CERTIK

Security Assessment

Venus - RiskFund & Shortfall

CertiK Assessed on Aug 24th, 2023

CertiK Assessed on Aug 24th, 2023

Venus - RiskFund & Shortfall

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Binance Smart Chain (BSC)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 08/24/2023	N/A

CODEBASE

https://github.com/VenusProtocol/isolated-pools

View All in Codebase Page

COMMITS

base: 0f763af92f2c08c699b879a52b887978e7f0af3b

update1: 9930fd5298ddc677992b28360fc6eb7931ff6dcd

View All in Codebase Page

Vulnerability Summary

7	4	2	0	1	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	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 Mitigated	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.
1	Medium	1 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.
2	Informational	1 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 - RISKFUND & SHORTFALL

Summary

- Executive Summary
- Vulnerability Summary
- Codebase
- Audit Scope
- Approach & Methods

Summary

Dependencies

- Third Party Dependencies
- Assumptions
- Recommendations

Findings

- VPB-02 : Centralized Control of Contract Upgrade
- VPB-03 : Centralization Related Risks
- SSV-01 : `_transferOutOrTrackDebt()` Is Not Used When Closing Auction
- RFR-03 : Issue With `_swapAsset()` Depending On `convertibleBaseAsset` Price
- VPB-04 : Ineffectual Use of Reentrancy Guard
- SSV-03 : Potential To Fill Blocks To Stop Others Bids To Win Auction
- VPB-05 : Typos and Inconsistencies

Optimizations

- RFR-01 : Redundant Check on `amountOutMin`
- RHR-01 : View Function Equivalent to Compiler-Generated Getters
- SSV-04 : Check on Auction Status Can Be Simplified
- SSV-06 : Unnecessary Update to `marketDebt` in `_startAuction()`
- TDT-01 : Unchecked Blocks Can Optimize Contract
- VPB-01 : Custom Errors Can Be Used

Appendix

Disclaimer

CODEBASE | VENUS - RISKFUND & SHORTFALL

Repository

https://github.com/VenusProtocol/isolated-pools

Commit

base: 0f763af92f2c08c699b879a52b887978e7f0af3b
update1: 9930fd5298ddc677992b28360fc6eb7931ff6dcd

AUDIT SCOPE | VENUS - RISKFUND & SHORTFALL

8 files audited ● 5 files with Acknowledged findings ● 3 files without findings

ID	Repo	Commit	File	SHA256 Checksum
PSR	VenusProtocol/isolated-pools	0f763af	contracts/RiskFund/ProtocolShareReserve.sol	bdac4ddee86e6919f5b7f0c5a03d72c6df5daed83cecc39b4e88014af598c9eb
RHR	VenusProtocol/isolated-pools	0f763af	contracts/RiskFund/ReserveHelpers.sol	e5cd2a42208471dd5f66a8459691749cbc10b4caa51adef18e4ca7c9db3fbed1
RFR	VenusProtocol/isolated-pools	0f763af	contracts/RiskFund/RiskFund.sol	3f8fa6770320bf50b9a4d7552a7ed0d921bbc12263b7224ad6f289dae1b0d76f
SSV	VenusProtocol/isolated-pools	0f763af	contracts/Shortfall/Shortfall.sol	19b5980d593e0c67ea9a6d3610d5d2735c8e50cb413dc767cdd97cc6ce46d63e
TDT	VenusProtocol/isolated-pools	0f763af	contracts/lib/TokenDebtTracker.sol	06310e241b8639de2fb4b5b0849ff9ee47264d2ec938143ea8a8aed654dd955e
IPS	VenusProtocol/isolated-pools	0f763af	contracts/RiskFund/IProtocolShareReserve.sol	096a079173f83a2173ab7862a86b37864a889e0e2a42a17f4ffa58a3768e7723
IRF	VenusProtocol/isolated-pools	0f763af	contracts/RiskFund/IRiskFund.sol	a16ccd4db27697be2d5c3e214ee2b3be03a70fe2b4f7d4af4f230821b74e82b4
ISS	VenusProtocol/isolated-pools	0f763af	contracts/Shortfall/IShortfall.sol	65900b10ebd634a44df655370524e59c593cd08bcdf4fa7ec32b435f35261a86

