HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Venus
Date: April 26, 2023

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

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

Document

Name	Smart Contract Code Review and Security Analysis Report for Venus
Approved By	Yevhenii Bezuhlyi | SC Audits Department Head at Hacken OU
Type	Oracle
Platform	BSC
Language	Solidity
Methodology	Link
Website	
Changelog	23.12.2022 - Initial Review 19.01.2023 - Second Review 10.04.2023 - Third Review 26.04.2023 - Fourth Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Venus (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/VenusProtocol/oracle
Commit	1411c8fdf1f716ab3d610bb188d8e0a2bae05592
Whitepaper	https://venus-protocol.gitbook.io/v4/1ylMKfqax6oBEakCBM3M/getting-started/contracts-overview#oracle-contracts
Functional Requirements	https://venus-protocol.gitbook.io/v4/1ylMKfqax6oBEakCBM3M/oracles/resilient-oracle
Technical Requirements	
Contracts	File: ./contracts/interfaces/AggregatorV2V3Interface.sol SHA3: 1438fc10aa872a939deb46a3b09012d38328c02e9ac4a71a1d7b5ee2468fe97e File: ./contracts/interfaces/BEP20Interface.sol SHA3: 7286bb5b64dacb1f5e44eb35ed3ffe13a1fdbb7f7e6108afac018129110bf4a8 File: ./contracts/interfaces/FeedRegistryInterface.sol SHA3: 4b56470b0fe1ba1770b29d045a2fdbc297a2fcd9eddd4f0f2a73e80185206b23 File: ./contracts/interfaces/OracleInterface.sol SHA3: ce5367b470cf785812c981d49725ad34ea05881800a97f33b95aec4347edec08 File: ./contracts/interfaces/PythInterface.sol SHA3: edc6e9a0bcc7789d3ffed7ea27e21132434c3958c4d6311194f93a698ab2c302 File: ./contracts/interfaces/VBep20Interface.sol SHA3: 224b1e11f30e616b5fb68eb6d223ad48f277af5a83fd4880ceb9ea716b590c17 File: ./contracts/libraries/PancakeLibrary.sol SHA3: 98164da996f54f35a008075b54353bc4e6f0c593f045b0131765960c43d73936 File: ./contracts/oracles/BinanceOracle.sol SHA3: 5d1bdbf1b4ef8f56006d6a35ac84e0eea3b9d6bf02f1228c78c0cbd75f452b86 File: ./contracts/oracles/BoundValidator.sol SHA3: ae46c288b4a32d00d446af44eb970b518ed38c48f552928ab00177781950f5d7

	File: ./contracts/oracles/ChainlinkOracle.sol SHA3: cc46a5503413a25764b0f6d5797bf887d0c9f26c05d273958edf76384a207a14 File: ./contracts/oracles/PythOracle.sol SHA3: feb755c5a5dff410a3d7998428e8dc8e3768221370433d3337b18a632ba33bc2 File: ./contracts/oracles/TwapOracle.sol SHA3: 144ad637555f1b9e1f802de0498e98045a3738b719fe59a15002b517298982af File: ./contracts/PriceOracle.sol SHA3: 26c2a45e38eb165e0b258626db926fe7154e544f4c566a782acf8fccccb94511 File: ./contracts/ResilientOracle.sol SHA3: 9e08ef47e26ef23a4748127a5b2180c535dd2dd1e07bcf380fc6fe0d8441d3bd

Second review scope

Repository	https://github.com/VenusProtocol/oracle
Commit	355c5911ab97d9709429880d3a137bd43d96e956
Whitepaper	https://venus-protocol.gitbook.io/v4/1ylMKfqax6oBEakCBM3M/getting-started/contracts-overview#oracle-contracts
Functional Requirements	
Technical Requirements	
Contracts	File: ./contracts/interfaces/FeedRegistryInterface.sol SHA3: a929a4dc565a471273b7bdec367cca4a1542eefe8233e7d157bc2adc864efe8d File: ./contracts/interfaces/OracleInterface.sol SHA3: 0eed1177a86665fda8682ff23c073682b692dacd0c226fe83e702179cf10de24 File: ./contracts/interfaces/PythInterface.sol SHA3: 8bf190cca92d378e5a14f21b0b9f48f5f2f0dd6140c57f29108017ea7d2eb9c8 File: ./contracts/interfaces/VBep20Interface.sol SHA3: a9032d1c36c657f81d1a543f770090cee90fadbc75d4dca91c19d6b91c0c6d11 File: ./contracts/libraries/PancakeLibrary.sol SHA3: 1cb6fc669883c50e662a69b8fec293e8d286807563be5a17334ffb1e5bec6028 File: ./contracts/oracles/BinanceOracle.sol SHA3: 6a9959ac5886e61118617ca7b2155a393c656a142b70d24c37bd1ad43a8d5ebc File: ./contracts/oracles/BoundValidator.sol SHA3: df8e7a587b917a97bd1e9fe6acab4c3c3ca971251392a74321a6ab107b7c5d21

