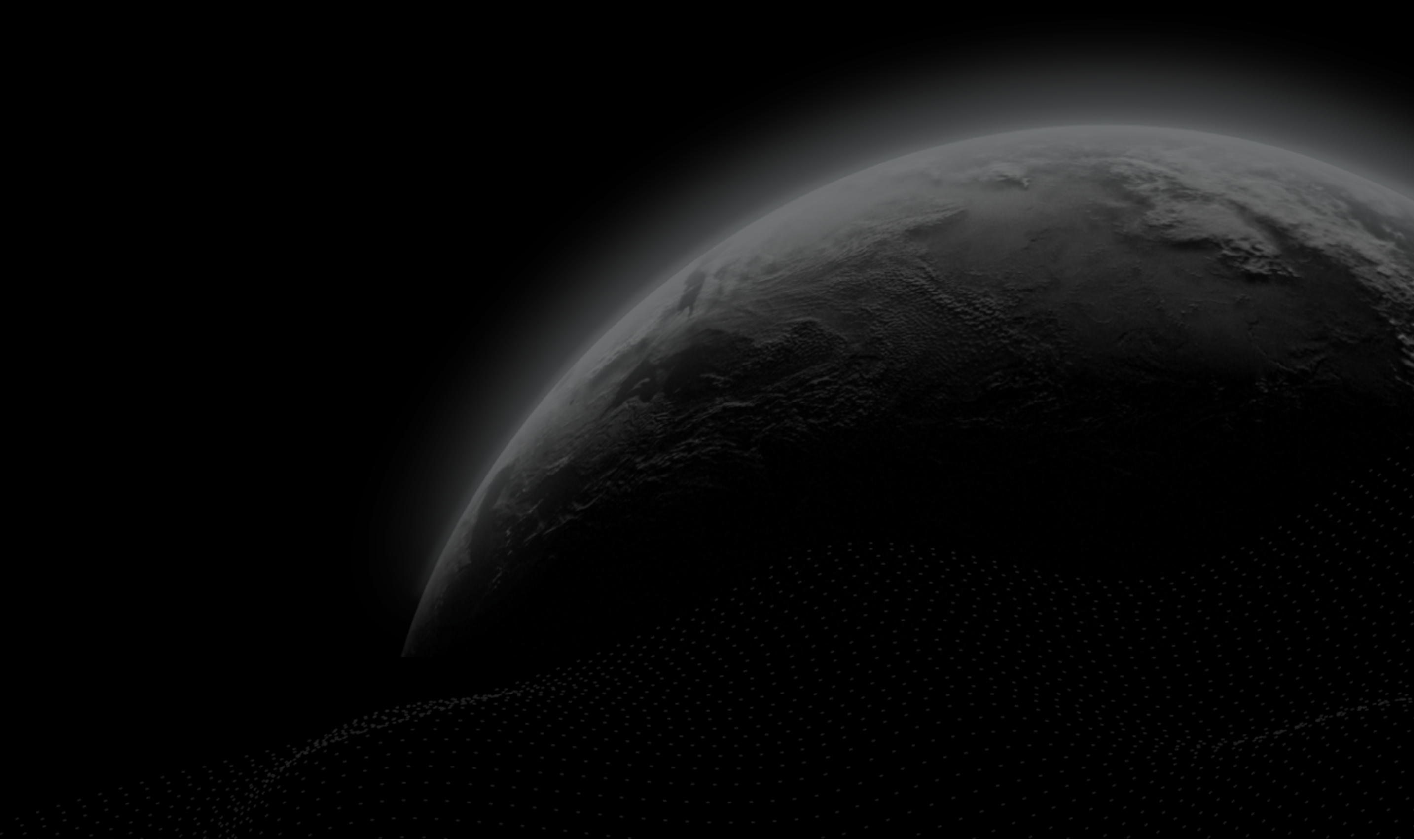
CERTIK

Security Assessment

Venus - Token Converter

CertiK Assessed on Nov 7th, 2023



CertiK Assessed on Nov 7th, 2023

Venus - Token Converter

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 11/07/2023	N/A

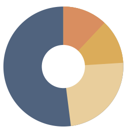
CODEBASE
protocol-reserve
View All in Codebase Page

COMMITS
base: e5ad0dffc7ef7b99d57cb9fe8947c9655c6cadc3
update1: 98779f04ec88d9a5abf3d148b4093adb08f11cdc
update2: 3e600898ea8522c50f1b1361e378866889de9f41
View All in Codebase Page

Highlighted Centralization Risks

ⓘ Contract upgradeability

Vulnerability Summary



25	18	2	1	4	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.
🟧 3	Major	1 Resolved, 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.
🟨 3	Medium	3 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨 6	Minor	4 Resolved, 1 Partially Resolved, 1 Acknowledged	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.

■	13	Informational	10 Resolved, 3 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 - TOKEN CONVERTER

CODEBASE | VENUS - TOKEN CONVERTER

Repository

protocol-reserve

Commit

base: e5ad0dffc7ef7b99d57cb9fe8947c9655c6cadc3

update1: 98779f04ec88d9a5abf3d148b4093adb08f11cdc

update2: 3e600898ea8522c50f1b1361e378866889de9f41

AUDIT SCOPE | VENUS - TOKEN CONVERTER

9 files audited ● 4 files with Acknowledged findings ● 1 file with Mitigated findings ● 4 files without findings

ID	Repo	Commit	File	SHA256 Checksum
● ATC	VenusProtocol/protocol-reserve	e5ad0df	TokenConverter/AbstractTokenConverter.sol	f2fbd791a98698c0e766b373f7ebe92e9554acab9833d8c75694487048a89e0c
● RFC	VenusProtocol/protocol-reserve	e5ad0df	TokenConverter/RiskFundConverter.sol	125a7bab4c65d5abfd8778c0c79d8495cca5ddba660ad7b5931f558206f60ca2
● XVS	VenusProtocol/protocol-reserve	e5ad0df	TokenConverter/XVSVaultConverter.sol	677a02b639c2ec3d8be62434e0cccd7789de3f8471a658260e5ac6c3e86de994
● RFV	VenusProtocol/protocol-reserve	e5ad0df	ProtocolReserve/RiskFundV2.sol	b38d97dc866e9cbf97e621c3baac634689d800d0fbe8568339c8c018880da963
● XVV	VenusProtocol/protocol-reserve	e5ad0df	ProtocolReserve/XVSVaultTreasury.sol	93a5a7f9973dbd4971fd41bdc4ddd05272695a5e9569293a3a2c2fcb62ac91c8
● IAT	VenusProtocol/protocol-reserve	e5ad0df	TokenConverter/IAbstractTokenConverter.sol	80d0b28a7ccf01b859cd1e3a593dc12014a934a7ed7d87f93cb84ec1e53923f4
● RFS	VenusProtocol/protocol-reserve	e5ad0df	ProtocolReserve/RiskFundStorage.sol	b8551de0c9a965a045b6c6f9d1b665ad561ec8185a11b367de189474d01acaea
● CUV	VenusProtocol/protocol-reserve	e5ad0df	Utils/Constants.sol	51e6a8b78635f990331591c2c2c88e929cae6faf4fa1490445d6da10c221654e
● VUV	VenusProtocol/protocol-reserve	e5ad0df	Utils/Validators.sol	07dc6575fba113948759519e8d06750abfc449dc9249d779173d2b93010bfa35

APPROACH & METHODS | VENUS - TOKEN CONVERTER

This report has been prepared for Venus to discover issues and vulnerabilities in the source code of the Venus - Token Converter 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 - TOKEN CONVERTER

This audit concerns the changes made in files outlined in PR: https://github.com/VenusProtocol/protocol-reserve/pull/9, with the commit audited being e5ad0dffc7ef7b99d57cb9fe8947c9655c6cadc3.