APPROACH & METHODS | VENUS - RISKFUND & SHORTFALL

This report has been prepared for Venus to discover issues and vulnerabilities in the source code of the Venus - RiskFund & Shortfall 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 - RISKFUND & SHORTFALL

This audit only concerns the changes introduced from commit: 108aa561f07115b8a571bfba92c89f6055b675c7 to commit 0f763af92f2c08c699b879a52b887978e7f0af3b in the following files:

- contracts/RiskFund/RiskFund.sol
- contracts/RiskFund/IRiskFund.sol
- contracts/RiskFund/ProtocolShareReserve.sol
- contracts/RiskFund/IProtocolShareReserve.sol
- contracts/RiskFund/ReserveHelpers.sol
- contracts/Shortfall/Shortfall.sol
- contracts/lib/TokenDebtTracker.sol
- contracts/Shortfall/IShortfall.sol

DEPENDENCIES | VENUS - RISKFUND & SHORTFALL

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:

- ERC20 Tokens;
- AMMs such as PancakeSwap;

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 a 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 increasing fees of third parties, migrating to new LP pools, etc.

Assumptions

Within the scope of the audit, assumptions are made about the intended behavior of the protocol in order to inspect consequences based on those behaviors. Assumptions made within the scope of this audit include:

- underlying assets used with the protocol are not deflationary or inflationary
- underlying assets used with the protocol do not include callback logic and do not violate check-effect-interaction pattern;
- underlying assets are not double-entry point tokens

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 assumptions about the behavior of the project are thoroughly reviewed and, if the assumptions do not match the intention of the protocol, documenting the intended behavior for review.

FINDINGS | VENUS - RISKFUND & SHORTFALL

7	0	2	1	2	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Venus - RiskFund & Shortfall. Through this audit, we have uncovered 7 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
VPB-02	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	**Mitigated**
VPB-03	**Centralization Related Risks**	**Centralization**	**Major**	**Mitigated**
SSV-01	`_transferOutOrTrackDebt()` Is Not Used When Closing Auction	Logical Issue	Medium	Resolved
RFR-03	Issue With `_swapAsset()` Depending On `convertibleBaseAsset` Price	Logical Issue	Minor	Resolved
VPB-04	Ineffectual Use Of Reentrancy Guard	Coding Issue	Minor	Resolved
SSV-03	Potential To Fill Blocks To Stop Others Bids To Win Auction	Logical Issue	Informational	Acknowledged
VPB-05	Typos And Inconsistencies	Inconsistency	Informational	Resolved

VPB-02 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	Major	contracts/RiskFund/ProtocolShareReserve.sol (base): 13; contracts/RiskFund/ReserveHelpers.sol (base): 12; contracts/RiskFund/RiskFund.sol (base): 26; contracts/Shortfall/Shortfall.sol (base): 31~37	Mitigated

Description

In the contracts `Shortfall` , `RiskFund` , `ReserveHelpers` , and `ProtocolShareReserve` are upgradeable contracts and the `admin` of the proxy has the ability to update the implementation contract behind the proxy contract.

Any compromise to the `admin` account may allow a hacker to take advantage of this authority and change the implementation contract which is pointed by proxy and steal all tokens held by the contracts or implement and execute other potentially malicious functionality.

Recommendation

We recommend that the team make efforts to restrict access to the admin of the proxy contract. A strategy of combining a time-lock and a multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

Short Term:

A combination of a time-lock and a multi signature (⅔, ⅗) wallet mitigate the risk by delaying the sensitive operation and avoiding a single point of key management failure.

- A time-lock with reasonable latency, such as 48 hours, for awareness of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromised;
  AND
- A medium/blog link for sharing the time-lock contract and multi-signers addresses information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.

- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Long Term:

A combination of a time-lock on the contract upgrade operation and a DAO for controlling the upgrade operation mitigate the contract upgrade risk by applying transparency and decentralization.