	File: ./contracts/oracles/ChainlinkOracle.sol SHA3: 881e8272ad33a99c2e3419be8f5777461d6d06576c2f16210cb9ea7a1fcbc194 File: ./contracts/oracles/PythOracle.sol SHA3: 3d2a64ddf34015747bf9790b283a71cc749ca602fbf0cf6e98fd98a9bb873221 File: ./contracts/oracles/TwapOracle.sol SHA3: 019a8d6442e2b3e038d4b9cee56a6c46a83302dedce6cb0703fc485a4f39f516 File: ./contracts/PriceOracle.sol SHA3: c1c390a130ff7df2173055f709d1643610fc5a69d61274dc6d595495df50c507 File: ./contracts/ResilientOracle.sol SHA3: 4eb943b07bf5cd26cc8d00ceacbc189613ae6ccbdc1793ce7ca6893834948ff9

Third review scope

Repository	https://github.com/VenusProtocol/oracle
Commit	c916c26117099e9f7dc0bc8333e4e0162ce1b7c2
Whitepaper	https://venus-protocol.gitbook.io/v4/1ylMKfqax6oBEakCBM3M/getting-started/contracts-overview#oracle-contracts
Functional Requirements	
Technical Requirements	
Contracts	File: ./contracts/interfaces/FeedRegistryInterface.sol SHA3: cad4841a41bb5d2016f025e0b9be401e980d7e7dd6a564a9c829ac092aab2574 File: ./contracts/interfaces/OracleInterface.sol SHA3: 2cdabe0f3287911fde6837d78568780a5c3619d9a0ce6e654e3c935af4e79915 File: ./contracts/interfaces/PublicResolverInterface.sol SHA3: 6a5fc13054cd05b787b161993f62275e1661fed2476fa4240bef7a515b3eaa0a File: ./contracts/interfaces/PythInterface.sol SHA3: d1789f5c3ab73b70077bf36c498e19efc18de19386ecdf94afc3adf283dfb1ab File: ./contracts/interfaces/SIDRegistryInterface.sol SHA3: 8e900f5ff77d6d6e015751408b3a365dc8850046d1a8efcd707aca8300cb16d4 File: ./contracts/interfaces/VBep20Interface.sol SHA3: 8e33f4d371da4e2ae4a52537fd73e26d70c10c41e1298399b386daf32fa02546 File: ./contracts/libraries/PancakeLibrary.sol SHA3: cd85bbbfb29f528174da8ab5b53129c89c9faa37f47e8ccca826273d6cda1389

	File: ./contracts/oracles/BinanceOracle.sol SHA3: b39c84aade69f7fb9e5339e2c254ba7064433de080f7bfb275dbc3776cd54a9f File: ./contracts/oracles/BoundValidator.sol SHA3: 4992138e13fc79023c290d24ce00694f83f3239deb08db100a3c9aecbd9301c1 File: ./contracts/oracles/ChainlinkOracle.sol SHA3: c61d47815a6058b51522d6419782fe31a94c0126d27a56e67f4e81b215dce7cf File: ./contracts/oracles/PythOracle.sol SHA3: 4bc57fe31b4c408c22e55163937f58b5bced576a29a7ba187bc82af53a32eaab File: ./contracts/oracles/TwapOracle.sol SHA3: bf282a3568e6ec8332bbc0221c0f3b4bf2a14aa0c241be943514a4c1cea4fc4b File: ./contracts/PriceOracle.sol SHA3: c3e3f501d4cd40aa7bac7c2ac4eea0f0b1bd2567a87b5ab7474877a46959ccab File: ./contracts/ResilientOracle.sol SHA3: 05c4b55feeb0c3b98b2a65dac28118eb464bbf4529d6f05f3782cdacdc5952ea