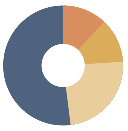
THIRD PARTY DEPENDENCIES | VENUS - TOKEN CONVERTER

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
- Oracles

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

FINDINGS | VENUS - TOKEN CONVERTER



25	0	3	3	6	13
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Venus - Token Converter. Through this audit, we have uncovered 25 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
RFV-03	RiskFundV2 Missing Or Improper Implementation	Logical Issue	Major	● Resolved
VPB-07	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	● **Mitigated**
VPB-08	**Centralization Related Risks**	**Centralization**	**Major**	● **Mitigated**
ATC-03	Issues With Functions Supporting Fee On Transfer	Logical Issue	Medium	● Resolved
TCV-01	`vXVS` Case Not Handled By `XVSVaultConverter`	Logical Issue	Medium	● Resolved
VPB-13	Difference Between `amountDiff` And Actual Sum Of Amounts Taken From Pools	Logical Issue	Medium	● Resolved
ATC-04	Ensure `convertConfigurations` Entries Do Not Create A Cycle	Logical Issue	Minor	● Resolved
RFC-02	Insufficient Check To `isComptroller()`	Logical Issue	Minor	● Resolved
RFT-01	AssetsReserves Is Updated After Transfer Of Tokens	Concurrency	Minor	● Partially Resolved
RFV-05	Issue If Funds Are Swept During An Auction	Logical Issue	Minor	● Acknowledged
VPB-10	Missing Or Insufficient Checks	Logical Issue	Minor	● Resolved

ID	Title	Category	Severity	Status
VPU-01	Divide Before Multiply	Incorrect Calculation	Minor	● Resolved
ATC-05	Slots Of Storage Placeholder In `AbstractTokenConverter`	Coding Style	Informational	● Resolved
ATT-01	Potential Issues With Added Checks	Inconsistency	Informational	● Resolved
RFC-04	Function `setPoolsAssetsDirectTransfer()` Should Be Called Atomically With Initialization	Inconsistency	Informational	● Resolved
RFV-06	Issue With `PoolStateUpdated` Event	Inconsistency	Informational	● Resolved
RFV-07	`RiskFundV2` Does Not Implement A `reInitializer()`	Logical Issue	Informational	● Acknowledged
TCV-02	Inefficient Conversion When Asset Is `vXVS` Or `vUSDC`	Design Issue	Informational	● Acknowledged
VPB-01	Upgrade Planning	Design Issue	Informational	● Resolved
VPB-03	Potential Reentrancy	Concurrency	Informational	● Resolved
VPB-04	Inconsistent Event Structure When Setting Addresses	Inconsistency	Informational	● Resolved
VPB-05	Incomplete NatSpec Comments	Inconsistency	Informational	● Resolved
VPB-11	Potential Out-Of-Gas Exception	Logical Issue	Informational	● Acknowledged
VPB-12	Typos And Inconsistencies	Inconsistency	Informational	● Resolved
VPB-14	Inaccurate Naming Of `postSweepToken()`	Coding Style	Informational	● Resolved

RFV-03 | RISKFUNDV2 MISSING OR IMPROPER IMPLEMENTATION

Category	Severity	Location	Status
Logical Issue	● Major	ProtocolReserve/RiskFundV2.sol (base): 21	● Resolved

Description

The interface of the `RiskFund` that will be upgraded is:

```
interface IRiskFund {
    function swapPoolsAssets(
        address[] calldata markets,
        uint256[] calldata amountsOutMin,
        address[][] calldata paths,
        uint256 deadline
    ) external returns (uint256);

    function transferReserveForAuction(address comptroller, uint256 amount) external
returns (uint256);

    function updateAssetsState(address comptroller, address asset) external;

    function convertibleBaseAsset() external view returns (address);

    function getPoolsBaseAssetReserves(address comptroller) external view returns
(uint256);
}
```

However, not all of the functionality used in other contracts are implemented in `RiskFundV2`. In particular, the following functions are used in the contract `Shortfall`, but are not properly implemented by `RiskFundV2`:

- `getPoolsBaseAssetReserves()` is not implemented. This is called in the `Shortfall` contract in the function `_startAuction()`, which will make it impossible to start or restart and auction without upgrading.
- `transferReserveForAuction()` is implemented, but with a different amount of inputs. In particular, there is an added `bidder` input. This is called in the `Shortfall` contract in the function `closeAuction()`, which will make it impossible to close an auction without upgrading. In addition, the shortfall contract has logic to handle if the transfer to the bidder fails and instead of reverting tracking the debt to avoid potential DOS attacks. As such the funds should first be transferred to the shortfall contract and not sent directly to the bidder.

Scenario

Scenario 1:

Assume that the currently deployed shortfall contract (whose implementation contract is 0x916e607af3250ecb2fd4ea82a37eb2756a20e1fc) is used while the risk fund is upgraded to `RiskFundV2` .

`startAuction()` is called, which will call `riskFund.getPoolsBaseAssetReserves(comptroller)` . Since `getPoolsBaseAssetReserves()` is not implemented it will cause a revert. This will make it impossible to start or restart an auction without upgrading.

Scenario2 :

Assume that the currently deployed shortfall contract (whose implementation contract is 0x916e607af3250ecb2fd4ea82a37eb2756a20e1fc) is used while the risk fund is upgraded to `RiskFundV2` and that `getPoolsBaseAssetReserves()` is implemented so that auctions can be started. Furthermore assume an auction is in a state that is ready to be closed.

`closeAuction()` is called, which will eventually call `riskFund.transferReserveForAuction(comptroller, riskFundBidAmount);` . This will revert as the implementation of `RiskFundV2` takes 3 inputs as opposed to 2. This will make it impossible to close the auction without upgrading, causing the last bidders funds to be held and also allow more bids to be placed.

Recommendation

We recommend ensuring that `RiskFundV2` properly implements functions of the original `RiskFund` that are used by other contracts.

Alleviation

`[CertiK, 10/24/2023]` : The client made changes resolving the finding in commit: cf5e3713edb5c4a4f574881cb9f82d375625237e.

VPB-07 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	● **Major**	**ProtocolReserve/RiskFundV2.sol (base): 21~22; ProtocolReserve/XVSVaultTreasury.sol (base): 15~16; TokenConverter/AbstractTokenConverter.sol (base): 18; TokenConverter/RiskFundConverter.sol (base): 20~21; TokenConverter/XVSVaultConverter.sol (base): 14~15**	● **Mitigated**

Description

Contracts `XVSVaultConverter` , `RiskFundConverter` , `XVSVaultTreasury` , and `RiskFundV2` 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. In particular, they can steal all tokens held by the contract.

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.

Alleviation

`[Venus, 10/23/2023]` : 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 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.

VPB-08 | CENTRALIZATION RELATED RISKS

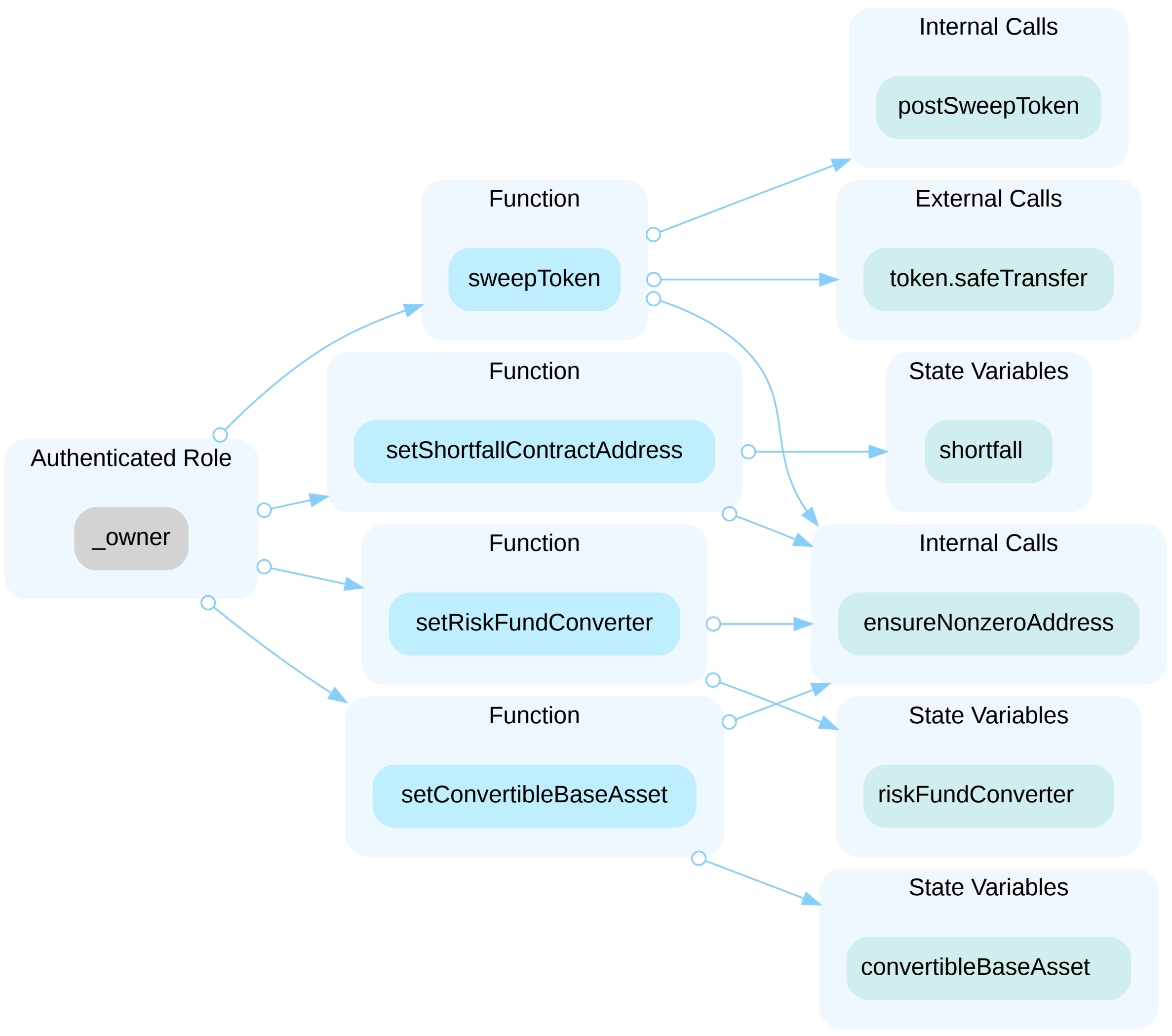
Category	Severity	Location	Status
Centralization	● **Major**	**ProtocolReserve/RiskFundV2.sol (base): 58, 68, 78, 123; ProtocolReserve/XVSVaultTreasury.sol (base): 61, 71; TokenConverter/AbstractTokenConverter.sol (base): 114~115, 125~126, 138, 145, 157~158, 351; TokenConverter/RiskFundConverter.sol (base): 115, 128~129**	● **Mitigated**

