



Venus - RiskOracle Integration

Security Assessment

CertiK Assessed on Feb 19th, 2025





CertiK Assessed on Feb 19th, 2025

Venus - RiskOracle Integration

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES

DeFi

ECOSYSTEM

Binance Smart Chain
(BSC)

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 02/19/2025

KEY COMPONENTS

N/A

CODEBASE

<https://github.com/VenusProtocol/venus-protocol><https://github.com/VenusProtocol/governance-contracts>


View All in Codebase Page

COMMITTS

RiskSteward Base: [b9e45ebe13e65b6ee323487b2c11104de9687227](https://github.com/VenusProtocol/venus-protocol/commit/b9e45ebe13e65b6ee323487b2c11104de9687227)RiskSteward Update1: [45648b9dcc0fc74fc3546ea0dfc25d55433235db](https://github.com/VenusProtocol/venus-protocol/commit/45648b9dcc0fc74fc3546ea0dfc25d55433235db)RiskSteward Update2: [02d89861ecddb947bfe1165ba8ddb0485f7c5cd9](https://github.com/VenusProtocol/venus-protocol/commit/02d89861ecddb947bfe1165ba8ddb0485f7c5cd9)

View All in Codebase Page

Highlighted Centralization Risks

 Contract upgradeability

Vulnerability Summary



17

Total Findings

13

Resolved

0

Mitigated

1

Partially Resolved

3

Acknowledged

0


Declined

 0 Critical

Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.

 2 Major 2 Acknowledged

Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.

 2 Medium 2 Resolved

Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

 2 Minor 2 Resolved

Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.

11

Informational

9 Resolved, 1 Partially Resolved, 1 Acknowledged



Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | VENUS - RISKORACLE INTEGRATION

Summary

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

Summary

Dependencies

[Third Party Dependencies](#)

[Out Of Scope Dependencies](#)

[Recommendations](#)

Findings

[RSV-01 : Centralization Related Risks](#)

[RSV-02 : Contract Upgrade Centralization Risk](#)

[RSR-04 : Updates May Be Called Out Of Order Or To Prevent Newer Updates From Being Processed](#)

[SFD-02 : Functions Will Always Return Success Even When They Fail](#)

[RSR-05 : Missing Zero Address Validation](#)

[RSV-03 : Missing Input Validation](#)

[MCR-01 : Unnecessary Cases If Core Comptroller Interface Is Updated To Be Compatible With Isolated Pools Interface](#)

[MCR-03 : Event Not Indexed](#)

[MCR-04 : Usage of Magic Numbers](#)

[MCR-06 : Unnecessary Inheritance](#)

[MFD-01 : Inconsistent Grouping Of Functions](#)

[RSR-06 : Indexed Dynamic Data Type In Event](#)

[RSR-07 : Disabled Function Does Not Emit An Error](#)

[RSV-04 : Typos And Inconsistencies](#)

[RSV-05 : Unnecessary Imports](#)

[SFD-03 : Aliased Function Not Included In Updated Interface](#)

[VPB-01 : Missing Or Incomplete Natspec](#)

Optimizations

RSR-01 : User-Defined Getters

RSR-02 : Repeat Calculation

I Appendix

I Disclaimer

CODEBASE | VENUS - RISKORACLE INTEGRATION

Repository

<https://github.com/VenusProtocol/venus-protocol>

<https://github.com/VenusProtocol/governance-contracts>

Commit

RiskSteward Base: [b9e45ebe13e65b6ee323487b2c11104de9687227](#)

RiskSteward Update1: [45648b9dcc0fc74fc3546ea0dfc25d55433235db](#)

RiskSteward Update2: [02d89861ecddb947bfe1165ba8ddb0485f7c5cd9](#)

RiskSteward Update3: [850443bab6e1af9b1bcdcf7b84975556c14dff7b](#)

Align Comptroller Facets Base: [6b6d90fac4ae83e4c25e70882044d4b1a0fb4aa4](#)

Align Comptroller Facets Update1: [bc49c803b7ad086bd3e1f43f6b2edad150b341c4](#)

Align Comptroller Facets Update2: [aa316c5062597111f58bdf090fa891d1fe923440](#)











Align Comptroller Interface Base: [6b6d90fac4ae83e4c25e70882044d4b1a0fb4aa4](#)

