	25.10.2022 - Initial Review 29.11.2022 - Second Review 08.12.2022 - Third Review 16.12.2022 - Fourth Review 17.01.2023 - Fifth 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 is smart contracts in the repository:

Initial review scope

Repository	https://github.com/Blueberryfi/blueberry-core
Commit	ea837f348ab12e073b99e5e93fbc95e5052c2608
Functional Requirements	Liquidation Logic Blueberry Liquidation Model General Overview
Technical Requirements	Is not provided

Contracts:

File: ./contracts/BlueBerryBank.sol
SHA3: 5507564885215ded473a6af1a289a96b3d6db79da997e7613613c1b416312665

File: ./contracts/Governable.sol
SHA3: 9239d1066cb7b8cde439123e6d29cfd701903c99278380318ca4718ca6964540

File: ./contracts/interfaces/band/IStdReference.sol
SHA3: 9aef1410e7c7be6ee8ea0696fd14d5f50351588a9358c07b491630f7fe216b11

File: ./contracts/interfaces/chainlink/IFeedRegistry.sol
SHA3: 9426ebf3c16db0ce33b495c2c5ee5c7fa458252358e93eed4caaca2046be3ee6

File: ./contracts/interfaces/compound/ICErc20.sol
SHA3: 88e9f0951a78c3bd3be86b118b3e9e84209d6392cb2d2a590708d95190d78ed3

File: ./contracts/interfaces/IBank.sol
SHA3: 94ad2a4692f9949cc3b7de74268d7589eb79c7aa6322c38c6022b0c337a40098

File: ./contracts/interfaces/IBaseOracle.sol
SHA3: a8d065db9f6774297d08bdce807f0fb2d1bf3171c1bd07fa0433e209fd93c25e

File: ./contracts/interfaces/ichi/IIchiFarm.sol
SHA3: 16120767b516255a833a8fc5c382ab838655af517c3c12578e2dc7f5386f6610

File: ./contracts/interfaces/ichi/IICHIVault.sol
SHA3: 39f71a3e4ba33148eb225f7ed369662b804dc528ce9076d69bd889725d5a7108

File: ./contracts/interfaces/IERC20Wrapper.sol
SHA3: 124c36f2b46959be944521721a1e0113afd3a1f068996233922a5da4b0ee9a66

File: ./contracts/interfaces/IOracle.sol
SHA3: 19aa3afac78fe6f46703ba74ff507250d594aabae31d913843cf8c55accc69fe

File: ./contracts/interfaces/ISafeBox.sol
SHA3: 834571d0e3a57703483adbac44b985c19e1b0f03a8fff1629697d4cbf5c2b990

File: ./contracts/interfaces/IWERC20.sol
SHA3: 56a81f68e519a3b19f20b912b88f35d4b7e63d4de4009f03a83025db28afea12

File: ./contracts/interfaces/IWETH.sol
SHA3: 30f4f6eee2cc3c1765b1079eeb1249cca2d7fdf7e3f546b1eff5ed6f066880b8

File: ./contracts/interfaces/IWIchiFarm.sol
SHA3: 39eabe230cda20740f354b795612c8544099455c5a657f058e1f7f76f1c7752e

File: ./contracts/interfaces/uniswap/v3/IUniswapV3Pool.sol
SHA3: c98fa4639033d185797166d2251fde4a03f9c40944aae8eff4eed81e6545f1a4

File: ./contracts/interfaces/uniswap/v3/IUniswapV3SwapCallback.sol
SHA3: f7d19b9f5f39f508b7b69d6d8e1b8715febffa02a300928eae912a71621c39a0

File: ./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolActions.sol
SHA3: e950d5d54539ccd76b7819034831a9f6de7f94c518d6fbaf2355c4bff92464a4

File:
./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolDerivedState.sol
SHA3: 0bc5b04159a7fb067d8827545a982adf1a7da9b732624cfabb496615c9f993a2

File: ./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolEvents.sol
SHA3: ab4eb3f3006341ccfe156a714740ee6ebfe229de44908a3307a4b275b3ba24c0

File: ./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolImmutables.sol
SHA3: 7b7b446c870fda15525e85270834eeb6cc8422e560fc17cd52d331fc72dc060c

File:
./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolOwnerActions.sol
SHA3: 89061eefd33f5ba3b760f25cda2fa8771f0f2d2964d2c0f60de5a9b17db92538

File: ./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolState.sol
SHA3: 52af630bdc851093296900b9959a26e40593d89940e88b5b33ceef385ded9b5f

File: ./contracts/libraries/UniV3/FixedPoint96.sol
SHA3: 27e69bcae8e943dc738f2bdc10acca33d33f9b90d3b7dd8bdcd45cf674eb1936

File: ./contracts/libraries/UniV3/FullMath.sol
SHA3: 7bccf5b61188ca6ee6b504396946829aa72e0ee76798adab8fffee9b9b873763

File: ./contracts/libraries/UniV3/LiquidityAmounts.sol
SHA3: a71d45532db06314ff7316905f027ec217ed247b5c48a36a64df14ac6b3251b0

File: ./contracts/libraries/UniV3/LowGasSafeMath.sol
SHA3: 980ea4be6c0302bef2fa00e97864ee97cfe34dff36604a42cdaed27cac7c6b77

File: ./contracts/libraries/UniV3/OracleLibrary.sol
SHA3: cd8acfb641163b5b694081acb853bfbaa339ff48e36dde77616179ec17caa8ad

File: ./contracts/libraries/UniV3/PoolActions.sol
SHA3: 877f54f8df2be8dd22ad038c36ee4842cf211873ba05b457cb91782b10fe354f

File: ./contracts/libraries/UniV3/PoolVariables.sol
SHA3: 68625c36c22782f3c9d6687eb119971389acf270c51f5bc4bc3aabd8bb9ab97e

File: ./contracts/libraries/UniV3/PositionKey.sol
SHA3: bf9c0b5fb3b3dd97c346bad60e07b47ca2ce42b7756f33935d72f832a92923c4

File: ./contracts/libraries/UniV3/SafeCast.sol
SHA3: e7d1147dc9518a03d561f90b4299da723b52d7aa9855321e50b54af31a592f86

File: ./contracts/libraries/UniV3/SqrtPriceMath.sol
SHA3: 9c5440f8e16aeee522c1586716c2e5ff0072c592ead44e0de4178477566534fc

File: ./contracts/libraries/UniV3/TickMath.sol
SHA3: 0dbdf1daf34270e97efb0bfb37d6ac6669d205dbf8f1b265486fac4271f6888f

File: ./contracts/libraries/UniV3/UnsafeMath.sol
SHA3: 932eb260ce03331507e0e50350599cfbf6c1d15c55988b8a463e3d2166fe173e

File: ./contracts/oracle/AggregatorOracle.sol
SHA3: 0b8101430ccc4d62c8ca448cf2dceea158e190665934f065da46ab98c7c2b3b6

File: ./contracts/oracle/BandAdapterOracle.sol
SHA3: 9ba9eca976becec440ab27bc2658972e303d627829f4131b1054d73e123697e1

File: ./contracts/oracle/ChainlinkAdapterOracle.sol
SHA3: bb4d1f4359acf77aac519df2008d9154b1dcbf0a17504d8e660d3687e7e0c909