Description

RiskFundV2

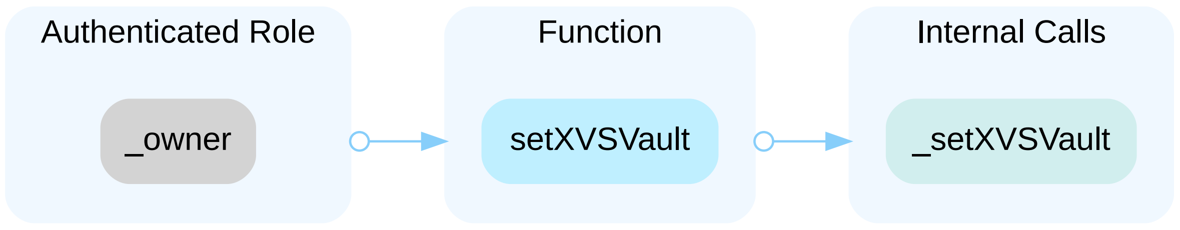
In the contract `RiskFundV2` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and do the following:

- Sweep any token, including those that are allocated to reserves. This can be used to drain the contract of all funds.
- Set the shortfall contract address, which is allowed to transfer reserves. This can be used to drain the contract of all reserves.
- Set the riskFundConverter, which is allowed to increase each pools reserve values.
- Set the convertibleBaseAsset, which is the token that is auctioned off in the shortfall contract;



XVSVaultTreasury

In the contract `XVSVaultTreasury` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and change the `xvsVault`, which is called to get the `xvsStore`. The `xvsStore` address is where the function `fundXVSVault` sends `XVS` to, so an attacker can use this to have XVS sent to a contract they control as opposed to the `xvsStore`.

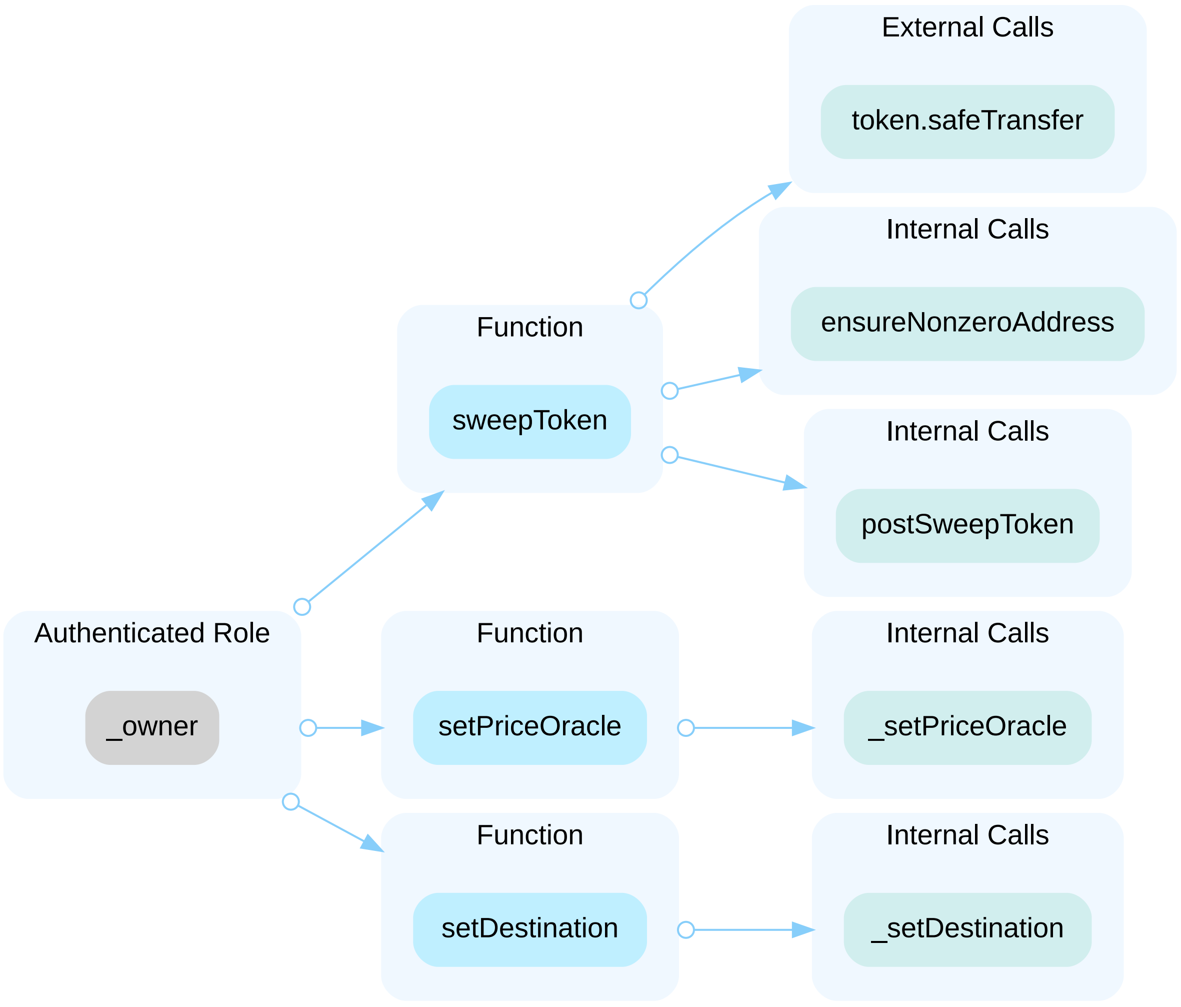


In the contract `XVSVaultTreasury`, the role `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` can grant addresses the privilege to call the function `fundXVSVault()`. Any compromise to the `DEFAULT_ADMIN_ROLE` or accounts granted this privilege may allow a hacker to take advantage of this authority and transfer funds from the `XVSVaultTreasury` to the `xvsStore` .