Align Comptroller Interface Update1: [bc49c803b7ad086bd3e1f43f6b2edad150b341c4](#)

AUDIT SCOPE | VENUS - RISKORACLE INTEGRATION

10 files audited ● 2 files with Acknowledged findings ● 2 files with Partially Resolved findings

● 1 file with Resolved findings ● 5 files without findings

ID	Repo	File	SHA256 Checksum
● MCR	VenusProtocol/governance-contracts	 MarketCapsRiskSteward.sol	62edf0467e54413eafccd6eb0bb25eb0e90f1b2634bcd1becfcf45b055eb08d
● RSR	VenusProtocol/governance-contracts	 RiskStewardReceiver.sol	94a53dc8411d7517db9d20229a1b0164818237f61f23b811ec235723d010345b
● PFD	VenusProtocol/venus-protocol	 PolicyFacet.sol	69d74d9899b812b5cad5cd87b76430a04f3b3bf18e7c8a710c3e72308b4cf828
● SFD	VenusProtocol/venus-protocol	 SetterFacet.sol	b335684b043eb8579d3ea53ebdb755cb40b52872b2ded41e3e06a76918e862ca
● MFD	VenusProtocol/venus-protocol	 MarketFacet.sol	07c38d1bd69aa39dd7a04c5911be1e2d14ad70dfc62cdb6adebb633e2a05ac9b
● IRS	VenusProtocol/governance-contracts	 IRiskSteward.sol	0c32cc69e5516f52525acca0931359bf7a8d381d1feacd98773ebcfcd25d020b9
● IRR	VenusProtocol/governance-contracts	 IRiskStewardReceiver.sol	e9626e767bd65fe3c39d387923e203958cc36e976067e1cbfea2b084bc05c533
● IMF	VenusProtocol/venus-protocol	 IMarketFacet.sol	b074cc7882be496a52ea3c18d8e72dfb9b9a122e46fbc3b087695c850d0f02d0
● IPF	VenusProtocol/venus-protocol	 IPolicyFacet.sol	4f7f5285977c50f5a853010b0ea338d90dca931008defa4ffecdd8090fe030a
● ISF	VenusProtocol/venus-protocol	 ISetterFacet.sol	ee930dcd27b3936d95dc7ec9d37d6a031dd206db5be1199192c9fdc664a1f795

APPROACH & METHODS | VENUS - RISKORACLE INTEGRATION

This report has been prepared for Venus to discover issues and vulnerabilities in the source code of the Venus - RiskOracle Integration project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

SUMMARY | VENUS - RISKORACLE INTEGRATION

This audit concerns the changes made in files outlined in the following PRs:

- [PR-115](#)
- [PR-548](#)

Note that any centralization risks present in the existing codebase before these PRs were not considered in this audit and only those added in these PRs are addressed in the audit. We recommend all users carefully review the centralization risks, much of which can be found in our previous audits, which can be found here: <https://skynet.certik.com/projects/venus>.

PR-115

This PR implements the `RiskStewardReceiver` and the `MarketCapsRiskSteward` contracts. The `RiskStewardReceiver` is designed to fetch updates from the `RISK_ORACLE` (intended to be Chaos Labs `RiskOracle`), validate them, and then call the appropriate Risk Steward contract to process them. The `MarketCapsRiskSteward` is the initial Risk Steward contract designed to process supply and borrow cap updates. It does so via the function `processUpdate()`, which is only callable by the `RiskStewardReceiver`, where it checks that the updated borrow or supply cap is within a configurable percentage of the current value, after which it updates the borrow or supply cap.

PR-548

This PR adds view functions and alias functions to the Core Pool Comptroller in order to make it compatible with the interface of the Isolated Pools Comptroller.

DEPENDENCIES | VENUS - RISKORACLE INTEGRATION

Third Party Dependencies

The protocol is serving as the underlying entity to interact with third party protocols. The third parties that the contracts interact with are:

- Third Party Oracles (Chaos Labs [RiskOracle](#))

The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. Moreover, updates to the state of a project contract that are dependent on the read of the state of external third party contracts may make the project vulnerable to read-only reentrancy. In addition, upgrades of third parties can possibly create severe impacts, such as returning invalid prices, returning invalid exchange rates, etc.