File: ./contracts/oracle/CoreOracle.sol
SHA3: 29dad940a51bae2797043e199d42c6ea71c1dde66fa612dc0b41baa05e8b00e4

File: ./contracts/oracle/IchiLpOracle.sol
SHA3: a7b79f051a4a77dd10eb3dfad45b29eabb6f1a1a6eaf7a6f5d2b7e7c04556965

File: ./contracts/oracle/ProxyOracle.sol
SHA3: f11b712c93e50e0ab0e051480ff76f7f76e867eca50fc7f786c821b582970705

File: ./contracts/oracle/UniswapV3AdapterOracle.sol
SHA3: 2aa0f67340c6ba55ab3440a667e9fd355e7bc4d56a6bad488037f9c5e554f607

File: ./contracts/oracle/UsingBaseOracle.sol
SHA3: d31e3dc7a5a9dc4a77b12875d0ff3edb583dee3e0728990a9a5ddd70e6ce1915

File: ./contracts/SafeBox.sol
SHA3: 57ffc7643551b024dcf8c7f7ae5f4a9eb9b257d66f76dd91690c3f37e62485c3

File: ./contracts/spell/BasicSpell.sol
SHA3: 47427358470f01b8344d1bd192a4921c4b12149eb451b542bf4e3f23cc3a8b64

File: ./contracts/spell/IchiVaultSpell.sol
SHA3: caa06dec17cd87b7d8b767dcb270a79ec8bb01229e27d26736d539421c8f62ee

File: ./contracts/spell/WhitelistSpell.sol
SHA3: f0170754c7cb1929f192ca0e6703a858a2f95ee93ecb52e6d903967195e8cb3a

File: ./contracts/utils/BBMath.sol
SHA3: 49e4050fe27b915febf384c01605b9ef9ae066dc997f456e52bee2eaae0bb6e8

File: ./contracts/utils/BConst.sol
SHA3: 4fc3bcb5f0f2f913b77baeb14a7344317549b701b6125fe27abb87711ef96137

```
File: ./contracts/utils/BNum.sol
SHA3: 7eb0023f15f4b87a708e62d616c71b9dfffb6dd1940c8b324baeb40d77ef8735

File: ./contracts/utils/ERC1155NaiveReceiver.sol
SHA3: 165a27c35e1ffd38f66492fcec1fa07afe8b609b83d9c489c90ddbafe732c0b6

File: ./contracts/wrapper/WERC20.sol
SHA3: dd8693e31ea5cc5aa3fe01b5ea8214371efe05c48d7fda0d514fabe8fdbb4e94

File: ./contracts/wrapper/WIchiFarm.sol
SHA3: 16bb0ea3900adde6f6489f9db3c9ac30c127a7844151f8006b835a6e7495d787
```

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	ea837f348ab12e073b99e5e93fbc95e5052c2608
Functional Requirements	Liquidation Logic Blueberry Liquidation Model General Overview
Technical Requirements	Is not provided

Contracts:

File: ./contracts/BlueBerryBank.sol
SHA3: 1eca8fec26afc41c00e93acfcb114cf2900747b904689ae467fdbe6cd662b2aa

File: ./contracts/BlueBerryErrors.sol
SHA3: 4f2af23dbc6197823fe1feda3051e7cf0f12c9dafee517e20d8f2420ff4c0a37

File: ./contracts/interfaces/band/IStdReference.sol
SHA3: 0f0355c059c5c8a0d407b42a5aac60d25e807917123f9f35a6978732af34615e

File: ./contracts/interfaces/chainlink/IFeedRegistry.sol
SHA3: 8ca9ce016071ca31dcfd31e1834f636d7ffba517d5aa63e47739fbdd92150614

File: ./contracts/interfaces/compound/ICErc20.sol
SHA3: 57926c4db8449aa467eb93ec520652bd08b1a0c10e87a51b21c68b65b6a58898

File: ./contracts/interfaces/IBank.sol
SHA3: 828fe20c972b230833b48442db455aa6e8378c371697f48e37d6fac8615bc8be

File: ./contracts/interfaces/IBaseOracle.sol
SHA3: 1db54613a110c42336a1f00b14699f86dedf950eed8997f844be8e92d94a2ff0

File: ./contracts/interfaces/ichi/IIchiFarm.sol
SHA3: 93a2a992238d541ee423eb82f4f21d56ef549e90f2879cb359dad016de708ad2

File: ./contracts/interfaces/ichi/IICHIVault.sol
SHA3: 0ee42a97b4d028aed773067d0dbc8e1d8bbcf4a660d7a300cd702b28157bed7a

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

File: ./contracts/interfaces/IERC20Wrapper.sol
SHA3: 5e70bfb5126232fdda15f38f95baa1eff1b6cfa6da1fa4802f076dc7d7c99a08

File: ./contracts/interfaces/IOracle.sol
SHA3: 32f2fbebb98b7d0268007464b001731248ee5087dd5498d97253abd0ffbf7714

File: ./contracts/interfaces/IProtocolConfig.sol
SHA3: f06d7c3be81c57b3323534b17eea6b64f14fa5876b17b2bb593880df6b89739b

File: ./contracts/interfaces/ISafeBox.sol
SHA3: b8e2f4d2578cafa664df86acc6e0ec4f154af813d5257cca4d9232ee85d64aef

File: ./contracts/interfaces/IWERC20.sol
SHA3: 654a86c582b5819881339ef13ce004b64733bff68326b137588be0ca17570bea

```
File: ./contracts/interfaces/IWETH.sol
SHA3: 7081cbd847c3a89c76b55c1691855822dee655e2f8467037a1e8aec169fc3a3b

File: ./contracts/interfaces/IWIchiFarm.sol
SHA3: 49bf251356eea49824b2e9065caee1989fca859d5747ef86d76a353dcb3d0da6

File: ./contracts/interfaces/uniswap/v3/IUniswapV3Pool.sol
SHA3: c98fa4639033d185797166d2251fde4a03f9c40944aae8eff4eed81e6545f1a4

File: ./contracts/interfaces/uniswap/v3/IUniswapV3SwapCallback.sol
SHA3: 3f78a57a3faea6f932f5327c3c2b2f4981bb55e680ce0b2d3ed300af1aa0aa59

File: ./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolActions.sol
SHA3: e950d5d54539ccd76b7819034831a9f6de7f94c518d6fbaf2355c4bff92464a4

File:
./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolDerivedState.sol
SHA3: 0bc5b04159a7fb067d8827545a982adf1a7da9b732624cfabb496615c9f993a2

File: ./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolEvents.sol
SHA3: ab4eb3f3006341ccfe156a714740ee6ebfe229de44908a3307a4b275b3ba24c0

File: ./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolImmutables.sol
SHA3: 7b7b446c870fda15525e85270834eeb6cc8422e560fc17cd52d331fc72dc060c

File:
./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolOwnerActions.sol
SHA3: 89061eefd33f5ba3b760f25cda2fa8771f0f2d2964d2c0f60de5a9b17db92538

File: ./contracts/interfaces/uniswap/v3/pool/IUniswapV3PoolState.sol
SHA3: 52af630bdc851093296900b9959a26e40593d89940e88b5b33ceef385ded9b5f

File: ./contracts/libraries/BBMath.sol
SHA3: 23620274b6f4defed929541c349fc60a3cea3074ecb7fe05637d3e7058b8fd69

File: ./contracts/libraries/UniV3/FixedPoint96.sol
SHA3: 27e69bcae8e943dc738f2bdc10acca33d33f9b90d3b7dd8bdcd45cf674eb1936

File: ./contracts/libraries/UniV3/FullMath.sol
SHA3: 7bccf5b61188ca6ee6b504396946829aa72e0ee76798adab8fffee9b9b873763

File: ./contracts/libraries/UniV3/LiquidityAmounts.sol
SHA3: a71d45532db06314ff7316905f027ec217ed247b5c48a36a64df14ac6b3251b0

File: ./contracts/libraries/UniV3/LowGasSafeMath.sol
SHA3: 9f00076b6e72340de1874d00408673769792995O4f1da88c629ad9039acd63e6

File: ./contracts/libraries/UniV3/OracleLibrary.sol
SHA3: cd8acfb641163b5b694081acb853bfbaa339ff48e36dde77616179ec17caa8ad

File: ./contracts/libraries/UniV3/PoolActions.sol
SHA3: 2eb209c8938d807ab32f7035e3056a066fbaafed2af58778f2080703b300f164

File: ./contracts/libraries/UniV3/PoolVariables.sol
SHA3: 4f1daf1adfb3e2e5d1125f83d670253e3d5254cebc18f49c1f363d05d4dc9718

File: ./contracts/libraries/UniV3/PositionKey.sol
SHA3: bf9c0b5fb3b3dd97c346bad60e07b47ca2ce42b7756f33935d72f832a92923c4
```