Fourth review scope

Repository	https://github.com/VenusProtocol/oracle
Commit	62ff8e2521ae7fa75431ec4ea71440a7694762ed
Whitepaper	https://venus-protocol.gitbook.io/v4/1ylMKfqax6oBEakCBM3M/getting-started/contracts-overview#oracle-contracts
Functional Requirements	
Technical Requirements	
Contracts	File: ./contracts/interfaces/FeedRegistryInterface.sol SHA3: cad4841a41bb5d2016f025e0b9be401e980d7e7dd6a564a9c829ac092aab2574 File: ./contracts/interfaces/OracleInterface.sol SHA3: 2cdabe0f3287911fde6837d78568780a5c3619d9a0ce6e654e3c935af4e79915 File: ./contracts/interfaces/PublicResolverInterface.sol SHA3: 6a5fc13054cd05b787b161993f62275e1661fed2476fa4240bef7a515b3eaa0a File: ./contracts/interfaces/PythInterface.sol SHA3: d1789f5c3ab73b70077bf36c498e19efc18de19386ecdf94afc3adf283dfb1ab File: ./contracts/interfaces/SIDRegistryInterface.sol SHA3: 8e900f5ff77d6d6e015751408b3a365dc8850046d1a8efcd707aca8300cb16d4

	File: ./contracts/interfaces/VBep20Interface.sol SHA3: 8e33f4d371da4e2ae4a52537fd73e26d70c10c41e1298399b386daf32fa02546 File: ./contracts/libraries/PancakeLibrary.sol SHA3: cd85bbbfb29f528174da8ab5b53129c89c9faa37f47e8ccca826273d6cda1389 File: ./contracts/oracles/BinanceOracle.sol SHA3: eacf7f437553380e8d8681179ad97e7850c1e4862bc9fa7bf5c24734ba47f69a File: ./contracts/oracles/BoundValidator.sol SHA3: e4ab515f8e83008eccec8cb5b7d31837f5746c707e2366c778640875bda2c99e File: ./contracts/oracles/ChainlinkOracle.sol SHA3: 8819d69ff8aa6b31299f593a754f384dc8693e432248a9ffccd75e3792ebdc8d File: ./contracts/oracles/PythOracle.sol SHA3: 54ff14fbe54f28c344bbc9306db16b93bce5dae4fd2eb0d3a363847475122af7 File: ./contracts/oracles/TwapOracle.sol SHA3: d9497e3ae44a1d584ad8b3e02432acfe570cfff7f18d0951582219df8249b4fc File: ./contracts/ResilientOracle.sol SHA3: c91feb9b9f0b7a7a4d6bfcdbbd8e83c32f49ea7596a2d7858fe4e6319715ea51

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 assets 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 the code quality

Executive Summary

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

Documentation quality

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

- Project description with technical details is provided.
- Code is covered with the NatSpec comments.

Code quality

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

Test coverage

Code coverage of the project is **78.82%** (branch coverage).

Security score

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

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

Summary

According to the assessment, the Customer's smart contract has the following score: **9.21**.

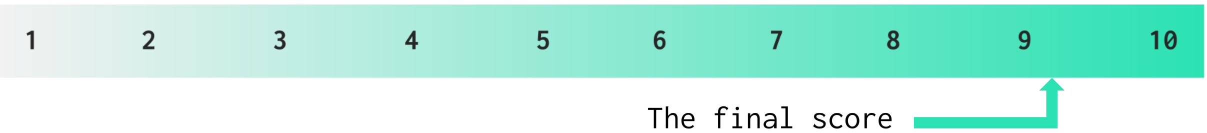

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
23 December 2022	7	2	1	0
17 January 2023	0	0	0	0
10 April 2023	4	4	0	0
26 April 2023	1	0	0	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.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Passed
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	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.	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes 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.	Not Relevant
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Not Relevant
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant

Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer.	Not Relevant
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. 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

Venus is an oracle system with following contracts:

- *ResilientOracle* - oracle aggregator which includes functionality for setting, updating oracle configurations for various tokens, pausing and unpausing the contract, and retrieving prices from different sets of oracles for tokens.
- *BinanceOracle* - the contract fetches prices of assets from Binance oracle.
- *BoundValidator* - the contact is used to validate prices from two different sources, according to the upper and lower bound ratios config for each vToken in this contract.
- *ChainlinkOracle* - contract which fetches prices of assets from Chain Link oracle.
- *PythOracle* - contract which fetches prices of assets from Pyth oracle.
- *TwapOracle* - contract which fetches prices of assets from PancakeSwap oracle.

Privileged roles

The *owner* of oracle may specify the contract which controls the list of access roles.