Out Of Scope Dependencies

The protocol is serving as the underlying entity to interact with out-of-scope dependencies. The out-of-scope dependencies that the contracts interact with are:

- Core Pool And Isolated Pools Comptrollers And VTokens

The scope of the audit treats out-of-scope dependencies as black boxes and assumes their functional correctness.

Recommendations

We recommend constantly monitoring the third parties involved to mitigate any side effects that may occur when unexpected changes are introduced, as well as vetting any third party contracts used to ensure no external calls can be made before updates to its state. Additionally, we recommend all out-of-scope dependencies are carefully vetted to ensure they function as intended.

FINDINGS | VENUS - RISKORACLE INTEGRATION



17

Total Findings

0

Critical

2

Major

2

Medium

2

Minor

11

Informational

This report has been prepared to discover issues and vulnerabilities for Venus - RiskOracle Integration. Through this audit, we have uncovered 17 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
RSV-01	Centralization Related Risks	Centralization	Major	● Acknowledged
RSV-02	Contract Upgrade Centralization Risk	Centralization	Major	● Acknowledged
RSR-04	Updates May Be Called Out Of Order Or To Prevent Newer Updates From Being Processed	Logical Issue	Medium	● Resolved
SFD-02	Functions Will Always Return Success Even When They Fail	Logical Issue	Medium	● Resolved
RSR-05	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
RSV-03	Missing Input Validation	Logical Issue	Minor	● Resolved
MCR-01	Unnecessary Cases If Core Comptroller Interface Is Updated To Be Compatible With Isolated Pools Interface	Logical Issue	Informational	● Acknowledged
MCR-03	Event Not Indexed	Design Issue	Informational	● Resolved
MCR-04	Usage Of Magic Numbers	Coding Issue	Informational	● Resolved
MCR-06	Unnecessary Inheritance	Coding Style	Informational	● Resolved

ID	Title	Category	Severity	Status
MFD-01	Inconsistent Grouping Of Functions	Logical Issue	Informational	● Resolved
RSR-06	Indexed Dynamic Data Type In Event	Design Issue	Informational	● Resolved
RSR-07	Disabled Function Does Not Emit An Error	Inconsistency	Informational	● Resolved
RSV-04	Typos And Inconsistencies	Inconsistency	Informational	● Resolved
RSV-05	Unnecessary Imports	Coding Style	Informational	● Resolved
SFD-03	Aliased Function Not Included In Updated Interface	Inconsistency	Informational	● Resolved
VPB-01	Missing Or Incomplete Natspec	Inconsistency	Informational	● Partially Resolved

RSV-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	● Major	MarketCapsRiskSteward.sol (RiskSteward Base): 110, 126~128; RiskStewardReceiver.sol (RiskSteward Base): 128, 137, 148, 186	● Acknowledged

Description

Note that any centralization risks present in the existing codebase before the PR's in scope of this audit were not considered. Only those added to the in-scope PRs are addressed. We recommend all users carefully review the centralization risks, much of which can be found in our previous audits, which can be found here: <https://skynet.certik.com/projects/venus>.

MarketCapsRiskSteward

In the contract `MarketCapsRiskSteward`, the `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` can grant addresses the privilege to call the function `setMaxDeltaBps()`.

Any compromise to the `DEFAULT_ADMIN_ROLE` or accounts granted this privilege may allow the hacker to take advantage of this authority and set the max delta bps to a small non-zero value to prevent updates from process or to a large value to allow malicious updates to successfully process.

In the contract `MarketCapsRiskSteward`, the `RISK_STEWARD_RECEIVER` has the privilege to call the function `processUpdate()`.

Any compromise to the `RISK_STEWARD_RECEIVER` may allow the hacker to take advantage of this authority to process malicious supply and borrow cap updates.

RiskStewardReceiver

In the contract `MarketCapsRiskSteward`, the `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` can grant addresses the privilege to call the following functions:

- `pause()`
- `unpause()`
- `setRiskParameterConfig()`
- `toggleConfigActive()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or accounts granted this privilege may allow the hacker to take advantage of this authority and do the following:

- Pause the contract to cause a denial of service.
- Unpause the contract to allow actions to be performed when they are not expected.