File: ./contracts/libraries/UniV3/SafeCast.sol
SHA3: e7d1147dc9518a03d561f90b4299da723b52d7aa9855321e50b54af31a592f86

File: ./contracts/libraries/UniV3/SqrtPriceMath.sol
SHA3: 5fc27340ea5dd3508edc463298536a5aa4f06273742b8d0ceddd66b38c340222

File: ./contracts/libraries/UniV3/TickMath.sol
SHA3: 0dbdf1daf34270e97efb0bfb37d6ac6669d205dbf8f1b265486fac4271f6888f

File: ./contracts/libraries/UniV3/UnsafeMath.sol
SHA3: 932eb260ce03331507e0e50350599cfbf6c1d15c55988b8a463e3d2166fe173e

File: ./contracts/oracle/AggregatorOracle.sol
SHA3: 1c41764e47040632794cc49cd1e77d7f489d309daed0123786ddbf60f2583141

File: ./contracts/oracle/BandAdapterOracle.sol
SHA3: 4d655bc0009483fe318b2d7544deea6e39a5f275ff6206dde66cb326dac61ceb

File: ./contracts/oracle/ChainlinkAdapterOracle.sol
SHA3: 634e36a842314599c2c07e483d2f9fa417f0d1ea669270c0e11b840cd1eb57ee

File: ./contracts/oracle/CoreOracle.sol
SHA3: 13124f345d8749e3be16e235e689e702bb365fa4ade8d19108b30903377fc860

File: ./contracts/oracle/IchiLpOracle.sol
SHA3: 47ceb732a050bc6445d5a3b655cebf82607b6a86d70d49eb695a94665c75729b

File: ./contracts/oracle/UniswapV3AdapterOracle.sol
SHA3: 3ea85f026a4c6d82c1aad52203a5bf6efdda6efe0f1a645e74ab06a0f0193f91

File: ./contracts/oracle/UsingBaseOracle.sol
SHA3: 6a29eece2b3c956c4b3b75e686b6261bde72879740cc005504950c32a1c86492

File: ./contracts/ProtocolConfig.sol
SHA3: 6ff3e8d353143b5cb1438a63f9fc18cc5f9735bf23c914ad571586d0cba49e41

File: ./contracts/SafeBox.sol
SHA3: 5a38591bb130a85d5e87fe8ac18ca028ded59ec34becdc817ae832760e66ce76

File: ./contracts/spell/BasicSpell.sol
SHA3: 618014a31ee568b468fad174c1b1226bb61156a9e836095b844062fa214c88a8

File: ./contracts/spell/IchiVaultSpell.sol
SHA3: cef7a00dbae2c6d56071d53a61b52ee5411639011638e117e3ddc32f4f984d22

File: ./contracts/spell/WhitelistSpell.sol
SHA3: 3c1758538edaa2aa568c936b9bb369b14ce0021fc4db9a7a2964e94648ddb540

File: ./contracts/utils/ERC1155NaiveReceiver.sol
SHA3: 38ca0ae472d8e2f2a39d2fbc4bf2c046261d465a4dcfb059f4102e81c3c6e7bd

File: ./contracts/wrapper/WERC20.sol
SHA3: aa970d9d3ace441e69aaabd38ae84691219f4b953563fcf10047a3d5f65df93b

File: ./contracts/wrapper/WIchiFarm.sol
SHA3: 8a66045b8fadbb22d57bca7ada689ee4751648d06a81605ae4375df31398fa8b

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

Third review scope

Repository	https://github.com/Blueberryfi/blueberry-core/
Commit	e29604381923f4ac32206b28d3db72fc39e79795
Functional Requirements	Liquidation Logic Blueberry Liquidation Model General Overview
Technical Requirements	Is not provided

Contracts:

File: ./contracts/BlueBerryErrors.sol
SHA3: 36ee0eb9fb3d6a8d276bf7689bd3da2d1922ebffe7e032adce77c623697d2b6b

File: ./contracts/interfaces/band/IStdReference.sol
SHA3: 0f0355c059c5c8a0d407b42a5aac60d25e807917123f9f35a6978732af34615e

File: ./contracts/interfaces/chainlink/IFeedRegistry.sol
SHA3: 8ca9ce016071ca31dcfd31e1834f636d7ffba517d5aa63e47739fbdd92150614

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

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

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

File: ./contracts/interfaces/compound/IComptroller.sol
SHA3: 4dbfa976dba22e1be6a071317f6f29c4e752c53fedd173801211441178517995

File: ./contracts/interfaces/IBaseOracle.sol
SHA3: 1db54613a110c42336a1f00b14699f86dedf950eed8997f844be8e92d94a2ff0

File: ./contracts/interfaces/ichi/IICHIVault.sol
SHA3: 0ee42a97b4d028aed773067d0dbc8e1d8bbcf4a660d7a300cd702b28157bed7a

File: ./contracts/interfaces/IERC20Wrapper.sol
SHA3: 5e70bfb5126232fdda15f38f95baa1eff1b6cfa6da1fa4802f076dc7d7c99a08

File: ./contracts/interfaces/IOracle.sol
SHA3: 32f2fbebb98b7d0268007464b001731248ee5087dd5498d97253abd0ffbf7714

File: ./contracts/interfaces/ISafeBox.sol
SHA3: bc2dce9223cb3dcdea0be8f8059aaaaa4b2fd58eced13bec84647bed7361e396

File: ./contracts/interfaces/uniswap/v2/IUniswapV2Pair.sol
SHA3: 7b2ce01e3c2e094df0c75e34b6d313f19b5ef14be2a35e011de7efb218704395

File: ./contracts/libraries/BBMath.sol
SHA3: 23620274b6f4defed929541c349fc60a3cea3074ecb7fe05637d3e7058b8fd69