AbstractTokenConverter

In the contract `AbstractTokenConverter` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and

- remove any token from the contract, including any market assets
- change the oracle to a malicious contract, in order to convert tokens for an inflated price
- change the destination address to an account they control



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

- `pauseConversion()`
- `resumeConversion()`
- `setConversionConfig()`

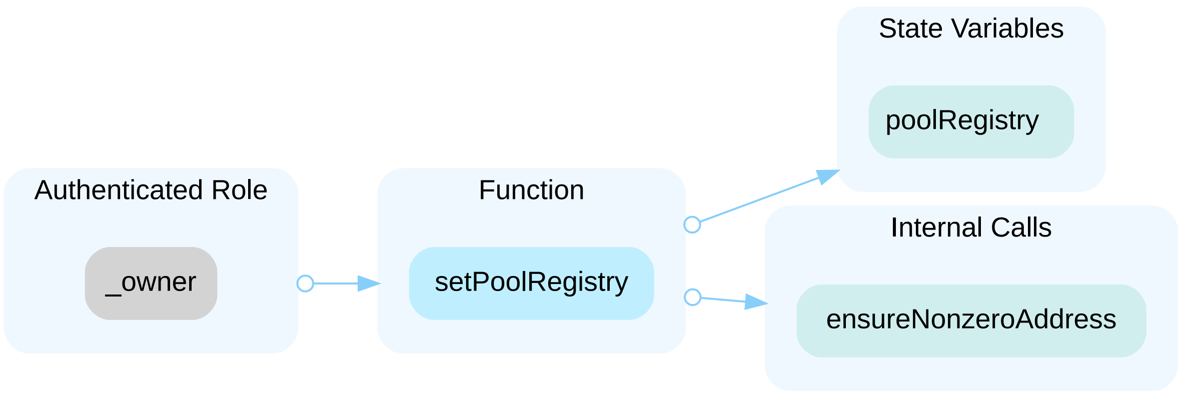
Any compromise to the account may allow the hacker to take advantage of this authority and

- allow the resumption of token conversion while the contract is vulnerable
- pause conversion to prevent swapping during a critical period
- allow a conversion configuration that causes damage to the protocol, such as allowing the same token as a `tokenIn` in a pair and a `tokenOut` in a pair, or using a high percentage for the incentive (the cap is 50% incentive).

Note: Contracts `XVSVaultConverter` and `RiskFundConverter` both inherit `AbstractTokenConverter`, and consequently inherit its centralization risk.

RiskFundConverter

In the contract `RiskFundConverter` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and change the `poolRegistry` to a contract which verifies fake comptrollers and assets are part of the Venus protocol. The fake contracts could be used to steal funds from users or make unexpected external calls in the function logic.



In `RiskFundConverter`, the role `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` can grant addresses the privilege to call the function `setPoolsAssetsDirectTransfer()`. Any compromise to the account may allow a hacker to set an asset from a certain comptroller to be sent directly to the `destinationAddress` without being recorded in the mappings `poolsAssetsReserves` and `assetsReserves`.

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, 10/23/2023]` : The owner of the contracts RiskFundV2, XVSVaultTreasury, RiskFundConverter and XVSVaultConverter 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 Venus 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 Timelock) 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:

XVSVaultTreasury:

- fundXVSVault

RiskFundConverter:

- setPoolsAssetsDirectTransfer
- pauseConversion
- resumeConversion
- setConversionConfig

XVSVaultConverter:

- pauseConversion
- resumeConversion
- setConversionConfig

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

ATC-03 | ISSUES WITH FUNCTIONS SUPPORTING FEE ON TRANSFER

Category	Severity	Location	Status
Logical Issue	● Medium	TokenConverter/AbstractTokenConverter.sol (base): 286~310	● Resolved

Description

The functions supporting fee on transfer tokens do not perform any checks to ensure that the protocol does not lose an unexpected amount in a conversion. In particular, the amount converted is always based on the input amount and does not take into account the amount the destination address receives.

Scenario

Assume that a malicious token, call it tokenA, is supported and is configured to enable conversion with tokenB. In particular assume that `convertConfigurations[tokenAddressIn][tokenAddressOut].incentive = 0.1e18` . As soon as this is done, the token upgrades its implementation so that whenever a transfer to the token converter is made, it transfers 0 tokens.

Assume for simplicity that both the price of tokenA = 1 USD and price of tokenB = 1 USD.

- `convertExactTokensSupportingFeeOnTransferTokens(1e18, 0, tokenA, tokenB, exploiter)` is called.
- Consequently 0 tokenA is transferred from the user due to the malicious upgrade.
- Now `conversionWithIncentive = 1.1e18` so that `tokenInToOutConversion = 1.1e18` .
- Then `amountOutMantissa = 1.1e18` and `amountConvertedMantissa = 1e18` .
- Then the `actualAmountIn = 0` and `actualAmountOut = 1.1e18` .
- Thus the exploiter will receive `1.1e18` tokenB and the `destinationAddress` will receive 0 tokenA.

Recommendation

We recommend basing the converted amount on the actual amount of tokens received by the destination address.

Alleviation

`[CertiK, 10/25/2023]` : The client made the recommended changes in commits:

- a5caf65aac3a12962c7f363c0a8c1af342abd9a0;
- 7f289d05c6993f8c24dea0fbeba536df33d428fd;
- df13780553242e7e106b810403d52198746eddc3.

TCV-01 | `vXVS` CASE NOT HANDLED BY `XVSVaultConverter`

Category	Severity	Location	Status
Logical Issue	● Medium	TokenConverter/AbstractTokenConverter.sol (base): 157~159; TokenConverter/XVSVaultConverter.sol (base): 43~44	● Resolved

Description

Market `vXVS` is not fully handled by the `XVSVaultConverter`. Market `vXVS` has underlying asset `XVS` and is a market that is expected to be distributed by `ProtocolShareReserve` to `XVSVaultConverter` like all other markets. However, it is expected that asset `XVS` would not be a `tokenOut` option in the `XVSVaultConverter`, since it is the token to be accumulated and sent to the `XVSVaultTreasury`, so providing it as an asset to be converted at an incentive would be counter to the purpose of the contract.

The only way `XVS` can be sent from the `XVSVaultConverter` to the `XVSVaultTreasury` is through the token converting functions that send the `tokenInAddress` token to the `destinationAddress`. Without any additional functionality, it appears the only method for retrieving sent `XVS` tokens from the `vXVS` market is through function `sweepToken()`.

Recommendation

We recommend handling the cases where the `vXVS` market sends its assets to the `XVSVaultConverter` contract.

Alleviation

`[CertiK, 10/24/2023]`: The client made changes resolving the finding in commits:

- 9f579b1ba0498ba74bc9e0acb007d22d8fbf9ca5;
- dc70688315616cba35501068d08ae8698994c904.

VPB-13 | DIFFERENCE BETWEEN `amountDiff` AND ACTUAL SUM OF AMOUNTS TAKEN FROM POOLS

Category	Severity	Location	Status
Logical Issue	● Medium	ProtocolReserve/RiskFundV2.sol (base): 184; TokenConverter/RiskFundConverter.sol (base): 273~278	● Resolved

Description

The amount that should be taken from each pool to account for the `amountDiff` is given by determining the percentage of the pools assets against the total reserves of the asset and taking that percentage of the amount.

However, as these operations round down the sum of all the amounts may be slightly less than the amountDiff. Causing the reserves balance to be slightly higher than the balance of the contract.

This can cause an issue when `transferReserveForAuction()` is called as in certain scenarios an auction cannot be closed when it should be. This can cause a delay in the closing of an auction and allow for more bids to be placed.

A similar but less severe discrepancy occurs in `RiskFundConverter`.

Scenario

Assume there are only two pools poolA and poolB. Further assume that `poolAssetsFunds[poolA][convertibleBaseAsset] = 1e18+1`, `poolAssetsFunds[poolB][convertibleBaseAsset] = 1e18`, and the contract balance of `convertibleBaseAsset` is `3e18`.

- `sweepToken(convertiblebaseAsset, to, 1e18)` is called
- In the logic of `postSweepToken()` we then see that `balanceDiff = 3e18 - 2e18 - 1 = 1e18 - 1` and `balanceDiff < amount`
- Then `amountDiff = 1`.
- Then the poolShare for poolA is `((1e18+1)*1e18)/(2e18+1) = 500000000000000000.2499...`, which will round down to `500000000000000000`.
- The poolShare for poolB is `499999999999999999`.
- Thus the `poolAmountShare` for poolA will be `(500000000000000000 * 1)/1e18 = .5` which will round down to 0.
- Similarly the `poolAmountShare` for poolB will round down to 0.
- Thus the poolAssetsFunds is not reduced for either of them and the balance after this is `2e18`.
- Assume an auction was started for `poolB` and is then closed so that `transferReserveForAuction(poolB, bidder, 1e18)` is called and will succeed leaving only `1e18` in the contract.
- Furthermore assume no more reserves are allocated, an auction is started for `poolA`, and the auction is attempted to be closed so that `transferReserveForAuction(poolA, bidder, 1e18+1)` is called to collect the total amount

stored by poolAssetsFunds[poolA][convertibleBaseAsset]. This will attempt to transfer `1e18+1` tokens when it only holds `1e18` causing it to revert.