- Set the risk parameter configuration for any update type, where they can set the `riskSteward` to any address and `debounce` to any non-zero value. This could be used to bypass important checks and to allow updates to be processed in quick succession.
- Toggle a configuration for an update type. This can cause a denial of service if it is toggled off unexpectedly or allow an unintended configuration to be used if it is toggled on.

In addition, some of the aliased functions in the adjustment to the core comptroller were privileged functions. The entities that have access to the original privileged function also have access to the aliased privileged function. However, as the functions are simply aliases, they hold the same power.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (2/3, 3/5) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
AND

- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
OR
- Remove the risky functionality.

I Alleviation

[Venus, 02/19/2025] : "On BNB chain, we'll use the AccessControlManager (ACM) deployed at 0x4788629abc6cfca10f9f969efdeaa1cf70c23555. In this ACM, only 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 (Normal Timelock) has the DEFAULT_ADMIN_ROLE. And this contract is a Timelock contract used during the Venus Improvement Proposals.

On BNB chain, we'll grant [a] (Normal), [b] (Fast-track) and [c] (Critical) timelocks to execute the following functions:

- MarketCapsRiskSteward.setMaxDeltaBps(uint256)
- RiskStewardReceiver.pause()
- RiskStewardReceiver.unpause()
- RiskStewardReceiver.setRiskParameterConfig(string,address,uint256)
- RiskStewardReceiver.toggleConfigActive(string)

The current config for the three Timelock contracts on BNB chain are:

normal: 24 hours voting + 48 hours delay

fast-track: 24 hours voting + 6 hours delay

critical: 6 hours voting + 1 hour delay

[a] 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396

[b] 0x555ba73dB1b006F3f2C7dB7126d6e4343aDBce02

[c] 0x213c446ec11e45b15a6E29C1C1b402B8897f606d"

[Certik, 02/19/2024] : The client has provided all steps towards mitigation on the BNB chain. However, we leave this finding as *acknowledged* until it can be verified on chain.

RSV-02 | CONTRACT UPGRADE CENTRALIZATION RISK

Category	Severity	Location	Status
Centralization	● Major	MarketCapsRiskSteward.sol (RiskSteward Base): 24; RiskStewardReceiver.sol (RiskSteward Base): 25	● Acknowledged

Description

The contracts `MarketCapsRiskSteward` and `RiskStewardReceiver` are upgradeable; the corresponding `admin` role in each respective proxy has the authority to update the implementation contract behind each contract.

Any compromise to the `admin` account in each proxy may allow a hacker to take advantage of this authority and change the implementation contract the proxy points to, and therefore execute potential malicious functionality in the implementation contract.

Recommendation

We recommend that the team make efforts to restrict access to the admin of the proxy contract. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (2/3, 3/5) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND

- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

AND

- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.

OR

- Remove the risky functionality.

I Alleviation

[Venus, 02/14/2025] : "The admin of the contracts on BNB Chain (where these contracts will be deployed) will be the ProxyAdmin contract `0x1BB765b741A5f3C2A338369DAb539385534E3343` .

The owner of this ProxyAdmin contract is the Normal Timelock contract (`0x939bD8d64c0A9583A7Dcea9933f7b21697ab6396`), used to execute the normal Venus Improvement Proposals (VIP) on BNB chain. For normal VIPs, the time configuration is: 24 hours voting + 48 hours delay before the execution.

So, these contracts will be upgraded only via a Normal VIP, involving the Venus Community/Governance in the process."

[Certik, 02/18/2025] : The client has provided all steps towards mitigation on the BNB chain. However, we leave this finding as *acknowledged* until it can be verified on chain.

RSR-04 | UPDATES MAY BE CALLED OUT OF ORDER OR TO PREVENT NEWER UPDATES FROM BEING PROCESSED

Category	Severity	Location	Status
Logical Issue	● Medium	RiskStewardReceiver.sol (RiskSteward Base): 207	● Resolved

Description

There are scenarios where a previous update can be processed after a later update, causing the protocol to use earlier updates than desired. In addition, it allows for scenarios where a previous update can be processed to ensure that the latest update will expire before it can be processed.

Scenario

Lets assume that update1 and update2 are posted by the `RISK_ORACLE` and are for the same type and market. Furthermore, lets assume that update1 is posted before update2. Note that `updateId` is strictly increasing so that necessarily the `updateId` for update1 is less than the `updateId` for update2.