- A time-lock with reasonable latency, such as 48 hours, for community awareness of privileged operations;
  AND
- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
  AND
- A medium/blog link for sharing the time-lock contract, multi-signers addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Permanent:

Renouncing ownership of the `admin` account or removing the upgrade functionality can *fully* resolve the risk.

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

Note: we recommend the project team consider the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

`[Venus, 08/11/2023]` : The admin of the contracts will the ProxyAdmin contract deployed at 0x6beb6D2695B67FEb73ad4f172E8E2975497187e4.

The owner of this ProxyAdmin contract is 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396, the Normal Timelock used to execute the normal Venus Improvement Proposals (VIP). For normal VIPs, the time config is: 24 hours voting + 48 hours delay before the execution.

So, this contract will be upgraded only via a Normal VIP, involving the community in the process.

VPB-03 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	Major	contracts/RiskFund/ProtocolShareReserve.sol (base): 57; contracts/RiskFund/ReserveHelpers.sol (base): 66; contracts/RiskFund/RiskFund.sol (base): 95, 107~108, 124, 135, 152~157, 194, 220; contracts/Shortfall/Shortfall.sol (base): 337, 351, 365, 378, 393, 406, 419	Mitigated

Description

Shortfall.sol

In the contract `Shortfall`, the role `onlyOwner()` has the authority over the function `updatePoolRegistry()`. Any compromise to the `onlyOwner` account may allow the hacker to take advantage of this authority and update the `poolRegistry` contract address.

In addition, the role `DEFAULT_ADMIN_ROLE` can grant addresses the privilege to call the following functions in the contract `Shortfall`:

- `updateNextBidderBlockLimit()`
- `updateIncentiveBps()`
- `updateMinimumPoolBadDebt()`
- `updateWaitForFirstBidder()`
- `pauseAuctions()`
- `resumeAuctions()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or these privileged functions may allow the hacker to take advantage of this authority and do the following:

- Update the time a bidder has to make the next bid before an auction can be closed. This can be used to win an auction for a low bid;
- Update the incentive BPS to allow them to potentially gain more funds from the auction;
- Update the minimum pool debt needed to start the auction;
- Update the time needed for the first bid before the auction will need to be restarted;
- Pause auctions, which prevents auctions from being started or restarted;
- Resume auctions, which allows auctions to be started or restarted;

ProtocolShareReserve.sol

In the contract `ProtocolShareReserve` the role `onlyOwner` has authority over the function `setPoolRegistry()`. Any compromise to the `onlyOwner` account may allow the hacker to take advantage of this authority and change the `poolRegistry` address.

RiskFund.sol

In the contract `RiskFund` the role `onlyOwner` has authority over the following functions:

- `setPoolRegistry()`
- `setShortfallContractAddress()`
- `setPancakeSwapRouter()`
- `setMaxLoopsLimit()`

Any compromise to the `onlyOwner` account may allow the hacker to take advantage of this authority and do the following:

- Set the `poolRegistry` address to a contract they control;
- Set the `shortfall` contract address to an address they control to steal any reserve that is supposed to be transferred for auction;
- Set the PancakeSwap router to a malicious contract to steal any tokens that are intended to be swapped;
- Change the max loops, which limits the amount of `markets` that can be input into `swapPoolsAssets()`;

In the contract `RiskFund` the role `shortfall` has authority over the function `transferReserveForAuction()`. Any compromise to the `shortfall` account may allow the hacker to take advantage of this authority and steal all reserves from the contract.

The role `DEFAULT_ADMIN_ROLE` can grant addresses the privilege to call the following functions in the contract `RiskFund`.

- `swapPoolsAssets()`
- `setMinAmountToConvert()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or these privileged functions may allow the hacker to take advantage of this authority and do the following:

- Swap a pools assets to the convertible base asset as well as set the minimum output amounts. If a hacker gained access to this, they can set a low minimum output amount and manipulate the pool to steal assets;
- Set the minimum amount that must be converted in order to swap assets.