ResilientOracle - access to the functions is controlled by a set of custom roles. Functionality which is controlled by the roles:

- pausing/unpausing of the contract;
- list of oracles for different tokens;
- enabling/disabling oracles.

BinanceOracle - no privileged roles.

BoundValidator - access to the functions is controlled by a set of custom roles. Functionality which is controlled by the roles:

- configuration of price boundaries.

ChainlinkOracle - access to the functions is controlled by a set of custom roles. Functionality which is controlled by the roles:

- setting tokens configurations;
- setting price feed addresses for different tokens.

PythOracle - access to the functions is controlled by a set of custom roles. Functionality which is controlled by the roles:

- setting tokens configurations;
- setting price feed addresses for different tokens.

TwapOracle - access to the functions is controlled by a set of custom roles:

- setting tokens configurations;
- setting price feed addresses for different tokens.

Risks

- The oracle system highly relies on third party oracles; before using the system, it is necessary to make sure that all the oracle addresses are set up correctly.
- The contracts in the system are upgradable, the logic may be updated by the owner.
- The address of the contract, which controls the list of access roles, may be changed by the owner or set up incorrectly.
- The Resilient Oracle aggregator contract may be paused.
- The module responsible for the managing of access roles is out of the audit scope.

Findings

🟥🟥🟥🟥 Critical

No critical severity issues were found.

🟧🟧🟧 High

1. Non-Finalized Code

The production code should not contain any functions or variables used solely in the test environment or TODO comments. It means that the code is not finalized, and additional changes will be introduced in the future. Malicious actors will be able to manipulate the users to trigger them not to interact with the unfinalized contracts.

This can lead to a loss of funds.

Path: ./contracts/oracles/ChainlinkOracle.sol : lines 23, 64

Recommendation: finalize code and remove TODO comments.

Status: Fixed (revised commit: 355c591)

🟨🟨 Medium

1. Missing Events Emitting

Contracts do not emit events after changing important values.

Events for critical state changes should be emitted for tracking things off-chain.

Path: ./contracts/ResilientOracle.sol

Recommendation: emit or remove *GlobalEnable* event.

Status: Fixed (revised commit: 355c591)

2. Inefficient Gas Model

Contracts use loops without optimization.

This will lead to higher Gas expenses.

Paths:
./contracts/ResilientOracle.sol: setTokenConfigs();
./contracts/oracles/BoundValidator.sol: setValidateConfigs();
./contracts/oracles/ChainlinkOracle.sol: setTokenConfigs();
./contracts/oracles/PythOracle.sol: setTokenConfigs();
./contracts/oracles/PythOracle.sol: TwapOracle();

Recommendation: cache arrays in a loop.

Status: Fixed (revised commit: 355c591)

3. Inefficient Gas Model - Redundant State Constant

The project has a contract with the unused public state constant.

This leads to higher Gas expenses during the contract deployment.

Path: ./contracts/oracles/TwapOracle.sol : expScale

Recommendation: remove unused state constant.

Status: Fixed (revised commit: 62ff8e2)

4. Inefficient Gas Model - Redundant Library

Starting with Solidity *^0.8.0*, SafeMath functions are built-in. In such a way, the library is redundant.

This may lead to the higher Gas expenses during the contract deployment and interactions with the functions.

Path: ./contracts/oracles/PythOracle.sol

Recommendation: rework the contract to remove SafeMath library from the code.

Status: Fixed (revised commit: 62ff8e2)

5. Contradiction - Missing Validation

The project depends on various oracles for price data, which could sometimes be outdated. To address this concern, each asset is equipped with a *maxStalePeriod* parameter, but there are no upper limit boundaries set for this parameter.

This allows for the acceptance of older price data in cases where an extremely higher value is specified.

Paths:
./contracts/oracles/ChainlinkOracle.sol : setTokenConfig()
./contracts/oracles/PythOracle.sol : setTokenConfig()

Recommendation: add validation to restrict the maximum value for the *maxStalePeriod* parameter, possible restriction is 15 minutes, but the value should be chosen according to the potential risks which may be accepted.

Status: Mitigated (Client response: Chainlink and Binance Oracle can keep non updated some feeds until 24 hours if the price doesn’t change too much. So, we should put 24 hours as the upper limit, but we don’t know if new tokens will have a longer heartbeat value in the future, so we prefer to keep it without an upper limit.)

■ Low

1. Floating Pragma

The contracts use floating pragma *>=0.8.0*.