Scenario 1: If the update2 is posted by the `RISK_ORACLE` before `UPDATE_EXPIRATION_TIME - debounce` time of update1 and directly processing update2 before update1 still updates within the allowed range, then the following scenario can happen.

- No updates are processed between the time that update1 and update2 are posted by the `RISK_ORACLE`.
- update2 is processed by calling either `processUpdateById()` or `processUpdateByParameterAndMarket()` as it is the latest update.
- The debounce period goes by and then a user calls `processUpdateById()` and inputs the `updateId` for `update1`.
 - This succeeds as it is passed the debounce period, but still within the `UPDATE_EXPIRATION_TIME`.
 - The parameters are updated to an older update provided by the oracle.

Scenario2: If update2 is posted by the `RISK_ORACLE` within the `debounce` period of update1, then the following scenario can happen.

- No updates are processed until just before the update expiration time of update1.
- update1 is processed `UPDATE_EXPIRATION_TIME - 1` seconds after it is posted by the `RISK_ORACLE` by calling `processUpdateById()`.
 - The update succeeds as it is just under the expiration time.

- update2 is desired to be processed, however, it will be unable to be processed as the debounce period must pass before it can be called, after which time it will be expired.

Recommendation

We recommend ensuring that previous updates cannot be processed, which can be done by storing the last processed `updateId` for each type and market, and then checking that the `updateId` of the update to be processed for that market and type are greater than it. Furthermore, we recommend ensuring that `processUpdateById()` and `processUpdateByParameterAndMarket()` are called in a timely manner to prevent scenario 2.

Alleviation

[Certik, 02/11/2025] : The client updated the code to ensure that the debounce period is greater than the `UPDATE_EXPIRATION_TIME` so that Scenario 1 is not possible. In addition, they stated that Scenario 2 is acceptable and that it will be handled by resubmitting the update after the debounce period has passed. Alternatively it could be done via governance.

SFD-02 | FUNCTIONS WILL ALWAYS RETURN SUCCESS EVEN WHEN THEY FAIL

Category	Severity	Location	Status
Logical Issue	● Medium	SetterFacet.sol (Align Comptroller Facets Base): 122, 132, 141, 151, 191, 202	● Resolved

Description

The updates to the functions do not return the value obtained from the internal function call. As a result, they will always return the default value of 0, which is interpreted as success. This can result in assuming that the function was successfully called, when it was not, which can cause issues if further changes to the protocol are made assuming the function call was successful.

Recommendation

We recommend returning the value obtained from the internal function calls.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes resolving this finding in commit [bc49c803b7ad086bd3e1f43f6b2edad150b341c4](#).

RSR-05 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	RiskStewardReceiver.sol (RiskSteward Base): 147	Resolved

Description

The input `riskSteward` is not checked to be `address(0)`. If it set to be `address(0)`, then it allows a risk parameter configuration to be active while being unsupported, since to check if an update type is supported the following check is used in the codebase

```
if (riskParameterConfigs[updateType].riskSteward == address(0)) {  
    revert UnsupportedUpdateType();  
}
```

Recommendation

We recommend adding a check the passed-in address is not `address(0)` to prevent unexpected errors.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes resolving this finding in commit [45648b9dcc0fc74fc3546ea0dfc25d55433235db](#).

RSV-03 | MISSING INPUT VALIDATION

Category	Severity	Location	Status
Logical Issue	Minor	MarketCapsRiskSteward.sol (RiskSteward Base): 97, 111; RiskStewardReceiver.sol (RiskSteward Base): 152	Resolved

Description

The following input validations should be added to prevent unexpected behavior:

RiskStewardReceiver

- When `debounce` is set, it should be checked to be greater than the `UPDATE_EXPIRATION_TIME` to ensure that updates are not processed too quickly.

MarketCapsRiskSteward

- When setting the `maxDeltaBps` it is not checked to be less than some maximum value.

Recommendation

We recommend adding the input validations mentioned above.