This scenario demonstrates a case where the inconsistency of the sum of the amounts subtracted from the reserves and the amount of tokens allocated to reserves that are transferred can cause a revert when closing an auction.

Recommendation

We recommend ensuring that the amount of tokens allocated to reserves that are transferred is equal to the sum of the amounts that are subtracted from the individual pool reserves.

Alleviation

`[CertiK, 10/24/2023]` : The client made changes resolving the finding in commit: b8e3e7b5069103c005a7215598cd0ec04f306a51.

ATC-04 | ENSURE `convertConfigurations` ENTRIES DO NOT CREATE A CYCLE

Category	Severity	Location	Status
Logical Issue	● Minor	TokenConverter/AbstractTokenConverter.sol (base): 157~158	● Resolved

Description

In general, it should be ensured before update that no combination of `convertConfigurations` entries creates a cycle where a user can input a token A and get token A as output through a sequence of conversions, where token conversion has been incentivized, causing the user to get more token A as output than they originally input.

It is assumed, for instance, that any token configured as a `tokenInAddress` within the contract will never be set as a `tokenOutAddress` , and further, that only the base asset relevant to the Token Converter contract instance will be used as the `tokenInAddress` . However, we recommend including logic ensuring this is the case when configuring each Token Converter contract instance.

Recommendation

We recommend adding in logic into each Token Converter contract instance which prevents cycles of token conversion from being formed directly within the contract.

Alleviation

`[CertiK]` : The team made changes resolving the finding in commits e37341ce91123454f7a0faecb81569d9141a5882 and 1c558e84934784f331940ffbc27f9efa294fe432.

RFC-02 | INSUFFICIENT CHECK TO `isComptroller()`

Category	Severity	Location	Status
Logical Issue	● Minor	TokenConverter/RiskFundConverter.sol (base): 161~162, 162~163, 173~174, 174~175	● Resolved

Description

Functions `getPoolAssetReserve()` and `updateAssetsState()` both make a check that `isComptroller()` returns true when called on the provided `comptroller` input address. However, this check only ensures that the user-provided input is a contract containing this function and that it returns the expected boolean. This check does not ensure that the address provided is a comptroller that is a legitimate part of protocol, or that the same expected logic is in the contract.

Similarly, both functions check that the input `asset` is a nonzero address. However, this does not ensure that address provided is for a market existing within any of the Venus protocol pools.

For function `updateAssetsState()` , both checks are unnecessary since the check to function `ensureAssetListed()` for the provided `comptroller` and `asset` is sufficient to ensure that the combination is either part of the core pool or isolated pools.

For function `getPoolAssetReserve()` , since the function is for external viewing purposes only, it should be documented that if the `comptroller` and `asset` values provided are not a valid combination within the protocol, the return value will be zero. Alternatively, the check to `ensureAssetListed()` can be used as a replacement in the view function.

Recommendation

We recommend relying on function `ensureAssetListed()` to verify that the combination of `comptroller` and `asset` addresses provided are for a market existing in a legitimate pool of the protocol.

Alleviation

`[CertiK, 10/24/2023]` : The client made the recommended changes in commit: 06914bd575affe5ce719797d6b53590ad44651db.

RFT-01 | ASSETSRESERVES IS UPDATED AFTER TRANSFER OF TOKENS

Category	Severity	Location	Status
Concurrency	● Minor	TokenConverter/RiskFundConverter.sol (update1): 193~194	● Partially Resolved

Description

Note: This finding concerns the topic of read-only reentrancy. The finding is set as a severity of minor because the Venus protocol does not currently support ERC20 tokens with hooks, and native BNB is not used within the scope of the audited files. The finding is made with consideration to future support of tokens with hooks.

Function `RiskFundConverter` has a function `balanceOf()` which reads the mapping `assetsReserves` for a given `tokenAddress`.

Please state all intended uses for this function.

The read of `assetsReserves` is being considered under the context of read-only reentrancy, when tokens with hooks are converted: during token conversion logic, the `assetsReserves` mapping is updated in the `postConversionHook()` logic of this contract, after transfer of tokens to an outgoing destination. If the `balanceOf()` function, the `assetsReserves` mapping, or the `poolAssetsReserves` mapping is read by any other part of the protocol, or by a third party, then this value may not accurately represent amounts within the contract in all instances.

Additionally, please specify what all the current up-to-date intended recipients are for the release of funds through the `ProtocolShareReserve` contract. This contract has a similar issue to that outlined above in that `assetsReserves` and `totalAssetsReserves` are only updated after a transfer of funds. In the event of transferring tokens with a hook, if the destination contracts interact with external accounts, it may be possible to read the state of the `ProtocolShareReserve` before it is consistently updated. If any other part of the protocol relies on reading this information, it is possible that could be manipulated in the future.

Recommendation

We recommend following check-effect-interact wherever possible to mitigate some of the concern regarding read-only reentrancy. The issue primarily concerns tokens that are being transferred to outgoing destinations not controlled by the protocol.

Additionally, please provide the requested information outlined above in order to determine whether further precautions should be taken, if tokens with hooks are supported in the future.

Alleviation

`[CertiK, 11/06/2023]` : The team states that the main intended purpose of `balanceOf()` read is for use in `AbstractTokenConverter` in order to record the balance of each token in `RiskFundConverter` , while ignoring donations that potentially alter accounting.

Further, see below for their response regarding the current and future recipients of funds by the `ProtocolShareReserve` .

`[Venus, 11/06/2023]` : "The recipients of the funds in the ProtocolShareReserve (PSR) contract are:

- VTreasury (https://github.com/VenusProtocol/venus-protocol/blob/develop/contracts/Governance/VTreasury.sol)
- RiskFundConverter
- SingleTokenConverter (several instances)

Right now, the PSR contract is sending funds also to the RiskFund contract (https://github.com/VenusProtocol/isolated-pools/blob/develop/contracts/RiskFund/RiskFund.sol), but that will change as soon as the RiskFundConverter is deployed.

So, every recipient is under control by the Venus protocol, and they can be changed only by the community with a VIP (Venus Improvement Proposal)"

The recipient contract of the funds in the RiskFundConverter will be the RiskFundV2 contract, which doesn't read the state of the RiskFundConverter when it receives the funds. So, we consider is safe to keep the code as it is.

RFV-05 | ISSUE IF FUNDS ARE SWEPT DURING AN AUCTION

Category	Severity	Location	Status
Logical Issue	● Minor	ProtocolReserve/RiskFundV2.sol (base): 202	● Acknowledged

Description

Currently `convertibleBaseAsset` reserves can also be swept when `sweepToken()` is called. This allows for a scenario where reserves are being auctioned, but while the auction is ongoing and before it is closed they are swept. If this is done, then the auction will not be able to be closed as there will not be enough reserves to transfer.

Scenario

Assume there is a single pool, poolA, and that `poolAssetsFunds[poolA][convertibleBaseAsset] = 100` and that the balance of the convertible base asset of the risk fund is also 100.

- An auction for the entire risk fund of poolA is started.
- While the auction is ongoing and before `closeAuction()` is called, `sweepToken(convertibleBaseAsset, to, 100)` is called.
- This will transfer 100 convertibleBaseAsset from the risk fund to the `to` address. In the process it will also update `poolAssetsFunds[poolA][convertibleBaseAsset] = 0`.
- Enough time passes between bids so that `closeAuction()` can be called. However, this will call `transferReserveForAuction()`, which will revert when the following check is made:

```
if (amount > poolReserve) {
            revert InsufficientPoolReserve(comptroller, amount, poolReserve);
        }
```

Recommendation

We recommend ensuring that tokens that are currently being auctioned cannot be swept.

Alleviation

`[Venus, 10/23/2023]` : Issue acknowledged. I won't make any changes for the current version.

Sweep tokens would be an emergency action, with a higher priority than an auction. So, a reverted auction is acceptable in this scenario. The side effects of the auction could be resolved later by Governance.

VPB-10 | MISSING OR INSUFFICIENT CHECKS

Category	Severity	Location	Status
Logical Issue	● Minor	ProtocolReserve/RiskFundV2.sol (base): 182; ProtocolReserve/XVSVaultTreasury.sol (base): 70~71; TokenConverter/AbstractTokenConverter.sol (base): 203~204, 248~249, 291~292, 325~326; TokenConverter/RiskFundConverter.sol (base): 237~238, 246, 260~261, 274~275	● Resolved

Description

Missing or insufficient validation checks for function parameters can result in unexpected behavior and security vulnerabilities.