Contracts should be deployed with the same compiler version and flags that they 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/*

Recommendation: consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Status: Fixed (revised commit: 355c591)

2. **Outdated Solidity Version**

Using an outdated compiler version can be problematic, especially if publicly disclosed bugs and issues affect the current compiler version. The project may use compiler version 0.6.0.

Path: ./contracts/PriceOracle.sol

Recommendation: use a contemporary compiler version.

Status: Fixed (revised commit: 355c591)

3. **Duplication of Well-Known Contracts**

The custom implementation of the commonly used contracts and interfaces may cause issues during the development process.

Paths:
./contracts/interfaces/BEP20Interface.sol
./contracts/interfaces/AggregatorV2V3Interface.sol

Recommendation: use stable imports from commonly used packages (e.g. OpenZeppelin).

Status: Fixed (revised commit: 355c591)

4. **Using AbiCoderV2 in Solidity ^0.8.0**

ABICoder is built-in in Solidity ^0.8.0. The usage of the pragma is redundant.

Paths:
./contracts/ResilientOracle.sol;
./contracts/oracles/TwapOracle.sol;
./contracts/oracles/ChainlinkOracle.sol;
./contracts/oracles/BoundValidator.sol;

Recommendation: remove redundant pragma.

Status: Fixed (revised commit: 355c591)

5. **Missing License Identifier**

SPDX license identifier is not provided in the source file.

Path: ./contracts/PriceOracle.sol;

Recommendation: consider adding a license identifier before final deployment.

Status: Fixed (revised commit: 355c591)

6. Unused Storage Variable

Variable *isPriceOracle* is declared but never used. It can be removed to save Gas.

Path: ./contracts/PriceOracle.sol: isPriceOracle;

Recommendation: remove unused variables.

Status: Fixed (revised commit: 355c591)

7. Unused Try/Catch Parameter

The variable *_price* is declared but never used. It can be removed to save Gas.

Path: ./contracts/ResilientOracle.sol: updatePrice;

Recommendation: remove, use or comment out the variable name.

Status: Fixed (revised commit: 355c591)

8. Redundant Contract

The project has abstract contracts which are never used.

This may lead to potential confusion for developers during the project development.

Paths:
./contracts/PriceOracle.sol : PriceOracle
./contracts/interfaces/PythInterface.sol : AbstractPyth

Recommendation: remove the redundant contracts or rework the project to use the contracts if needed.

Status: Fixed (revised commit: 62ff8e2)

9. Interface Marked As An Abstract Contract

The project has an abstract contract which has no implemented functions.

This may lead to potential confusion for developers during the project development.

Path: ./contracts/PriceOracle.sol : PriceOracle

Recommendation: rework the logic to convert abstract contracts to the interface or remove the contract if it is redundant.

Status: Fixed (revised commit: 62ff8e2)

10. Code Duplication

The project has a contract which imports the same contract twice.

Path: ./contracts/ResilientOracle.sol :
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol

Recommendation: remove the redundant duplication.

Status: Fixed (revised commit: 62ff8e2)

11. Redundant Function

The project has a contract with the unused internal function.

This may impact the code readability and maintainability.

Path: ./contracts/oracles/ChainlinkOracle.sol : _compareStrings()

Recommendation: remove the unused function.

Status: Fixed (revised commit: 62ff8e2)

12. Inefficient Gas Model

The function, with a string argument, is called from different parts of the contract with a static input, which is packed to the storage during the contract deployment.

This leads to the redundant Gas usage during the contract deployment and function calls.

Paths:
./contracts/Governance/AccessControlled.sol : _checkAccessAllowed();
./contracts/ResilientOracle.sol : pause(), unpause(), setTokenConfigs(), setOracle(), enableOracle(), setTokenConfig();
./contracts/oracles/TwapOracle.sol : setTokenConfigs(), setTokenConfig();
./contracts/oracles/PythOracle.sol : setTokenConfigs(), setUnderlyingPythOracle(), setTokenConfig();
./contracts/oracles/ChainlinkOracle.sol : setUnderlyingPrice(), setDirectPrice(), setTokenConfigs(), setTokenConfig();
./contracts/oracles/BoundValidator.sol : setValidateConfigs(), setValidateConfig().

Recommendation: it is recommended to replace the strings with the signatures of the functions with the function selectors using the function interface *IContractName.functionName.selector*. To implement the recommendation, it is needed to rework the *_checkAccessAllowed* function to accept argument which represents the function signature.

Status: Reported (According to the client explanation: the potential fix of the issue would cause the huge changes in the codebase)

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.