Alleviation

[Certik, 02/11/2025]: The client made the recommended changes resolving this finding in commits

- [fba41ad41d0a3497742157f1b030e8ffde7bb789](#)
- [02d89861ecddb947bfe1165ba8ddb0485f7c5cd9](#)

MCR-01 | UNNECESSARY CASES IF CORE COMPTROLLER INTERFACE IS UPDATED TO BE COMPATIBLE WITH ISOLATED POOLS INTERFACE

Category	Severity	Location	Status
Logical Issue	● Informational	MarketCapsRiskSteward.sol (RiskSteward Base): 144~149, 159~163	● Acknowledged

Description

If the changes in scope of this audit are made to update the Core Comptroller Interface to be compatible with the Isolated Pools Interface prior to deploying the risk steward contracts, then the `IIsolatedPoolsComptroller` interface can be used for all comptrollers allowing the cited logic to be simplified.

Recommendation

We recommend clarifying if the risk steward contracts will be deployed prior to the updates to the Core Comptroller Interface.

Alleviation

[Venus, 02/06/2025]: "Our original deployment plan was to first publish and start using this risk steward. Then in a second deployment we will update the Core Pool Comptroller.

The next phase of the risk steward will include removing the core pool interface as well as other updates."

MCR-03 | EVENT NOT INDEXED

Category	Severity	Location	Status
Design Issue	● Informational	MarketCapsRiskSteward.sol (RiskSteward Base): 53, 58, 63	● Resolved

Description

If an event is not indexed in a smart contract, it means that the event's parameters are not tagged with the `indexed` keyword. This has implications for how the event data can be searched and filtered when looking through blockchain logs.

Without indexing, the event will still emit the data as part of the transaction log, but users won't be able to query for these events using the parameters. They'll have to retrieve the entire set of logs and manually sift through them to find events with the specific data. This can be less efficient and more time-consuming, especially on a blockchain with a high volume of transactions and events.

Recommendation

To mitigate this issue, it is recommended to index the most relevant parameters in the event to be defined.

Alleviation

[Certik, 02/10/2025] : The client made the recommended changes resolving this finding in commit [e56271b5b97e6617b6c59588fdb544ab0f29587a](#).

MCR-04 | USAGE OF MAGIC NUMBERS

Category	Severity	Location	Status
Coding Issue	● Informational	MarketCapsRiskSteward.sol (RiskSteward Base): 202	● Resolved

Description

The contract contains "magic numbers" (hardcoded numeric values) without any explanation or constants to define their purpose. This reduces code readability and maintainability, making auditing harder and potentially hiding unintended logic or vulnerabilities.

Recommendation

We recommend to define all numeric values as named constants with descriptive names that explain their purpose.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes resolving this finding in commit [533a943df93fd38ef8d0fcedcc8acb290220b291](#).

MCR-06 | UNNECESSARY INHERITANCE

Category	Severity	Location	Status
Coding Style	● Informational	MarketCapsRiskSteward.sol (RiskSteward Base): 24	● Resolved

Description

The `MarketCapsRiskSteward` inherits `AccessControlledV8` which already inherits `Ownable2StepUpgradeable` and `Initializable`, making it unnecessary to inherit them again. In addition instead of calling `__Ownable2Step_init` and `__AccessControlled_init_unchained()` separately, `__AccessControlled_init()` can be called.

Similarly this is the case for `RiskStewardReceiver` as well.

Recommendation

We recommend removing any unnecessary inheritance and calling `__AccessControlled_init()` to simplify the process.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes in commit [5e3ba094d92d317e68ec3fcce786f0deb64de77a](#).

MFD-01 | INCONSISTENT GROUPING OF FUNCTIONS

Category	Severity	Location	Status
Logical Issue	● Informational	MarketFacet.sol (Align Comptroller Facets Base): 248~255	● Resolved

Description

The function `isMarketListed()` is not grouped with the other external view functions.

Recommendation

We recommend grouping the functions consistently to improve readability.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes resolving the finding in commit [5fd60ea17e00621a477d553489886a457f271592](#).

RSR-06 | INDEXED DYNAMIC DATA TYPE IN EVENT

Category	Severity	Location	Status
Design Issue	● Informational	RiskStewardReceiver.sol (RiskSteward Base): 61, 73	● Resolved

Description

When attempting to index dynamic data types like `string`, `bytes`, `array`, or `struct` in Solidity, they don't get stored in their original form. Instead, the Ethereum log system stores the `Keccak-256` hash of these data types.

While this approach ensures efficiency and cost-effectiveness, developers must be aware of it to correctly use and interpret logs, and they cannot retrieve the original string from its hash alone.