RiskFundConverter

- The function `postSweepToken()` is missing a check that the `amount` used as input is nonzero. Use of a zero amount may result in an ambiguous error message, since in that case, the `assetsReserves` value is not prevented from being nonzero, and can cause a division by zero.
- The function `postConversionHook()` is missing a check that the `assetsReserves` value for the `tokenOutAddress` is nonzero. This may result in an ambiguous error message due to a division by zero.

XVSVaultTreasury

- The function `fundXVSVault()` is missing a check that `amountMantissa` is a nonzero value.

RiskFundV2

- The function `postSweepToken()` is missing a check that the `amount` used as input is nonzero. Use of a zero `amount` may result in an ambiguous error message, since in that case, the `assetsReserves` value is not prevented from being nonzero, and can cause a division by zero.

AbstractTokenConverter

- The conversion functions are missing a check that `to` is neither the `tokenInAddress` nor `tokenOutAddress`.

Recommendation

We recommend adding the missing checks listed above.

Alleviation

`[CertiK, 10/24/2023]` : The client made the recommended changes in commits:

- df2eeb4c782abfebd92aae1334734d646a5e5c21;
- f76345d1780e82e91ae138a3cdbdf9b38fbfa3ff.

VPU-01 | DIVIDE BEFORE MULTIPLY

Category	Severity	Location	Status
Incorrect Calculation	● Minor	ProtocolReserve/RiskFundV2.sol (update1): 203, 228; ProtocolReserve/RiskFundV2.sol (base): 182, 201; TokenConverter/AbstractTokenConverter.sol (base): 433, 435; TokenConverter/RiskFundConverter.sol (base): 246, 249	● Resolved

Description

Performing integer division before multiplication truncates the low bits, losing the precision of calculation.

```
433         uint256 tokenInToOutConversion = (tokenInUnderlyingPrice *
conversionWithIncentive) / tokenOutUnderlyingPrice;
```

```
435         amountOutMantissa = (amountInMantissa * tokenInToOutConversion) /
EXP_SCALE;
```

```
246             uint256 poolShare = (poolsAssetsReserves[pools[i]][tokenOutAddress]
* EXP_SCALE) / assetReserve;
```

```
249             uint256 poolAmountInShare = (poolShare * amountIn) / EXP_SCALE;
```

```
182             uint256 poolShare = (poolAssetsFunds[pools[i]][tokenAddress] *
EXP_SCALE) / assetReserves;
```

```
201             uint256 poolAmountShare = (poolShare * amount) / EXP_SCALE;
```

Recommendation

We recommend applying multiplication before division to avoid loss of precision.

Alleviation

`[CertiK, 10/24/2023]` : The client made changes resolving this finding in commits:

- 9145119fac5cf60b08e1cc5abbe364167b485458;
- 7f289d05c6993f8c24dea0fbeba536df33d428fd;

- d0c5d17d66f27b2ac2e1c200ce9165568cb3fb1c;
- 3e600898ea8522c50f1b1361e378866889de9f41.

ATC-05 | SLOTS OF STORAGE PLACEHOLDER IN `AbstractTokenConverter`

Category	Severity	Location	Status
Coding Style	● Informational	TokenConverter/AbstractTokenConverter.sol (base): 25~35, 40~41	● Resolved

Description

The storage placeholder `__gap` variable in `AbstractTokenConverter` , along with the variables that are currently defined, take up a total of 51 storage slots.

Recommendation

If the intention is to take up the customary number of 50 storage slots, the `__gap` variable can instead be made a `uint256[46]` type variable.

Alleviation

`[CertiK, 10/24/2023]` : The client made changes resolving the finding in commit: a71fe6fda9d41d7a3f4dbd430947bd5a6633e248.

ATT-01 | POTENTIAL ISSUES WITH ADDED CHECKS

Category	Severity	Location	Status
Inconsistency	● Informational	TokenConverter/AbstractTokenConverter.sol (update1): 176~182	● Resolved

Description

The following checks were added to the `setConversionConfig()` function:

```
        if (
            (tokenAddressIn == tokenAddressOut) ||
            (tokenAddressIn != _getDestinationBaseAsset()) ||
            conversionConfigurations[tokenAddressOut][tokenAddressIn].enabled
        ) {
            revert InvalidTokenConfigAddresses();
        }
```

1. First check ensures that `tokenAddressIn != tokenAddressOut`.
2. Second check ensures that `tokenAddressIn == _getDestinationBaseAsset()`.
3. Third check ensures that the reverse mapping is not enabled.

However, there are scenarios where the third check can cause potential issues if the destination base asset is changed. Currently the only converter that allows for such a scenario is the `RiskFundConverter`. Please share the procedure that will be followed when there will be a change in the `convertibleBaseAsset` and updates to conversion configs in the `RiskFundConverter`.

Scenario

Assume that `conversionConfigurations[tokenA][tokenB]` is enabled in `RiskFundConverter`, so that necessarily `tokenA` is currently the convertible base asset. Further assume that the convertible base asset is changed to be `tokenB`.

- As `tokenB` is the new convertible base asset it may be desired to convert `tokenA`.
- However, `conversionConfigurations[tokenA][tokenB]` is enabled, so the third check will prevent the conversion.
- In addition, `conversionConfigurations[tokenA][tokenB]` cannot be disabled as the second check will prevent it.

In order to resolve this, the convertible base asset would need to be changed back and all configurations for it disabled.

Recommendation

We recommend either:

1. adjusting the logic to prevent any issues with conversion configurations when the convertible base asset of the `RiskFund` is changed, or
2. ensuring manually that all conversion configurations for the contract are disabled before updating the convertible base asset of the `RiskFund` contract.

Alleviation

`[Venus, 11/03/2023]` : "This function will be invoked in a VIP (Venus Improvement Proposal) by Governance, so there will be opportunities to check (via simulation) whether the change will work (or not). If the transaction is reverted we'll have to disable first the previously enabled conversion, then change the base asset, and finally set up the new conversions (everything in the same VIP)"

RFC-04 | FUNCTION `setPoolsAssetsDirectTransfer()` SHOULD BE CALLED ATOMICALLY WITH INITIALIZATION

Category	Severity	Location	Status
Inconsistency	● Informational	TokenConverter/RiskFundConverter.sol (base): 128~129	● Resolved

Description

If function `setPoolsAssetsDirectTransfer()` is not called atomically with the `initialize()` function during initialization in order to directly transfer the base convertible asset to the `RiskFundV2` contract, then when `updateAssetsState()` is called, the value will be recorded in `poolsAssetsReserves[comptroller][asset]` for the `asset`. Consequently, since it is assumed that the convertible base asset will not be valid as a `tokenOut` address in a conversion, the amount recorded in the `poolsAssetsReserves` can only be removed through sweeping the token from the contract.

While this may be the intended purpose of function `sweepToken()`, the action could be avoided for the initial convertible base asset by calling these functions atomically.

Recommendation

We recommend calling `setPoolsAssetsDirectTransfer()` atomically with initialization.

Alleviation

`[CertiK, 10/24/2023]` : The client made the recommended changes in commit: 83e5e1b2dc0e3ee445c410a7e3ae0588b835b6f1.

RFV-06 | ISSUE WITH `PoolStateUpdated` EVENT

Category	Severity	Location	Status
Inconsistency	● Informational	ProtocolReserve/RiskFundV2.sol (base): 36~37, 154, 203	● Resolved

Description

The event `PoolStateUpdated` is emitted in the functions `updatePoolAssetsReserve()` and `updatePoolState()`. In the function `updatePoolState()` it indicates that the pools assets were incremented by the amount, however, in the function `updatePoolAssetsReserve()` it indicates that the pools assets were decremented by the amount. In addition, the comment above the event states "Emitted when pool states is updated with amount transferred to this contract" however it is also emitted when an amount of the reserve is swept.

Recommendation

We recommend adding a new event for when pool reserves are swept and emitting this in `updatePoolAssetsReserve()` as opposed to the `PoolStateUpdated()` event.

Alleviation

`[CertiK, 10/24/2023]`: The client made the recommended changes in commit: 42e67b7189c3f243a3ec8667bd08761d53a4eebe.