ReserveHelpers.sol

In the contract `ReserveHelpers` the role `owner` has authority over the function `sweepToken()`. Any compromise to the `owner` account may allow the hacker to take advantage of this authority and remove any token in excess of its recorded `assetsReserves` mapping value. In particular, if the attacker has control over the `owner` address and the `protocolIncome` address of `protocolShareReserve`, then it is possible to use reentrancy in the `releaseFunds()`

function to call into function `sweepToken()` before the state update in `releaseFunds()` is complete, potentially allowing for the removal of more tokens than intended by the protocol.

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 (⅔, ⅗) 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.

Alleviation

`[Venus, 08/24/2023]`: The owner of the contracts RiskFund, Shortfall and ProtocolShareReserve will be 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396, that is the Timelock contract used to execute the normal Venus Improvement Proposals (VIP). For normal VIPs, the time config is: 24 hours voting + 48 hours delay before the execution.

So, only the community, via a VIP will be able to execute the mentioned protected functions.

We'll use the AccessControlManager (ACM) deployed at 0x4788629abc6cfca10f9f969efdeaa1cf70c23555.

In this ACM, only 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 (Normal) has the DEFAULT_ADMIN_ROLE. And this contract is a Timelock contract used during the Venus Improvement Proposals.

The idea is to grant 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 to execute every mentioned function. Moreover, we'll allow [a] (Fast-track) and [b] (Critical) also to execute the following functions:

Shortfall:

```
updateNextBidderBlockLimit()

updateIncentiveBps()

updateMinimumPoolBadDebt()

updateWaitForFirstBidder()

pauseAuctions()

resumeAuctions()
```

RiskFund:

```
setMinAmountToConvert()
```

Specifically, the current config for the three Timelock contracts are:

normal: 24 hours voting + 48 hours delay

fast-track: 24 hours voting + 6 hours delay

critical: 6 hours voting + 1 hour delay

[a] 0x555ba73dB1b006F3f2C7dB7126d6e4343aDBce02

[b] 0x213c446ec11e45b15a6E29C1C1b402B8897f606d

Only the multisig wallet deployed at 0x7B1AE5Ea599bC56734624b95589e7E8E64C351c9 will be authorized to invoke

`RiskFund.swapPoolsAssets()`.

`[CertiK, 08/24/2023]`: The multisig wallet deployed at 0x7B1AE5Ea599bC56734624b95589e7E8E64C351c9 requires 3 out of the following 6 signers:

- 0xda49D166a75014264DA854a5038A407AFE7c4939
- 0x6C383f92DDBdB8e6813cD0C283aE775332467E42
- 0xa81a457Df460cad4F41f986724Ed4fAd807990D0
- 0x5EaacE704186C4DB1bdC0840ef4C6b2F9579775e
- 0xa05f990d647287e4E84715b813BC000aEA970467
- 0x55A9f5374Af30E3045FB491f1da3C2E8a74d168D

Considering the only action not subjected to a time-lock is time-sensitive and protected by a multi-sig, we have marked this as *mitigated*. However, this does not completely eliminate the risk and users and the team should constantly monitor these privileges.

SSV-01 `_transferOutOrTrackDebt()` IS NOT USED WHEN CLOSING AUCTION

Category	Severity	Location	Status
Logical Issue	Medium	contracts/Shortfall/Shortfall.sol (base): 289	Resolved

Description

The function `_transferOutOrTrackDebt()` was added to avoid auctions becoming locked if a user tokens are supported that may revert when transferring to certain addresses. For example tokens that revert when transferring to a blacklisted address or tokens that implement hooks that call the receiver allowing them to force the transaction to revert. However, the function `closeAuction()` uses `safeTransfer()` and not `_transferOutOrTrackDebt()` when transferring tokens to the winner of the auction.

Thus if the winner of the auction can cause the transfer of tokens to revert, they can prevent the auction from being closed and future auctions from being started.

Recommendation

We recommend using `_transferOutOrTrackDebt()` in place of `safeTransfer()` to protect against the possibility of the transfer of tokens reverting when transferring to the winner of the auction.