Recommendation

We recommend ensuring this behavior aligns with the expected design.

Alleviation

[Certik, 02/10/2025]: The client opted to remove the indexing as it improves readability in commit [97d929ae7e822e1478f5348735d61c8fc4332545](#).

RSR-07 | DISABLED FUNCTION DOES NOT EMIT AN ERROR

Category	Severity	Location	Status
Inconsistency	● Informational	RiskStewardReceiver.sol (RiskSteward Base): 258~261	● Resolved

Description

The function `renounceOwnership()` is disabled and does not emit an error. This can lead to confusion if it is called, as the call will succeed but do nothing.

Recommendation

We recommend considering adding a revert and error or clarifying why it is necessary to still have calls to

`renounceOwnership()` succeed.

Alleviation

[Certik, 02/18/2025]: The client made the recommended changes resolving this finding in commits

- [850443bab6e1af9b1bcdcf7b84975556c14dff7b;](#)
- [a8ea183304ce28bd2afa6078b843edf98b27e831.](#)

RSV-04 | TYPOS AND INCONSISTENCIES

Category	Severity	Location	Status
Inconsistency	● Informational	MarketCapsRiskSteward.sol (RiskSteward Base): 61, 76, 104, 105, 106, 120, 123, 143; RiskStewardReceiver.sol (RiskSteward Base): 81	● Resolved

Description

The following typos and inconsistencies were found in the codebase:

RiskStewardReceiver

- The comment above the error `ConfigNotActive` are not consistent with the error.

MarketCapsRiskSteward

- Often `increase` is used when referring to the max bps, however, it is a limit for increasing and decreasing.
- The comment above the error `UpdateNotInRange` uses "our" when it should use "out".
- The comment for the parameter `update` in the function `processUpdate()` does not describe the parameter.
- The comments above the function `processUpdate()` state that its access is controlled by `AccessControlManager`, when it is only callable by the `RISK_STEWARD_RECEIVER`. The `RiskStewardReceiver` may make calls to this in the functions `processUpdateById()` and `processUpdateByParameterAndMarket()` which can be called by any user.
- There is an extra line spacing in the function `_updateSupplyCaps()` that is not consistent with the spacing of the contract.

Recommendation

We recommend fixing the typos and inconsistencies mentioned above.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes resolving the finding in commit [7062f96528df4e79ea13f4ebf78571f74d2f3f47](https://github.com/certiklabs/venus-rsk-oracle-integration/commit/7062f96528df4e79ea13f4ebf78571f74d2f3f47).

RSV-05 | UNNECESSARY IMPORTS

Category	Severity	Location	Status
Coding Style	● Informational	MarketCapsRiskSteward.sol (RiskSteward Base): 7, 11, 12; RiskStewardReceiver.sol (RiskSteward Base): 9~11	● Resolved

Description

The cited imports are not used within their respective contracts and can be removed.

Recommendation

We recommend removing unused imports.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes resolving the finding in commit [1c6b6fdd294e48a85410f1767b00c8cb88c2466d](#).

SFD-03 | ALIASED FUNCTION NOT INCLUDED IN UPDATED INTERFACE

Category	Severity	Location	Status
Inconsistency	● Informational	SetterFacet.sol (Align Comptroller Facets Base): 315~323	● Resolved

Description

The `setActionsPaused()` function is an aliased version of the `_setActionsPaused()` function. However, it is not included in the updated `ISetterFacet` interface.

Recommendation

We recommend including it in the interface or if it is not intended to be aliased removing the alias function.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes resolving this finding in commit [f55d503a39eceffc119b601a635ff1724c26e4fa](#).

VPB-01 | MISSING OR INCOMPLETE NATSPEC

Category	Severity	Location	Status
Inconsistency	● Informational	MarketCapsRiskSteward.sol (RiskSteward Base): 86, 94, 109, 125, 138, 153, 167, 172, 177, 185, 199, 209; RiskStewardReceiver.sol (RiskSteward Base): 111, 117, 147, 185, 204~205, 224, 231, 235; PolicyFacet.sol (Align Comptroller Facets Base): 413; SetterFacet.sol (Align Comptroller Facets Base): 284, 498, 576, 595, 619, 658, 676, 690, 704, 715, 724	● Partially Resolved