RFV-07 | `RiskFundV2` DOES NOT IMPLEMENT A `reInitializer()`

Category	Severity	Location	Status
Logical Issue	● Informational	ProtocolReserve/RiskFundV2.sol (base): 22	● Acknowledged

Description

The currently deployed `RiskFund` is to be upgraded to `RiskFundV2`. As such a `reinitializer()` can be used to reinitialize the proxy according to the updated logic. In particular, this can be used to initialize the new `riskFundConverter` address as well as update deprecated states to their default.

Recommendation

We recommend implementing a `reinitializer()`.

Alleviation

`[Venus, 10/13/2023]`: Issue acknowledged. I won't make any changes for the current version. We prefer to set the new needed values using the setter functions in the VIP, and not update the deprecated states to their default values, just in case we need those values in the short term

TCV-02 | INEFFICIENT CONVERSION WHEN ASSET IS `vXVS` OR `vUSDC`

Category	Severity	Location	Status
Design Issue	● Informational	TokenConverter/AbstractTokenConverter.sol (base): 157~159; TokenConverter/RiskFundConverter.sol (base): 186~187; TokenConverter/XVSVaultConverter.sol (base): 43~44	● Acknowledged

Description

Markets `vXVS` and `vUSDC` are handled inefficiently in `RiskFundConverter` and `XVSVaultConverter` respectively.

- It is assumed that `XVS` is not directly transferred as an asset from the `RiskFundConverter` to the `RiskFundV2` contract since this is not in line with the specification.
- Additionally, since `XVSVaultConverter` has no method of direct transfer, it is assumed that `USDC` is converted for `XVS` within this contract, since `vUSDC` is a market, and since the specification states `XVS` is to be collected by `XVSVaultConverter` .
- Consequently, it is assumed that `XVS` will be offered as an incentivized asset in `RiskFundConverter` , to be converted in exchange for a user providing `USDC` .
- This allows a user to convert `USDC` to `XVS` via the `RiskFundConverter` and then take the received `XVS` and convert it back to USDC via the `XVSVaultConverter` .
- The process can be continued as long as assets are available in each contract, compounding profit by continuously converting between the two Token Converter contracts.

Since the protocol already has the desired asset, the logic could be adjusted to directly send funds to the corresponding destination, rather than relying on users to convert.

Recommendation

We recommend explicitly handling the cases where the `vUSDC` and `vXVS` markets send their assets to each token converter contract to prevent unnecessary losses from tokens that can be converted with the protocols own funds.

Alleviation

`[Venus, 10/25/2023]` : Issue acknowledged. I will fix the issue in the future, which will not be included in this audit engagement.

VPB-01 | UPGRADE PLANNING

Category	Severity	Location	Status
Design Issue	● Informational	ProtocolReserve/XVSVaultTreasury.sol (base): 28~29, 51; TokenConverter/RiskFundConverter.sol (base): 52~53; TokenConverter/XVSVaultConverter.sol (base): 19~20	● Resolved

Description

The contracts `XVSVaultConverter` , `RiskFundConverter` , and `XVSVaultTreasury` each include a storage placeholder variable, `__gap` , in the state of the child contract.

Additionally, contract `XVSVaultTreasury` includes the `virtual` keyword.

Please provide information on whether there is intention to ever use the child contracts listed as parent inheritances instead.

If not, the placeholder variable in the child contracts and the `virtual` keyword may not be necessary.

Recommendation

We recommend providing information on whether there is intention to ever use the child contracts listed as parent inheritances instead.

If not, the placeholder variable in the child contracts and the `virtual` keyword may not be necessary.

Alleviation

`[CertiK, 10/24/2023]` : The client made changes resolving the finding in commit: ade9bdaa9e2b9b91bd8a6613a028c197fb60c799.

VPB-03 | POTENTIAL REENTRANCY

Category	Severity	Location	Status
Concurrency	● Informational	ProtocolReserve/RiskFundV2.sol (base): 110, 111; ProtocolReserve/XVSVaultTreasury.sol (base): 81, 83; TokenConverter/RiskFundConverter.sol (base): 188, 191	● Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

This finding is considered informational because the reentrancy only causes out-of-order events and because the team has stated they do not currently support tokens with hooks.

External call(s)

```
110         IERC20Upgradeable(convertibleBaseAsset).safeTransfer(bidder, amount);
```

- This function call executes the following external call(s).
- In `SafeERC20Upgradeable._callOptionalReturn` ,
    - `returndata = address(token).functionCall(data,"SafeERC20: low-level call failed")`
- In `AddressUpgradeable.functionCallWithValue` ,
    - `(success,returndata) = target.call{value: value}(data)`

Events emitted after the call(s)

```
111         emit TransferredReserveForAuction(comptroller, amount);
```

External call(s)

```
81          IERC20Upgradeable(XVS_ADDRESS).safeTransfer(xvsStore, amountMantissa);
```

- This function call executes the following external call(s).
- In `SafeERC20Upgradeable._callOptionalReturn` ,

    - `returndata = address(token).functionCall(data,"SafeERC20: low-level call failed")`

- In `AddressUpgradeable.functionCallWithValue`,
    - `(success,returndata) = target.call{value: value}(data)`

Events emitted after the call(s)

```
        emit FundsTransferredToXVSStore(xvsStore, amountMantissa);
```

External call(s)

```
                token.safeTransfer(destinationAddress, balanceDifference);
```

- This function call executes the following external call(s).
- In `SafeERC20Upgradeable._callOptionalReturn`,
    - `returndata = address(token).functionCall(data,"SafeERC20: low-level call failed")`
- In `AddressUpgradeable.functionCallWithValue`,
    - `(success,returndata) = target.call{value: value}(data)`

Events emitted after the call(s)

```
                emit AssetTransferredToDestination(comptroller, asset,
balanceDifference);
```

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier to all user facing functions to prevent reentrancy.

Alleviation

`[CertiK, 11/03/2023]`: The client made the recommended changes in commits:

- 17a377cea76a320d3cf1ec0552ecdce494203566;
- 7458733937bbead87623c3e348b6bd5c8ebe57b8.

VPB-04 | INCONSISTENT EVENT STRUCTURE WHEN SETTING ADDRESSES

Category	Severity	Location	Status
Inconsistency	● Informational	ProtocolReserve/RiskFundV2.sol (base): 60~61, 70~71, 80~81; ProtocolReserve/XVSVaultTreasury.sol (base): 93~94; TokenConverter/AbstractTokenConverter.sol (base): 172~185, 623~626, 636~639; TokenConverter/RiskFundConverter.sol (base): 117~119	● Resolved

Description

Throughout the codebase, when an address is set the event is emitted prior to setting the new address in order to avoid needing a temporary variable to hold the old address to emit the event. However, in some places the event is emitted last and a temporary variable is used.

Recommendation

We recommend adopting one convention and having it be consistent throughout the codebase.

Alleviation

`[CertiK, 10/24/2023]` : The client made the recommended changes in commit: ecc0c216d455064320b8ae8755591a0668789ff8.

VPB-05 | INCOMPLETE NATSPEC COMMENTS

Category	Severity	Location	Status
Inconsistency	● Informational	ProtocolReserve/RiskFundV2.sol (base): 58, 68, 78, 99~101, 149~151; ProtocolReserve/XVSVaultTreasury.sol (base): 39, 49~50; TokenConverter/AbstractTokenConverter.sol (base): 346~350, 490~491, 642~645, 677; TokenConverter/RiskFundConverter.sol (base): 111~114, 122~127, 205~206, 211~212, 257~259, 313~315; TokenConverter/XVSVaultConverter.sol (base): 45~46	● Resolved

Description

XVSVaultTreasury

- The `constructor()` does not have a NatSpec comment for the `xvsAddress_` parameter.
- The function `initialize()` does not include comments for the `XVSVaultUpdated` event and the `ZeroAddressNotAllowed` error.

RiskFundV2

- The comments above the functions `setConvertibleBaseAsset()`, `setRiskFundConverter()`, `setShortfallContractAddress()` do not mention their access is only governance.
- The comments above the function `transferReserveForAuction()` do not reflect that it is only callable by `shortfall`.
- The comments above the function `updatePoolState()` do not reflect that is is only callable by `riskFundConverter`.

AbstractTokenConverter

- The comments above the function `sweepToken()` do not include the parameters `to` and `amount`.
- The comments above the function `postConversionHook()` do not include the parameter `tokenAddressOut`.
- The comments above the function `balanceOf()` do not include the return parameter `tokenBalance`.
- The comments above the function `_checkConversionPaused()` do not include the error `ConversionTokensPaused`.