File: ./contracts/libraries/UniV3/FullMath.sol
SHA3: 7bccf5b61188ca6ee6b504396946829aa72e0ee76798adab8fffee9b9b873763

```
File: ./contracts/libraries/UniV3/SafeCast.sol
SHA3: e7d1147dc9518a03d561f90b4299da723b52d7aa9855321e50b54af31a592f86

File: ././contracts/libraries/UniV3/TickMath.sol
SHA3: 0dbdf1daf34270e97efb0bfb37d6ac6669d205dbf8f1b265486fac4271f6888f

File: ./contracts/oracle/AggregatorOracle.sol
SHA3: 1c41764e47040632794cc49cd1e77d7f489d309daed0123786ddbf60f2583141

File: ./contracts/oracle/BandAdapterOracle.sol
SHA3: 4d655bc0009483fe318b2d7544deea6e39a5f275ff6206dde66cb326dac61ceb

File: ./contracts/oracle/ChainlinkAdapterOracle.sol
SHA3: 634e36a842314599c2c07e483d2f9fa417f0d1ea669270c0e11b840cd1eb57ee

File: ./contracts/oracle/CoreOracle.sol
SHA3: ee44b450d93d3255c27e300c4bd377a6aa290769b3d10106671bc060cf58614f

File: ./contracts/oracle/IchiLpOracle.sol
SHA3: 1989948f4c61dc74eaf57155ea2c804753b6031063f0e44640f440e3e77b9c9c

File: ./contracts/oracle/UniswapV2Oracle.sol
SHA3: 1f93fad7cbd68aa3830d7b1dbc4a35ad057cb2f5e67449383c625d2a8d9cb379

File: ./contracts/oracle/UniswapV3AdapterOracle.sol
SHA3: 3ea85f026a4c6d82c1aad52203a5bf6efdda6efe0f1a645e74ab06a0f0193f91

File: ./contracts/oracle/UsingBaseOracle.sol
SHA3: 6a29eece2b3c956c4b3b75e686b6261bde72879740cc005504950c32a1c86492

File: ./contracts/interfaces/IProtocolConfig.sol
SHA3: 1c9ef4d4b81ba32a66434526302f9f53d70fcdd039aae8340d8a5afdeb1ce28d

File: ./contracts/ProtocolConfig.sol
SHA3: 246b43c6b0adba1c82eb20d0a07697036c28bea533e175a7411a74446d60cae2

File: ./contracts/SafeBox.sol
SHA3: 4dae9118aabb7961f5107c3df56789fd3707c4753a4d19b9dba499c26d69cfb6
```

Fourth review scope

Repository	https://github.com/Blueberryfi/blueberry-core/
Commit	325e796dde11e785f864f8b23e854e5d9372af36
Functional Requirements	Liquidation Logic Blueberry Liquidation Model General Overview
Technical Requirements	Is not provided

Contracts:

File: ./contracts/utils/BlueBerryErrors.sol
SHA3: 36ee0eb9fb3d6a8d276bf7689bd3da2d1922ebffe7e032adce77c623697d2b6b

File: ./contracts/interfaces/band/IStdReference.sol
SHA3: 0f0355c059c5c8a0d407b42a5aac60d25e807917123f9f35a6978732af34615e

File: ./contracts/interfaces/chainlink/IFeedRegistry.sol
SHA3: 8ca9ce016071ca31dcfd31e1834f636d7ffba517d5aa63e47739fbdd92150614

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

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

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

File: ./contracts/interfaces/compound/IComptroller.sol
SHA3: 4dbfa976dba22e1be6a071317f6f29c4e752c53fedd173801211441178517995

File: ./contracts/interfaces/IBaseOracle.sol
SHA3: 1db54613a110c42336a1f00b14699f86dedf950eed8997f844be8e92d94a2ff0

File: ./contracts/interfaces/ichi/IICHIVault.sol
SHA3: 0ee42a97b4d028aed773067d0dbc8e1d8bbcf4a660d7a300cd702b28157bed7a

File: ./contracts/interfaces/IERC20Wrapper.sol
SHA3: 5e70bfb5126232fdda15f38f95baa1eff1b6cfa6da1fa4802f076dc7d7c99a08

File: ./contracts/interfaces/IOracle.sol
SHA3: 32f2fbebb98b7d0268007464b001731248ee5087dd5498d97253abd0ffbf7714

File: ./contracts/interfaces/ISafeBox.sol
SHA3: bc2dce9223cb3dcdea0be8f8059aaaaa4b2fd58eced13bec84647bed7361e396

File: ./contracts/interfaces/uniswap/v2/IUniswapV2Pair.sol
SHA3: 7b2ce01e3c2e094df0c75e34b6d313f19b5ef14be2a35e011de7efb218704395

File: ./contracts/libraries/BBMath.sol
SHA3: 23620274b6f4defed929541c349fc60a3cea3074ecb7fe05637d3e7058b8fd69

File: ./contracts/libraries/UniV3/UniV3WrappedLib.sol
SHA3: 929bc6bc287c916981c46bc23a463637868221e91c12b15e502128cf492d7f21

File: ./contracts/oracle/BaseAdapter.sol
SHA3: 4a55583866edb0cd4899109f67eb8860842fcfce57bca6163ad21c57d56713b7

File: ./contracts/oracle/AggregatorOracle.sol
SHA3: f66c01c3c093ed26fad1b2134176029ade09a3619cf669c3b7f7215578d982e6

File: ./contracts/oracle/BandAdapterOracle.sol
SHA3: b5747cb0745938b2a337f534eff9703272dda9ec455001572349a51aa53281bd

File: ./contracts/oracle/ChainlinkAdapterOracle.sol
SHA3: c3b55ecb4789fa829049efc870338eb5554727b3363f3c1acddf544c27e1ad83

File: ./contracts/oracle/CoreOracle.sol
SHA3: 0f8148816ea71a0d9fdc926e6455594d3fa90e4edb6a9ec66b9080486fe661db

File: ./contracts/oracle/IchiLpOracle.sol
SHA3: 1989948f4c61dc74eaf57155ea2c804753b6031063f0e44640f440e3e77b9c9c

File: ./contracts/oracle/UniswapV2Oracle.sol
SHA3: 1f93fad7cbd68aa3830d7b1dbc4a35ad057cb2f5e67449383c625d2a8d9cb379

File: ./contracts/oracle/UniswapV3AdapterOracle.sol
SHA3: 69d944c957bc5695e5a7c55154f1416072bf772e3d3a7d87133b0e9bb835d66d

File: ./contracts/oracle/UsingBaseOracle.sol
SHA3: 6a29eece2b3c956c4b3b75e686b6261bde72879740cc005504950c32a1c86492

File: ./contracts/interfaces/IProtocolConfig.sol
SHA3: df697c81891bed4410ce473ceecef9d171db0ae6482b6efeea5010439051275f

File: ./contracts/ProtocolConfig.sol
SHA3: 57b3f31adc172a327ee86144405785b6c9e57a718a7619f015b0904a0b0599ed

File: ./contracts/SafeBox.sol
SHA3: d19dc706edcddc065582c9c3cda450624a38f13b6e5268b103fdfedfeed71eb7

File: ./contracts/utils/BlueBerryConst.sol
SHA3: 68e70898824c946bc84f0cc2a819cd0d253bfbceabb650b78fb4aa50e47d9b51