Description

RiskStewardReceiver

- `constructor()` and `initialize()` do not have NatSpec comments.
- The NatSpec comments for `setRiskParameterConfig()` do not include the `riskSteward` parameter, the event emitted, or the potential errors.
- The NatSpec comments for `toggleConfigActive()` do not include the access control, events, or errors.
- The NatSpec comments for `processUpdateById()` include the errors `UpdateNotInRange` and `UnsupportedUpdateType`. However, these errors are emitted by the `riskSteward` contract being called and may not always be emitted if new `riskStewards` are utilized.
- The NatSpec comments for `processUpdateByParameterAndMarket()` do not include the events or errors.
- The functions `_processUpdate()`, `_getMarketUpdateTypeKey()`, and `_validateUpdateStatus()` do not have NatSpec comments.

MarketCapsRiskSteward

- `constructor()` and `initialize()` do not have NatSpec comments.
- The NatSpec comments for `setMaxDeltaBps()` do not include the event.
- The NatSpec comments for `processUpdate()` do not include the potential events it may emit.
- The NatSpec comments for `_updateWithinAllowedRange()` do not include the error.
- The functions `_updateSupplyCaps()`, `_updateBorrowCaps()`, `_processSupplyCapUpdate()`, `_processBorrowCapUpdate()`, `_validateSupplyCapUpdate()`, `_validateBorrowCapUpdate()`, and `_decodeBytesToUint256()` are missing NatSpec comments.

We assume that all functions should include NatSpec comments. which should include a description of the function, parameters, return values, errors, events, and access control. Please let us know if any if there is another convention you are following in terms of what NatSpec comments will be included for each function.

PolicyFacet

- The NatSpec comments for the function `getBorrowingPower()` do not include the return values.

SetterFacet

- The NatSpec comments for the function `setPrimeToken()` do not include the return value.
- The NatSpec comments for the parameter `vTokens` of the function `setMarketSupplyCaps()` reference changing borrow caps, when it should reference changing supply caps.
- The functions `__setPriceOracle()`, `__setCloseFactor()`, `__setCollateralFactor()`, `__setLiquidationIncentive()`, `__setMarketBorrowCaps()`, `__setMarketSupplyCaps()`, `__setPrimeToken()`, `__setForcedLiquidation()`, and `__setActionsPaused()` do not have NatSpec comments.

Recommendation

We recommend adding and completing the NatSpec comments mentioned above.

Alleviation

[Certik, 02/11/2025]: The client partially resolved the findings in commits

- [6416462dc1550fefe06817be8671cdfc29ac4de2;](#)
- [01297a13036f0c6c439468d7cfc3e1178763d15c;](#)
- [aa316c5062597111f58bdf090fa891d1fe923440.](#)

However, some of the newly added NatSpec comments do not include all parameters and return values.

OPTIMIZATIONS | VENUS - RISKORACLE INTEGRATION

ID	Title	Category	Severity	Status
<u>RSR-01</u>	User-Defined Getters	Gas Optimization	Optimization	● Resolved
<u>RSR-02</u>	Repeat Calculation	Code Optimization	Optimization	● Resolved

RSR-01 | USER-DEFINED GETTERS

Category	Severity	Location	Status
Gas Optimization	● Optimization	RiskStewardReceiver.sol (RiskSteward Base): 177~179	● Resolved

Description

The linked functions are equivalent to the compiler-generated getter functions for the respective variables.

Recommendation

We recommend removing any redundant getter functions.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes resolving this finding in commit [ca42c0a2cc0fdd2175a5b7dc71477766acc5902f](#).

RSR-02 | REPEAT CALCULATION

Category	Severity	Location	Status
Code Optimization	● Optimization	RiskStewardReceiver.sol (RiskSteward Base): 209~210, 220~221, 226, 251	● Resolved

Description

`_validateUpdateStatus()` and `_processUpdate()` both do the same calculation with `_getMarketUpdateTypeKey()`.

Recommendation

We recommend taking the result of the calculation as input to avoid repeating the calculation.

Alleviation

[Certik, 02/10/2025]: The client made the recommended changes resolving the finding in commit [f56d56622933981d6ea819060f249fd579245d2e](#).

APPENDIX | VENUS - RISKORACLE INTEGRATION

Finding Categories

Categories	Description
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Elevating Your Entire **Web3** Journey

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