Alleviation

`[CertiK, 08/11/2023]`: The client made the recommended changes in commit: 5ca628c1a94abe654d8cc7461e0fe920898f808c.

RFR-03 | ISSUE WITH `_swapAsset()` DEPENDING ON `convertibleBaseAsset` PRICE

Category	Severity	Location	Status
Logical Issue	Minor	contracts/RiskFund/RiskFund.sol (base): 257~258, 275~277, 288~294	Resolved

Description

The variable `minAmountToConvert` stores the minimum amount assets must be worth to convert into base asset.

If the intent is to ensure that the value in USD of the received `convertibleBaseAsset` is at least the `minAmountToConvert`, then the logic does not ensure this. This is because the logic ensures that the swap gives at least the `amountOutMin` of the convertible base asset and that the `amountOutMin` is at least the `minAmountToConvert`. However, the value of the convertible base asset is not taken into account and may be less than 1 USD.

Alternatively, if it is not desired to ensure that the value in USD of the received `convertibleBaseAsset` is at least the `minAmountToConvert`, then the check that `amountOutMin >= minAmountToConvert` will prevent valid swaps. This can happen if the `convertibleBaseAsset` is worth more than 1 USD or if the USD value of `underlyingAsset` is close to the `minAmountToConvert` as swap fees and slippage must be accounted for.

Recommendation

We recommend taking either of the following actions:

1. If it is desired to ensure that the value of the received `convertibleBaseAsset` in USD is at least the `minAmountToConvert`, calculate the USD value of the `amountOutMin` and instead of checking that `amountOutMin >= minAmountToConvert`, check that the USD value of the `amountOutMin` is greater than or equal to the `minAmountToConvert`. Then the check that `amountInUsd >= minAmountToConvert` should be adjusted to account for fees and acceptable slippage. This is because if they are equal then the swap fees and slippage will cause the received USD value to be less than the `minAmountToConvert`.
2. Alternatively, ensure that the price of the `convertibleBaseAsset`, expected fees, and acceptable slippage are accounted for when setting the `amountOutMin`. In addition, set the deadline to a short timeframe to ensure that there cannot be dramatic price movement before the transaction is executed. In addition, we then recommend removing or changing the check that `amountOutMin >= minAmountToConvert`.

Alleviation

`[CertiK, 08/11/2023]`: The client made the recommended changes in commits:

- fe4494023045e891943c9761c07593e44113fc4f;

- 9930fd5298ddc677992b28360fc6eb7931ff6dcd.

VPB-04 | INEFFECTUAL USE OF REENTRANCY GUARD

Category	Severity	Location	Status
Coding Issue	Minor	contracts/RiskFund/ProtocolShareReserve.sol (base): 89~93, 95~96; contracts/RiskFund/ReserveHelpers.sol (base): 66~67, 76~77, 77~78; contracts/RiskFund/RiskFund.sol (base): 152~157, 194~195, 209~210, 211~212; contracts/Shortfall/Shortfall.sol (base): 199~203, 254~255	Resolved

Description

The reentrancy guard `nonReentrant` is applied to the following locations:

- `sweepToken()` in `ReserveHelpers`
- `placeBid()` in `Shortfall`
- `closeAuction()` in `Shortfall`

Applying the `nonReentrant` modifier to only one of the user-facing functions within each contract limits its effectiveness and use as vulnerability protection within each contract. The only case in which the guard provides protection is if the external call is initiated within a function that includes the modifier and attempts to exploit a function that includes the modifier.

For example, since `sweepToken()` is inherited by `ProtocolShareReserve()` and user-facing function `releaseFunds()` includes multiple external calls without using the reentrancy guard, there is potential for cross-function reentrancy that begins in `releaseFunds()` and is exploited via function `sweepToken()`.

Likewise, since only `placeBid()` and `closeAuction()` have the `nonReentrant` modifier and `startAuction()` and `restartAuction()` are user-facing and exclude the modifier, reentrancy is possible, starting from `closeAuction()` moving into either `startAuction()` or `restartAuction()`.