Fifth review scope

Repository	https://github.com/Blueberryfi/blueberry-core/
Commit	e1f2172db1432d2fae364bcd50fd3a2307d04dab
Functional Requirements	Liquidation Logic Blueberry Liquidation Model General Overview
Technical Requirements	Architecture
Contracts: File: ./contracts/BlueBerryBank.sol SHA3: c87d53bd4443143cbaf5b5489b463638a8baf70d26746f2afe0c4878eb7ddda9	

```
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_2.sol
SHA3: a8c03e372a6a8ae4c2329e7a7d5fba5d520dd2593a091b19065c7318eb403ed2

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

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

File: ./contracts/interfaces/compound/IComptroller.sol
SHA3: 4dbfa976dba22e1be6a071317f6f29c4e752c53fedd173801211441178517995

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/IAny.sol
SHA3: ee4c47b8c07112a76dd9cc1f9ca47a5abecab8fc9e51856c0a1c88c245dd646e

File: ./contracts/interfaces/IBank.sol
SHA3: 3a5263f3a60762d1f2753a18ed054651f861ee4faa36f5412e1f4c36e6ae47b2

File: ./contracts/interfaces/IBaseOracle.sol
SHA3: 1db54613a110c42336a1f00b14699f86dedf950eed8997f844be8e92d94a2ff0

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

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

File: ./contracts/interfaces/ichi/IICHIVault.sol
SHA3: 0e76bdde9ae73591b11b3b7bee882a3fe8b83d9dbce14c3b7ca6b2952af8ffd6

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

File: ./contracts/interfaces/IERC20Ex.sol
SHA3: 05608259d7bdd7995f5f546a38f6c377ab6efb9eaa38d4330fb9894b0ae93f8a

File: ./contracts/interfaces/IERC20Wrapper.sol
SHA3: 5e70bfb5126232fdda15f38f95baa1eff1b6cfa6da1fa4802f076dc7d7c99a08

File: ./contracts/interfaces/IERC20WrapperOld.sol
```

SHA3: 7ee88efe9ae420686e5f45ae3b2fee09f130ac08dbf4cf1451b2f6195797ced4

File: ./contracts/interfaces/IHardVault.sol
SHA3: bfe0ff185d086ab5149c60e70abc8ce009da9c8482efce75ffc454a0f2f85ade

File: ./contracts/interfaces/IOracle.sol
SHA3: 91f016cf8578a7774e8bcfe692a5deaa0551d17d090e28f7aa766adc64f10bc9

File: ./contracts/interfaces/IProtocolConfig.sol
SHA3: 9d64b63300440e7e111a030538c2567fd6e2979d79ae297853d5a94ff89eac8d

File: ./contracts/interfaces/ISoftVault.sol
SHA3: 8336c1880f00e205ca6aaf9b3736e8d780f7ef67f04ffcf83f75e34ba8087762

File: ./contracts/interfaces/IWERC20.sol
SHA3: 654a86c582b5819881339ef13ce004b64733bff68326b137588be0ca17570bea

File: ./contracts/interfaces/IWETH.sol
SHA3: 7081cbd847c3a89c76b55c1691855822dee655e2f8467037a1e8aec169fc3a3b

File: ./contracts/interfaces/IWIchiFarm.sol
SHA3: 49bf251356eea49824b2e9065caee1989fca859d5747ef86d76a353dcb3d0da6

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

File: ./contracts/interfaces/uniswap/v2/IUniswapV2Factory.sol
SHA3: f73ec3d907a7c64495d8ec9afd98add30fbe32b0870cdb3db3dca4193456bcfd

File: ./contracts/interfaces/uniswap/v2/IUniswapV2Pair.sol
SHA3: 7b2ce01e3c2e094df0c75e34b6d313f19b5ef14be2a35e011de7efb218704395

File: ./contracts/interfaces/uniswap/v2/IUniswapV2Router01.sol
SHA3: 423056684cea6faa22ac55085b5a2750c289ab4c21c4c2857f6e8b92ae094e5d

File: ./contracts/interfaces/uniswap/v2/IUniswapV2Router02.sol
SHA3: be50364d56afc809cc18bc4b4599ebe8b4400ae6d6be820c09829caa16321c1d

File: ./contracts/libraries/BBMath.sol
SHA3: 1282ffb3c4450e2153b6d6179358518f267f79d6bf72b24ad67935aa7ac83c50

File: ./contracts/libraries/UniV3/UniV3WrappedLib.sol
SHA3: 929bc6bc287c916981c46bc23a463637868221e91c12b15e502128cf492d7f21

File: ./contracts/oracle/AggregatorOracle.sol
SHA3: f66c01c3c093ed26fad1b2134176029ade09a3619cf669c3b7f7215578d982e6

File: ./contracts/oracle/BandAdapterOracle.sol
SHA3: b5747cb0745938b2a337f534eff9703272dda9ec455001572349a51aa53281bd

File: ./contracts/oracle/BaseAdapter.sol
SHA3: 4a55583866edb0cd4899109f67eb8860842fcfce57bca6163ad21c57d56713b7

File: ./contracts/oracle/ChainlinkAdapterOracle.sol
SHA3: c3b55ecb4789fa829049efc870338eb5554727b3363f3c1acddf544c27e1ad83

File: ./contracts/oracle/CoreOracle.sol
SHA3: b97fcf922e180192c11ce573e1a37b6f42d9bf04a29ee8809f11b39293573d13

```
File: ./contracts/oracle/IchiLpOracle.sol
SHA3: 1989948f4c61dc74eaf57155ea2c804753b6031063f0e44640f440e3e77b9c9c

File: ./contracts/oracle/UniswapV2Oracle.sol
SHA3: 9e05373aed9024fb2bfb3372b9fede8d8eb5e3cb3bcafd22bdbd8d9063693baf

File: ./contracts/oracle/UniswapV3AdapterOracle.sol
SHA3: 69d944c957bc5695e5a7c55154f1416072bf772e3d3a7d87133b0e9bb835d66d

File: ./contracts/oracle/UsingBaseOracle.sol
SHA3: 6a29eece2b3c956c4b3b75e686b6261bde72879740cc005504950c32a1c86492

File: ./contracts/ProtocolConfig.sol
SHA3: 22945a3b3a4345ac019c6d1c0f4c4dade2231ae76ca237cc626705c81c376750

File: ./contracts/spell/BasicSpell.sol
SHA3: 4967f5b17d52f218d49c6a22e713eb08c2e1bb7df3f8c45df0318e37f2ef75e0

File: ./contracts/spell/IchiVaultSpell.sol
SHA3: 1a655d805423836e65f91a3dbfbbcb2dc45d2586871057126ec5f2d3ed7138b9

File: ./contracts/utils/BlueBerryConst.sol
SHA3: 68e70898824c946bc84f0cc2a819cd0d253bfbceabb650b78fb4aa50e47d9b51

File: ./contracts/utils/BlueBerryErrors.sol
SHA3: 2e375b15dde0fde44c6829708789cc9d70ebe96ad15dd9acd8c583318820de90

File: ./contracts/utils/ERC1155NaiveReceiver.sol
SHA3: 1bd90cc7c49c75afa3e34010a519b88451d365b755b23aeea829de2068ba4988

File: ./contracts/vault/HardVault.sol
SHA3: f62ddb3d114f34d7ad12c01c21b7b34c36a9eb365a38ef3199bad274a960b1b4

File: ./contracts/vault/SoftVault.sol
SHA3: 8b811150da0d7d00ed3d43cfcc1317563cbcf2705c632fbd0b2b89402996c719

File: ./contracts/wrapper/WERC20.sol
SHA3: aae506a86c8c91c3cd80152fea7106a20a677ae1026a9f987e0b5a59ced29a58

File: ./contracts/wrapper/WIchiFarm.sol
SHA3: 267309ccdf855b05143f17eeaccda0814dc3b27562cf11a5542d645197e4662e
```

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.