RiskFundConverter

- The comments above the function `setPoolRegistry()` do not reflect it is only callable by governance.
- The comments above the function `setPoolsAssetsDirectTransfer()` do not include the parameter `values`.

- The comments above the function `balanceOf()` do not include the return parameter.
- The comments above the function `getPools()` do not include the return parameter.
- The comments above the function `ensureAssetListed()` do not include the return parameter.
- The function `isAssetListedInCore()` is missing comments.
- The comments above the function `postSweepToken()` do not include the error `InsufficientBalance`.

XVSVaultConverter

- The comments above the function `balanceOf()` do not include the return parameter `tokenBalance`.

Recommendation

We recommend adding in the missing NatSpec comments to improve the readability of the codebase.

Alleviation

`[CertiK, 10/24/2023]`: The client made the recommended changes in commit: 2848374ac70572f97af3a3b10b063711d77500cb.

VPB-11 | POTENTIAL OUT-OF-GAS EXCEPTION

Category	Severity	Location	Status
Logical Issue	● Informational	ProtocolReserve/RiskFundV2.sol (base): 169, 181; TokenConverter/RiskFundConverter.sol (base): 220, 245, 273, 302	● Acknowledged

Description

When a loop allows an arbitrary number of iterations or accesses state variables in its body, the function may run out of gas and revert the transaction.

```
169         for (uint256 i; i < poolsLength; ++i) {
```

Function `RiskFundV2.postSweepToken` contains a loop and its loop condition depends on external calls: `IRiskFundConverter(riskFundConverter).getPools` .

```
181             for (uint256 i; i < poolsLength; ++i) {
```

Function `RiskFundV2.postSweepToken` contains a loop and its loop condition depends on external calls: `IRiskFundConverter(riskFundConverter).getPools` .

```
220             for (uint256 i; i < poolsLength; ++i) {
```

Function `RiskFundConverter.getPools` contains a loop and its loop condition depends on external calls: `IPoolRegistry(poolRegistry).getPoolsSupportedByAsset` .

```
245         for (uint256 i; i < pools.length; ++i) {
```

Function `RiskFundConverter.postConversionHook` contains a loop and its loop condition depends on external calls: `IPoolRegistry(poolRegistry).getPoolsSupportedByAsset` .

```
273             for (uint256 i; i < pools.length; ++i) {
```

Function `RiskFundConverter.postSweepToken` contains a loop and its loop condition depends on external calls: `IPoolRegistry(poolRegistry).getPoolsSupportedByAsset` .

```
302         for (uint256 i; i < coreMarkets.length; ++i) {
```

Function `RiskFundConverter.isAssetListedInCore` contains a loop and its loop condition depends on external calls: `IComptroller(CORE_POOL_COMPTROLLER).getAllMarkets`.

Recommendation

We recommend ensuring that the amount of pools and core markets does not cause an out of gas exception.

Alleviation

`[Venus, 10/13/2023]`: Issue acknowledged. I won't make any changes for the current version.

The check should be done in the functions where the pools are added. Specifically in the PoolRegistry.addPool function. That function is executable only by Governance, so there should be enough opportunities (via fork tests in the VIP simulation) to check if adding a new pool could generate an out-of-gas issue.

VPB-12 | TYPOS AND INCONSISTENCIES

Category	Severity	Location	Status
Inconsistency	● Informational	ProtocolReserve/RiskFundV2.sol (base): 57, 86~88, 90, 116, 192; TokenConverter/AbstractTokenConverter.sol (base): 31, 34, 57, 60, 63, 66, 112, 123, 149, 188, 197, 233, 278, 312, 499, 543, 643, 647, 677; TokenConverter/RiskFundConverter.sol (base): 30~31, 46~47	● Resolved

Description

RiskFundV2

- The comments above the function `setConvertibleBaseAsset()` state "ZeroAddressNotAllowed is thrown when risk fund converter address is zero", however, it should reference the convertible base asset as opposed to the risk fund converter.
- The comments above the function `transferReserveForAuction()` state "param bidder Amount transferred to bidder address". However, bidder is an address not an amount.
- The comments above the function `transferReserveForAuction()` state "@param amount Amount to be transferred to auction contract". However, this amount is transferred to the `bidder`.
- The comments above the function `transferReserveForAuction()` state "@return Number reserved tokens.". This is not descriptive and may be misinterpreted.
- The comments above the function `transferReserveForAuction()` state "@custom:error InvalidShortfallAddress is thrown on invalid shortfall address". However, this is thrown when the caller is not the `shortfall` address.
- The comments above the function `sweepToken()` state "Tokens are sent to admin (timelock)", however, they are transferred to the input `to` address.
- The comments above the function `updatePoolAssetsReserve()` state "@param amount Amount transferred to address(to)". However, the input amount is not necessarily the amount that will be transferred to `address(to)`. It is the amount of reserves that should be transferred to `address(to)`.

AbstractTokenConverter

- Throughout the file "convert" is used when "converted" or "conversion" should be used.
- The comment above the variable `conversionPaused` states "Boolean of if convert is paused" which should be "Boolean for if convert is paused".
- The comment above the variable `destinationAddress` states "Address at all incoming tokens are transferred to" which should be "Address that all incoming tokens are transferred to".
- The comment above function `convertExactTokens()` contains typo "fater" which should be the word "after."

- The comments above the functions to convert tokens state that they convert an exact amount. However, they only convert the exact amount if there is enough tokens held by the contract, otherwise the amount is adjusted.
- In the function `postConversionHook()` , `tokenInAddress` is used, where everywhere else in the file `tokenAddressIn` is used.

RiskFundConverter

- The comments above immutable value `VBNB` state that the address is "used to exclude the BNB market while in the `getPools()` method", however, the market is not excluded in this function.
- The comments above mapping `poolsAssetsDirectTransfer` states "this mapping would contain the assets for the pool which would be send to RiskFund directly." The comment can be modified to state "the mapping contains the assets for each pool which are sent to RiskFund directly."

Recommendation

We recommend fixing the typos and inconsistencies mentioned above.

Alleviation

`[CertiK, 10/24/2023]` : The client made the recommended changes in commits:

- 20cd1b15c8e0272b3d7c2d6433a6390ca98c368d;
- dc70688315616cba35501068d08ae8698994c904.

VPB-14 | INACCURATE NAMING OF `postSweepToken()`

Category	Severity	Location	Status
Coding Style	● Informational	ProtocolReserve/RiskFundV2.sol (base): 132~133; TokenConverter/AbstractTokenConverter.sol (base): 360~361	● Resolved

Description

Function `postSweepToken()` is called before token transfers in the `sweepToken()` function. It may be more descriptive to rename the function `preSweepToken()` since the function performs important checks and updates to state before transferring the specified token.

Recommendation

We recommend renaming the function `postSweepToken()` to `preSweepToken()`.

Alleviation

`[CertiK, 10/24/2023]`: The client made the recommended changes in commit: e99134211022180b0ca7c2f4b317ddb03d159487.

OPTIMIZATIONS | VENUS - TOKEN CONVERTER

ID	Title	Category	Severity	Status
VPB-06	`for` Loop Optimization	Gas Optimization	Optimization	● Resolved

VPB-06 | `for` LOOP OPTIMIZATION

Category	Severity	Location	Status
Gas Optimization	● Optimization	ProtocolReserve/RiskFundV2.sol (base): 169, 181; TokenConverter/RiskFundConverter.sol (base): 141, 149, 220, 245, 273, 302	● Resolved

Description

In general, the counter in a for loop can be incremented or decremented in an unchecked block as it cannot overflow or underflow, saving gas as it will not perform a check for overflow or underflow.

Additionally, it saves a small amount of gas to increment an index in a `for` loop from the left instead of from the right side as it performs fewer operations.

Recommendation

We recommend incrementing the index of the for loop in an unchecked block with a prefix increment.

Alleviation

`[CertiK, 11/03/2023]` : The client made the recommended changes in commits

- b04357db49b2fe81194c3df874ffb229de7dbbf1;
- b483870c74a5567af195b4ce214b9770057318ef.

APPENDIX | VENUS - TOKEN CONVERTER

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.
Incorrect Calculation	Incorrect Calculation findings are about issues in numeric computation such as rounding errors, overflows, out-of-bounds and any computation that is not intended.
Concurrency	Concurrency findings are about issues that cause unexpected or unsafe interleaving of code executions.
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.
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.

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.