Locations where `nonReentrant` Can Be Added

Adding a `nonReentrant` modifier to the following locations will increase the effectiveness of the reentrancy guard.

`ProtocolShareReserve`

- `releaseFunds()`

`Shortfall`

- `startAuction()`
- `restartAuction()`

`RiskFund`

- `swapPoolsAssets()`
- `transferReserveForAuction()`

Locations where Check-Effect-Interaction Pattern Can Be Applied

In general, check-effect-interaction can be followed, where possible, to prevent more general forms of reentrancy.

- Function `sweepToken()` in `ReserveHelpers` can emit event `SweepToken` before external call to `IERC20Upgradeable(_token)`.
- Function `releaseFunds()` in `ProtocolShareReserve` can emit event `FundsReleased` before external call to `IERC20Upgradeable(asset)`.
- Function `transferReserveForAuction()` in `RiskFund` can emit event `TransferredReserveForAuction` before external call to `IERC20Upgradeable(convertibleBaseAsset)`.

Recommendation

We recommend adding the `nonReentrant` modifier where specified and following the check-effect-interaction pattern where listed to prevent and protect against reentrancy.

Alleviation

`[CertiK, 08/11/2023]`: The client made the recommended changes in commit: 42a9f78dbe59afa8477235f8a9be20bcacad6b7c.

SSV-03 POTENTIAL TO FILL BLOCKS TO STOP OTHERS BIDS TO WIN AUCTION

Category	Severity	Location	Status
Logical Issue	Informational	contracts/Shortfall/Shortfall.sol (base): 71	Acknowledged

Description

An auction can be closed once the `nextBidderBlockLimit` amount of blocks passes between bids and at least one bid has been placed. A user could make a low first bid and then submit transactions that will fill an entire block. In addition, they pay a high enough gas price to ensure their transaction will be chosen over other transactions. They do this for the `nextBidderBlockLimit` amount of blocks to ensure that no other user can bid and then close the auction.

For example, the block gas limit is ~140 million. The average gas price is 3 gwei, so let's assume 10 gwei would be sufficient to ensure their transaction is chosen. With these assumptions it would cost 1.4 BNB per block they wish to fill. Thus if the `nextBidderBlockLimit = 100`, then it would cost them 140 BNB, which if the price of BNB is 250 USD is about 35,000 USD. If the auction is for 50,000 USD worth of assets, then such an attack could be profitable. Note that even if the amount of assets is worth 35,000 USD or less, the attacker would not make a profit and possibly incur a small loss, but they may still wish to execute the attack to cause damage to the protocol. Thus it should be ensured that the amount of assets for auction is significantly less than the cost of such an attack.

Note that the example is a very rough estimate. It should be carefully considered what the minimum gas price an attacker could use to execute such an attack would be in order to have a proper estimate of the potential cost.

Recommendation

We recommend ensuring that the `nextBidderBlockLimit` is set high enough based on the value of the auction, in order to make such an attack incur a significant loss.

Alleviation

`[Venus, 08/11/2023]`: Issue acknowledged. I will fix the issue in the future, which will not be included in this audit engagement.

VPB-05 | TYPOS AND INCONSISTENCIES

Category	Severity	Location	Status
Inconsistency	Informational	contracts/RiskFund/RiskFund.sol (base): 21; contracts/Shortfall/Shortfall.sol (base): 89~90, 346~347	Resolved

Description

- In the contract `Shortfall`, the comment above `nextBidderBlockLimit` states: "Time to wait for next bidder. initially waits for 10 blocks". However, the `DEFAULT_NEXT_BIDDER_BLOCK_LIMIT` is now 100 instead of 10.
- In the contract `Shortfall`, the comment above `updateIncentiveBPs()` contains a typo where the word "incentive" is misspelled as "inventive";
- In the contract `RiskFund`, the comment at the start of the contract states: "@title ReserveHelpers". However, the title should be "RiskFund".

Recommendation

We recommend fixing the typos and inconsistencies mentioned above.

Alleviation

`[CertiK, 08/11/2023]`: The client made the recommended changes in commit: 5effe7b6714855ced420295f849a06fe89f5dd59.