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 **6** out of **10**.

- Functional requirements are superficial.
- Description and UML diagrams of the system architecture are outdated.
- Technical description is not provided.
- It is intuitive how to run tests and the coverage tool.

Code quality

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

- No code applications were found.
- The development environment is configured.

Test coverage

Test coverage of the project is **74.4%** (branch coverage of files in the third scope).

- *BasicSpell.sol* contract has extremely low test coverage.

Security score

As a result of the audit, the code contains **1** high, **3** medium issues. The security score is **2** 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: **4.33**.

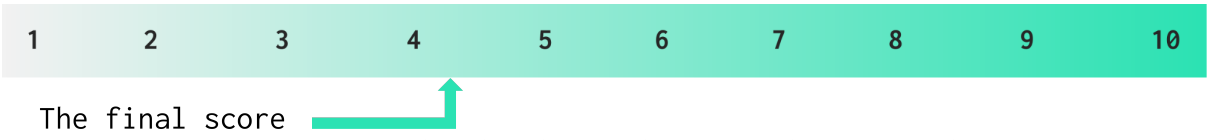

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
21 October 2022	9	4	5	0
28 November 2022	11	3	5	3
08 December 2022	6	1	3	0
16 December 2022	0	0	0	0
17 January 2023	0	3	1	0

Checked Items

We have audited the Customers' smart contracts for commonly known and more 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.	Failed
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 EIP-155	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	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.	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.	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 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Failed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

System Overview

Blueberry is a leveraged yield-farming product. There are the contracts:

- *BlueBerryBank* — main system contract, supports borrowing, lending and repaying debts.
- *BBMath utils* — utils for operating numbers with floating points.
- *WERC20* — ERC1155 wrapper of ERC20 tokens.
- *WIchiFarm* — wrapper of ICHI liquidity management protocol.
- *HardVault* - ERC1155 contract which represents the deposits as an ERC1155 collection.
- SoftVault - ERC20 token contract which represents the amount of tokens lended to the Compound protocol.
- *Spell contracts* — strategy contracts for interacting with *BlueBerryBank.*
- *Oracles - contracts which get the token prices from different oracles.*

Privileged roles

BlueBerryBank:

- Admin (owner) - may set arbitrary oracle implementation, update bank status, whitelist users and tokens.
- Whitelisted contract - may interact with the bank when the contract calls are not allowed.

Oracle:

- Admin - may update the liquidation threshold and whitelist ERC1155 collections.

Risks

- The Admin of the system may update the contracts implementation.
- The debt value increases when new deposits happen, unexpected position liquidation may happen if debt value increases a lot.
- The smart contract system highly relies on compound token implementation, which is out of scope.
- The *WIchiFarm* implementation highly depends on the *ichiFarm,* which is out of audit scope.
- The repository contains contracts that are out of the audit scope and are not considered safe. The audit does not guarantee that, if used, those contracts would not cause additional security issues. Example: *UniswapV2Oracle* contract that is vulnerable to the flashloan attack.

Findings

■■■■ Critical

1. Data Consistency

Users may unexpectedly manipulate their position risk values.

As position *underlyingToken* value is reassigned and *underlyingAmount* is increased during the lend process, users may lend a lot of invaluable tokens at first and then deposit some valuable tokens making the lent amount unexpectedly valuable.

This may lead to the inability to liquidate a position which is at risk and, therefore, funds loss.

Paths:
./contracts/spell/IchiVaultSpell.sol : increasePosition()
./contracts/BlueBerryBank.sol : lend()

Recommendation: implement the lend process safely, use bit-map functionality for deposits, or allow only one lend token per position.

Status: Fixed (fifth scope)

2. Data Consistency

Users may not be able to repay their debts and withdraw the collateral.

The contract borrows some tokens itself without adding collateral calling it “taking fee”.
Due to the high volatility of digital assets, some position collateral may become under debt value in a moment.

Debt liquidation requires debt repayment, so if position collateral is under debt value, it may never be liquidated.

This may lead to inconsistent system situations when unliquidatable debts are growing, and active users are not able to repay their debts to withdraw collateral.

Path: ./contracts/BlueBerryBank.sol : accrue()

Recommendation: rework the logic, positions where collateral is under debt value should not influence active users.

Status: Fixed (fifth scope)

3. Flashloan Attack

In case of calls from smart contracts are not restricted, flashloaner may make positions liquidatable and receive profit liquidating them.

The contract borrows some tokens itself without adding collateral calling it “taking fee”.

In such a way, flashloaner may increase debt and pay huge fee. The fee is borrowed, so total contract debt increases and some positions become under threshold. Flashloaner liquidates the positions receiving profit and then repays borrow and returns loaned assets.

This may lead to unexpected positions liquidation.

Path: ./contracts/BlueBerryBank.sol : accrue()

Recommendation: rework the logic, implement debt fee as percent value per time period.

Status: Fixed (fifth scope)

■■■ High

1. Highly Permissive Role Access

uTokens may be deposited to the *SafeBox* contract by the user or bank in exchange for a collateral token. The admin of the *SafeBox* contract may withdraw any amount of *uToken*.

For users, this may lead to the inability to withdraw the deposited tokens.

Path: ./contracts/SafeBox.sol : claim(), adminClaim()

Recommendation: add variable to account for the amount of deposited tokens and add logic to disallow withdrawing of *uTokens* deposited by users.

Status: Fixed (second scope)

2. Requirements Violation

According to the described requirements in the “Liquidation Logic” documentation, the liquidation threshold is fixed for different types of collateral tokens. The implementation of the *ProxyOracle* contract allows the admin to change the possible *borrowFactor* and liquidation risk threshold.

This may lead to unexpected risk for users and loss of funds in case of significant token price fluctuations.

Path: ./contracts/oracle/ProxyOracle.sol : setTokenFactors(), unsetTokenFactors()

Recommendation: add logic to validate liquidation risk threshold according to the documentation.

Status: Fixed (second scope)

3. Requirements Violation

According to the requirements, liquidation is the process when somebody pays debt for its owner and takes the collateral, but according to the current implementation of the *liquidate* function, the collateral is not sent to the liquidator.

This leads to an inability to make a profit for the position liquidator, so the position may not be liquidated and the system loses the assets' valuability.

Path: ./contracts/oracle/BlueBerryBank.sol : liquidate()

Recommendation: rework the logic to repay the collateral tokens to the liquidator account.

Status: Fixed (second scope)

4. **Requirements Violation**

According to the described requirements in the "Liquidation Logic" documentation, the liquidation threshold is fixed for different types of collateral tokens. The implementation of the *CoreOracle* contract allows the admin to change the possible liquidation risk threshold.

This may lead to unexpected risk for users and loss of funds in case of significant token price fluctuations.

Path: ./contracts/oracle/CoreOracle.sol : setTokenSettings(), removeTokenSettings()

Recommendation: add logic to validate the liquidation risk threshold according to the documentation.

Status: Fixed (fifth scope)

5. **Requirements Violation**

Force pragma updates of well-known contracts and libraries may lead to unexpected behavior.

The *FullMath*, *OracleLibrary*, *TickMath* libraries are designed for *pragma solidity <0.8.0*. After the force pragma update, some functions fail for specific inputs.

Paths:
./contracts/libraries/UniV3/FullMath.sol
./contracts/libraries/UniV3/TickMath.sol
./contracts/libraries/UniV3/OracleLibrary.sol

Recommendation: configure the project to be compiled with the old compiler version for some files, import the libraries directly from the source.

Status: Fixed (fourth scope)

6. **Data Consistency**

The number is casting from type *intX* to *intY* where *X > Y* may lead to silent overflow errors.

This may lead to unexpected contract behavior.

Paths:
./contracts/libraries/UniV3/OracleLibrary.sol : consult(),

getBlockStartingTickAndLiquidity(), getWeightedArithmeticMeanTick()
./contracts/libraries/UniV3/PoolVariables.sol : getTwap()

Recommendation: check number is within bounds of the new type before casting.

Status: Fixed (fourth scope)

7. **Denial of Service**

According to the described requirements and code comments, each *Bank* entity has the SafeBox, the SafeBox sets the *_uToken* automatically by getting the underlying token from the *cToken*. During the Bank creation, the admin sets underlying token and cToken, but the values are not validated, so it is possible to set *Bank* tokens which do not correspond to the *SafeBox* tokens.

This may lead to the denial of service situation and unexpected contract behavior (including funds lock, unexpected token transfers, etc.).

Path: ./contracts/BlueBerryBank.sol : addBank()

Recommendation: add logic to validate the token and its underlying token during the bank entity creation.

Status: Fixed (fifth scope)

8. **Data Consistency**

It is possible for the owner to change the *Bank* address at the *SafeBox* contract and vice versa. *SafeBox* is an extended *ERC-20* token, with information about the underlying asset.

This may lead to denial of service situations and funds lock.

Path: ./contracts/BlueBerryBank.sol : addBank(), updateSafeBox()

Recommendation: rework the logic, do not update *SafeBox* address until there are active users.

Status: Fixed (fifth scope)

9. **Data Consistency**

It is possible for the owner to change the *Bank* address at the *SafeBox* contract and vice versa. *SafeBox* is an extended *ERC-20* token, with information about the underlying asset.

This may lead to denial of service situations and funds lock.

Path: ./contracts/SafeBox.sol : setBank()

Recommendation: rework the logic, do not update *Bank* address until there are active users.

Status: Fixed (third scope)

10. Denial of Service Vulnerability

In case when the *allBanks* array is long enough the loop in the function will never be executed due to costly calls to the oracles contracts.

This may lead to failures due to the block Gas limit.

Path: ./contracts/BlueBerryBank.sol : setOracle()

Recommendation: add ability to paginate over the array values or limit the array length.

Status: New

Medium

1. Division Before Multiplication

The contract has the code which performs division before multiplication during the calculation.

This may lead to a loss of precision.

Path: ./contracts/oracle/ProxyOracle.sol : getCollateralValue(), getDebtValue()

Recommendation: perform multiplication before division.

Status: Fixed (second scope)

2. Redundant Code

The contract has the code which transfers zero collection tokens and approves a specific amount of tokens right after setting zero allowance.

This leads to redundant Gas usage.

Path: ./contracts/BlueBerryBank.sol : liquidate(), addBank()

Recommendation: remove redundant zero transfer.

Status: Fixed (second scope)

3. Missing Parameter Validation

According to the *BlueBerryBank* implementation, the *feeBps* value should be validated, but the validations are missed in the constructor. However, the validations are done in the setter *setFeeBps*.

Unexpectedly high fee value set in constructor may lead to funds loss.

Path: ./contracts/BlueBerryBank.sol : constructor()

Recommendation: validate parameters consciously.

Status: Fixed (second scope)

4. **Oracle Maximum Delay Time**

   The oracle maximum delay time parameters should be validated in order to prevent price manipulation by the system owner.

   Paths:
   ./contracts/oracle/BandAdapterOracle.sol : setMaxDelayTimes()
   ./contracts/oracle/ChainlinkAdapterOracle.sol : setMaxDelayTimes()

   Recommendation: provide the maximum possible delay time, e.g. *2 days*.

   Status: Fixed (second scope)

5. **Best Practice Violation**

   It is considered operating user assets in one place, separating it from other logic.

   It is recommended to restrict *spell contracts* operating user funds and move the functionality to the *bank contract*.

   Path: ./contracts/spell/*

   Recommendation: provide abstract layers consciously, implement fee taking at one contract to control the assets flow safely.

   Status: Fixed (fifth scope)

6. **Inconsistent Data**

   Due to oracle updates, some functionality may be broken as oracle may not support some bank tokens.

   This may lead to unexpected partial DoS situations.

   Path: ./contracts/BlueBerryBank.sol : setOracle()

   Recommendation: check if the new oracle supports bank tokens.

   Status: Fixed (fifth scope)

7. **Unscalable Functionality**

   The contracts system has the constant precision value equal to 10000; it is widely used across contracts.

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

   Paths:
   ./contracts/BlueBerryBank.sol : initialize(), setFeeBps(), accrue(), getPositionRisk(), doCutDepositFee(), doCutWithdrawFee()
   ./contracts/spell/BasicSpell.sol : doCutRewardsFee()

   Recommendation: it is recommended to avoid "magic numbers" during the development and replace it with the constant value.

   Status: Fixed (fifth scope)

8. Unscalable Functionality

The contracts system has the constant precision value equal to 10000; it is widely used across contracts.

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

Paths:
./contracts/ProtocolConfig.sol : setFeeDistribution()
./contracts/oracle/CoreOracle.sol : setTokenSettings()

Recommendation: it is recommended to avoid "magic numbers" during the development and replace it with the constant value.

Status: Fixed (fourth scope)

9. Contradiction

It is considered project should be consistent and contain no self contradictions. The name of the function and event contradict actual code behavior.

According to the implementation the function may change the `maxLTV` param for existing collaterals, however, according to the name it should only be possible to add data about new collateral tokens.

This may lead to wrong assumptions about the code purpose.

Path: ./contracts/spell/IchiVaultSpell.sol : addCollaterals(), CollateralsAdded

Recommendation: rename the function and event according to its purpose or update the logic.

Status: New

10. Unscalable Functionality

The contracts system has the constant precision value equal to `8000`; it is widely used across contracts.

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

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

Recommendation: it is recommended to avoid "magic numbers" during the development and replace it with the constant value.

Status: New

11. Non-Finalized Code

The Production Code has Functions for Tests.

This leads to the increase of Gas expenses during the contract deployment, the code violates the functional requirements if the contract is not deployed to the Ethereum Mainnet network.

Path: ./contracts/BlueBerryBank.sol : updateVault(), updateCToken()

Recommendation: remove functions which should not be present in production.