OPTIMIZATIONS | VENUS - RISKFUND & SHORTFALL

ID	Title	Category	Severity	Status
RFR-01	Redundant Check On `amountOutMin`	Code Optimization, Gas Optimization	Optimization	Resolved
RHR-01	View Function Equivalent To Compiler-Generated Getters	Code Optimization	Optimization	Resolved
SSV-04	Check On Auction Status Can Be Simplified	Gas Optimization, Code Optimization	Optimization	Resolved
SSV-06	Unnecessary Update To `marketDebt` In `_startAuction()`	Gas Optimization, Logical Issue	Optimization	Acknowledged
TDT-01	Unchecked Blocks Can Optimize Contract	Gas Optimization	Optimization	Acknowledged
VPB-01	Custom Errors Can Be Used	Gas Optimization	Optimization	Acknowledged

RFR-01 | REDUNDANT CHECK ON `amountOutMin`

Category	Severity	Location	Status
Code Optimization, Gas Optimization	Optimization	contracts/RiskFund/RiskFund.sol (base): 257~258	Resolved

Description

The input `amountOutMin` in function `_swapAsset()` of contract `RiskFund` includes two checks:

```
        require(amountOutMin != 0, "RiskFund: amountOutMin must be greater than 0 to swap vToken");
        require(amountOutMin >= minAmountToConvert, "RiskFund: amountOutMin should be greater than minAmountToConvert");
```

Only the latter check is needed, since after initialization of the contract, `minAmountToConvert` is necessarily a positive integer. Therefore, checking that `amountOutMin` is at least the `minAmountToConvert` ensures that the value is nonzero, and the explicit check is not required.

Recommendation

We recommend removing any redundant checks.

Alleviation

`[CertiK, 08/11/2023]`: The client removed the redundancy by removing the check that `amountOutMin >= minAmountToConvert` in commit: fe4494023045e891943c9761c07593e44113fc4f.

RHR-01 VIEW FUNCTION EQUIVALENT TO COMPILER-GENERATED GETTERS

Category	Severity	Location	Status
Code Optimization	Optimization	contracts/RiskFund/ReserveHelpers.sol (base): 24~25, 87~88	Resolved

Description

Mapping `poolsAssetsReserves` has been changed to `public` visibility, which includes a compiler-generated getter function for the mapping. However, function `getPoolAssetReserve()` returns the same information and is still used in many locations throughout the codebase.

The only difference in these functions is that `getPoolAssetReserve()` includes checks on the `comptroller` and `asset` address used.

Recommendation

We recommend leaving `poolsAssetsReserves` as `internal` visibility to streamline the codebase.

Alleviation

`[CertiK, 08/11/2023]`: The client made the recommended changes in commit: 908578705a8ab2cb6058bca205a6af3083895c13.

SSV-04 | CHECK ON AUCTION STATUS CAN BE SIMPLIFIED

Category	Severity	Location	Status
Gas Optimization, Code Optimization	Optimization	contracts/Shortfall/Shortfall.sol (base): 436~437, 533~534	Resolved

Description

The following check in `_startAuction()` can be streamlined:

```
        require(
            (auction.startBlock == 0 && auction.status == AuctionStatus.NOT_STARTED)
```

There is never an instance in which both `auction.startBlock` is 0 and `auction.status` is not `NOT_STARTED`. There is never an instance in which both `auction.status` is `NOT_STARTED` and `auction.startBlock` is nonzero. Therefore, the two conditions are equivalent, and the conjunction can be reduced to just one of the checks.

The following check in `_isStarted()` can be streamlined:

```
        return auction.startBlock != 0 && auction.status == AuctionStatus.STARTED;
```

The check that the `auction.startBlock` is nonzero is not necessary. If `auction.status` is `STARTED` then it is implicitly true that `auction.startBlock` is nonzero. Therefore, the check that `auction.status` is `STARTED` is sufficient, and the conjunction can be reduced to just check the status.

Recommendation

We recommend streamlining the checks mentioned above, as described.