Status: New

Low

1. Floating Pragma

Contracts should be deployed with the same compiler version and flags that have been tested thoroughly. Locking the Pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Paths: ./contracts/*

Recommendation: use a fixed version of the compiler (^ symbol should be removed from Pragma).

Status: Fixed (second scope)

2. Missing Zero Address Validation

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

Paths:
./contracts/Governable.sol : setPendingGovernor()
./contracts/SafeBox.sol : setBank()

Recommendation: add zero address validation.

Status: Fixed (second scope)

3. Default Variable Visibility

The lack of variable visibility may cause unexpected variable visibility in derived contracts.

Paths:
./contracts/spell/IchiVaultSpell.sol : swapPool
./contracts/Governable.sol : _gap

Recommendation: specify the needed visibility during the variable initialization.

Status: Fixed (second scope)

4. Redundant Function Arguments

The arguments of the function should match the specific state variables; otherwise, the transaction will be reverted.

This may lead to the redundant Gas usage because of the redundant required statements and variable memory allocation.

Path: ./contracts/BlueBerryBank.sol : takeCollateral()

Recommendation: remove the arguments *collToken* and *collId*.

Status: Fixed (second scope)

5. **Code Duplication**

   Modifiers should be used instead of code duplicates the modifier logic.

   This may lead to modifying code in several places instead of one when the logic is updated.

   Path: ./contracts/BlueBerryBank.sol : borrowBalanceCurrent()

   Recommendation: replace the call with the *poke* modifier.

   Status: Fixed (second scope)

6. **Checks-Effects-Interactions Pattern Violation**

   During functions, the state variables are being updated after the external calls, or checks are done after state variables are updated.

   This can lead to reentrancies, race conditions, or denial of service vulnerabilities.

   Paths:
   ./contracts/oracle/AggregatorOracle.sol : initialize()
   ./contracts/BlueBerryBank.sol : initialize()

   Recommendation: implement function according to the Checks-Effects-Interactions pattern.

   Status: Fixed (second scope)

7. **Functions that Could Be Declared as External**

   public functions that are never called by the contract should be declared *external* to save Gas.

   Path:./contracts/utils/ERC1155NaiveReceiver.sol : supportsInterface()

   Recommendation: use the *external* attribute for functions never called from the contract.

   Status: Fixed (fifth scope)

8. **Functions that Could Be Declared as External**

   public functions that are never called by the contract should be declared *external* to save Gas.

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

   Recommendation: use the *external* attribute for functions never called from the contract.

Status: Fixed (fourth scope)

9. **Default Variable Visibility**

   The lack of variable visibility may cause unexpected variable visibility in derived contracts.

   Paths: ./contracts/spell/IchiVaultSpell.sol : vaults

   Recommendation: specify the needed visibility during the variable initialization.

   Status: Fixed (fifth scope)

10. **Redundant Library**

    Since Solidity v0.8.0, the overflow/underflow check is implemented on the language level - it adds the validation to the bytecode during compilation.

    There is no need to use the SafeMath library.

    Path: ./contracts/libraries/UniV3/LowGasSafeMath.sol

    Recommendation: remove the SafeMath library.

    Status: Fixed (fourth scope)

11. **Checks-Effects-Interactions Pattern Violation**

    During functions, the state variables are being updated after the external calls, or checks are done after state variables are updated.

    This can lead to reentrancies, race conditions, or denial of service vulnerabilities.

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

    Recommendation: implement function according to the Checks-Effects-Interactions pattern.

    Status: Reported

12. **Missing Return Value Validation**

    Return value validation is necessary to handle errors and unexpected results during the calls to different functions.

    Paths:
    ./contracts/BlueBerryBank.sol : withdrawLend(), lend()
    ./contracts/spell/BasicSpell.sol : constructor()
    ./contracts/spell/IchiVaultSpell.sol : constructor()

    Recommendation: validate return value.

    Status: Fixed (fifth scope)

13. **Missing Return Value Validation**

    Return value validation is necessary to handle errors and unexpected results during the calls to different functions.

    Path: ./contracts/SafeBox.sol : constructor()

    Recommendation: validate return value.

    Status: Fixed (third scope)

14. **Floating Pragma**

    Contracts should be deployed with the same compiler version and flags that have been tested thoroughly. Locking the Pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

    Paths:
    ./contracts/interfaces/uniswap/v3/IUniswapV3Pool.sol
    ./contracts/interfaces/uniswap/v3/pool/*
    ./contracts/libraries/UniV3/FixedPoint96.sol
    ./contracts/libraries/UniV3/LiquidityAmounts.sol
    ./contracts/libraries/UniV3/PoolVariables.sol
    ./contracts/libraries/UniV3/PositionKey.sol
    ./contracts/libraries/UniV3/SqrtPriceMath.sol
    ./contracts/libraries/UniV3/UnsafeMath.sol

    Recommendation: use a fixed version of the compiler (^ symbol should be removed from Pragma).

    Status: Fixed (fifth scope)

15. **Floating Pragma**

    Contracts should be deployed with the same compiler version and flags that have been tested thoroughly. Locking the Pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

    Path: ./contracts/ProtocolConfig.sol

    Recommendation: use a fixed version of the compiler (^ symbol should be removed from Pragma).

    Status: Fixed (third scope)

16. **Floating Pragma**

    Contracts should be deployed with the same compiler version and flags that have been tested thoroughly. Locking the Pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Path: ./contracts/interfaces/IProtocolConfig.sol

Recommendation: use a fixed version of the compiler (^ symbol should be removed from Pragma).

Status: Fixed (fourth scope)

17. **Code Duplication**

The project has the oracles with the full code duplication of the function.

This may lead to difficulties during the development process.

Paths:
./contracts/oracle/BaseAdapterOracle.sol : setMaxDelayTimes()
./contracts/oracle/ChainlinkAdapterOracle.sol : setMaxDelayTimes()

Recommendation: consider moving the *setMaxDelayTimes* function to a separate abstract contract.

Status: Fixed (fourth scope)

18. **Code Duplication**

The project has the require statement *require(whitelistedTokens[token], 'token not whitelisted');* which is often used and may be replaced with the proper modifier.

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

Recommendation: consider moving the required statement to the modifier.

Status: Fixed (fifth scope)

19. **Missing Checks**

It is possible to create a *Bank* entity with a non-whitelisted token.

This may lead to inability to execute other *Bank* related functions until the token is not whitelisted.

Path: ./contracts/BlueBerryBank.sol : addBank()

Recommendation: validate if the token is whitelisted or not during the *Bank* creation.

Status: Fixed (fifth scope)

20. **Inconsistent Environment**

During the typical tests execution workflow, one of the tests is failing, which does not allow to measure the coverage properly.

Recommendation: update the tests to avoid failing.

Status: Fixed (fourth scope)

21. Missing Zero Address Validation

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

Path: ./contracts/BlueBerryBank.sol : addBank()

Recommendation: add zero address validation.

Status: Fixed (fifth scope)

22. Default Variable Visibility

The lack of variable visibility may cause unexpected variable visibility in derived contracts.

Path: ./contracts/SafeBox.sol : withdrawFeeWindowStartTime

Recommendation: specify the needed visibility during the variable initialization.

Status: Fixed (fourth scope)

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 report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted to 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, Consultant cannot guarantee the explicit security of the audited smart contracts.