Alleviation

`[CertiK, 08/11/2023]`: The client made the recommended changes in commit: 1dbf136d1daeda5b4f662c3f0635b194faa7f5cc.

SSV-06 UNNECESSARY UPDATE TO `marketDebt` IN `_startAuction()`

Category	Severity	Location	Status
Gas Optimization, Logical Issue	Optimization	contracts/Shortfall/Shortfall.sol (base): 444~448, 468~469	Acknowledged

Description

In function `_startAuction()`, each `vToken` that was a recorded market in the specified `comptroller` is updated in mapping `auction.marketDebt` to a value of 0 before a new list of markets is read from the `comptroller` and used to update the mapping with new values for `auction.marketDebt`.

Currently, markets can only be added to a `comptroller` and not removed. As a consequence, the `allmarkets` array in the `comptroller` is monotonically increasing, and retains all old markets.

This being the case, there is no need to zero out the entries of the `auction.marketDebt` mapping for the previous markets, because old markets will be retained and their entries are guaranteed to be updated.

Recommendation

We recommend removing the `for` loop which zeroes out the values of `auction.marketDebt` since these entries are guaranteed to be updated by the current design of the `comptroller`.

Alleviation

`[Venus, 08/11/2023]`: Issue acknowledged. I won't make any changes for the current version.

We prefer to not introduce additional implicit assumptions on other contracts' behavior.

TDT-01 | UNCHECKED BLOCKS CAN OPTIMIZE CONTRACT

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/lib/TokenDebtTracker.sol (base): 91, 140~141	Acknowledged

Description

The `totalTokenDebt[token]` is the sum of all `tokenDebt[token][user]`, so that `totalTokenDebt[token] >= tokenDebt[token][user]` for any user. This allows for unchecked blocks to be used in places where overflow or underflow is not possible.

- In the function `claimTokenDebt()`, the check that reverts if `amount > owedAmount` ensures that `tokenDebt[token][msg.sender] = owedAmount - amount` will not underflow, which is why it is placed in an unchecked block. However, this also ensures that `totalTokenDebt[token] -= amount` will not underflow as `totalTokenDebt[token] >= tokenDebt[token][msg.sender] = owedAmount`, so that it can be placed in an unchecked block.
- In the function `_transferOutOrTrackDebtSkippingBalanceCheck()`, if `totalTokenDebt[token] += amount` does not overflow, then `tokenDebt[token][to] += amount` cannot overflow as `totalTokenDebt[token] >= tokenDebt[token][user]` and they are both `uint256`. Thus `totalTokenDebt[token]` can be incremented first which will be checked for overflow and then in an unchecked block `tokenDebt[token][to]` can be incremented as it cannot overflow.

Recommendation

We recommend adding the unchecked blocks above to save gas.

Alleviation

`[Venus, 08/11/2023]`: Issue acknowledged. I will fix the issue in the future, which will not be included in this audit engagement.

VPB-01 | CUSTOM ERRORS CAN BE USED

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/RiskFund/ProtocolShareReserve.sol (base): 78; contracts/RiskFund/ReserveHelpers.sol (base): 53, 71, 89, 102, 104~108; contracts/RiskFund/RiskFund.sol (base): 77~78, 109~112, 137, 159, 162~163, 174~175, 196~200, 257~258, 280~284; contracts/Shortfall/Shortfall.sol (base): 177, 206~219, 257~261, 309, 321~323, 339, 353, 408, 421, 432, 435~439, 472	Acknowledged

Description

From Solidity `v0.8.4`, there are more gas-efficient ways to explain to users why an operation failed than through strings. Using custom errors can significantly reduce the size of the deployed bytecode and reduce the gas cost when calls revert.

Recommendation

We recommend considering the use of custom errors to reduce gas costs. For more information see: https://blog.soliditylang.org/2021/04/21/custom-errors/.

Alleviation

`[Venus, 08/11/2023]`: Issue acknowledged. I will fix the issue in the future, which will not be included in this audit engagement.

APPENDIX | VENUS - RISKFUND & SHORTFALL

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 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.
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.

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.

CertiK | Securing the Web3 World

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.