CERTIK

Security Assessment

Venus - Correlated token oracles

CertiK Assessed on Apr 12th, 2024

CertiK Assessed on Apr 12th, 2024

Venus - Correlated token oracles

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 04/12/2024	N/A

CODEBASE
https://github.com/VenusProtocol/oracle
View All in Codebase Page

COMMITS
Base: 2a2f31117eee109c05c52fb48c08a358313709b1
Update 1: 3e86e3234cf9f61cfb973fccef9fcf1afff2c25a
Update2: eeab989a9c6e2f44555cba629b944fb99835f193
View All in Codebase Page

Highlighted Centralization Risks

Contract upgradeability

Vulnerability Summary

11	8	0	0	3	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.
1	Major	1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
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.
1	Minor	1 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.

8	Informational	6 Resolved, 2 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 - CORRELATED TOKEN ORACLES

CODEBASE | VENUS - CORRELATED TOKEN ORACLES

Repository

https://github.com/VenusProtocol/oracle

Commit

Base: 2a2f31117eee109c05c52fb48c08a358313709b1

Update 1: 3e86e3234cf9f61cfb973fccef9fcf1afff2c25a

Update2: eeab989a9c6e2f44555cba629b944fb99835f193

AUDIT SCOPE | VENUS - CORRELATED TOKEN ORACLES

12 files audited ● 11 files with Acknowledged findings ● 1 file with Resolved findings

ID	Repo	File	SHA256 Checksum
ABN	VenusProtocol/oracle	contracts/oracles/AnkrBNBOracle.sol	b1ab4e8c20d2c979e9da60e8ecb1e2799939c64317ef7f2865eaa4098c633abf
BNB	VenusProtocol/oracle	contracts/oracles/BNBxOracle.sol	2c86c303d882c3e87e0db103981b22fcac71a11246f25e14b13789bf4d9c924d
OJO	VenusProtocol/oracle	contracts/oracles/OneJumpOracle.sol	fd81a0aa8ba4a22b9447d2fd1e26722193028d33cc7e4daddacb7f70130fe309
POV	VenusProtocol/oracle	contracts/oracles/PendleOracle.sol	ad7736222837874aa8042d3e84a9149e81e4aea55e168549f5f169b46b4cb5b0
SFO	VenusProtocol/oracle	contracts/oracles/SFraxOracle.sol	a9f132a462d43c4f51bae2a207349ea302042ba5987d17a623f5da4ae9d95904
SFE	VenusProtocol/oracle	contracts/oracles/SFrxETHOracle.sol	163270848f3e47bda45a98ef861e8179bc5c6106388f621b286270400bd7e335
SBN	VenusProtocol/oracle	contracts/oracles/SlisBNBOracle.sol	69944b339c6c1b2ce9898ad52dbafe5e090094a4ff547e85e9b0465f49dc4be5
SBB	VenusProtocol/oracle	contracts/oracles/StkBNBOracle.sol	a4c0580dd2e889a572652a6fd4be893242a09c05c82d4939dc250141f88c39bb
WBE	VenusProtocol/oracle	contracts/oracles/WBETHOracle.sol	46eee8e8f00202fe8d2253d74b2a6e4c6fad38ed9a99e2c504267e0827668f15
WET	VenusProtocol/oracle	contracts/oracles/WeETHOracle.sol	69d0614179caef40756717e070a0be90460376bc1ecdc795d68de0b283572e7e
WEH	VenusProtocol/oracle	contracts/oracles/WstETHOracle.sol	9c1d00ea8f8fb3e032c571281371f36fb23357e8930adb4cd866e3df80bae083
CTO	VenusProtocol/oracle	contracts/oracles/common/CorrelatedTokenOracle.sol	c0241b765f99076bf32b46a065bfb25cda6fad5c7a11ba643df79e31cd8f3073

APPROACH & METHODS

VENUS - CORRELATED TOKEN ORACLES

This report has been prepared for Venus to discover issues and vulnerabilities in the source code of the Venus - Correlated token oracles 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 - CORRELATED TOKEN ORACLES

This audit concerns the changes made in files outlined in:

- PR-165

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

DEPENDENCIES | VENUS - CORRELATED TOKEN ORACLES

Third Party Dependencies

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

- Third Party Token Contracts
- Third Party Staking Contracts
- Third Party 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 returning invalid prices, returning invalid exchange rates, etc.

Out Of Scope Dependencies

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

- Resilient Oracle

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

Recommendations

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

FINDINGS | VENUS - CORRELATED TOKEN ORACLES

11	0	1	1	1	8
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Venus - Correlated token oracles. Through this audit, we have uncovered 11 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-06	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	**Acknowledged**
POV-01	Missing Validation Of `PendleOracle` Parameters	Logical Issue	Medium	Resolved
SBB-01	Potential Divide By Zero	Logical Issue	Minor	Resolved
OJO-01	Discussion On Use Cases Of `OneJumpOracle`	Logical Issue	Informational	Resolved
SFO-01	Parameter Order Can Be Swapped	Coding Style	Informational	Resolved
SFO-02	Third Party's Rate May Be Manipulated	Logical Issue	Informational	Acknowledged
VPB-01	Discussion On Exchange Rate Choice	Inconsistency	Informational	Resolved
VPB-03	BNB Address Input Can Be Constant	Logical Issue	Informational	Resolved
VPB-04	Inconsistent Input Naming Convention	Inconsistency	Informational	Resolved
VPB-05	Oracles Assume Tokens Have 18 Decimals	Logical Issue	Informational	Acknowledged
VPU-01	Typos And Inconsistencies	Coding Style	Informational	Resolved

VPB-06 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	Major	contracts/oracles/AnkrBNBOracle.sol (Base): 12~13; contracts/oracles/BNBxOracle.sol (Base): 14~15; contracts/oracles/OneJumpOracle.sol (Base): 13~14; contracts/oracles/PendleOracle.sol (Base): 13~14; contracts/oracles/SFraxOracle.sol (Base): 12~13; contracts/oracles/SFrxETHOracle.sol (Base): 12~13; contracts/oracles/SlisBNBOracle.sol (Base): 13~14; contracts/oracles/StkBNBOracle.sol (Base): 15~16; contracts/oracles/WBETHOracle.sol (Base): 12~13; contracts/oracles/WeETHOracle.sol (Base): 12~13; contracts/oracles/WstETHOracle.sol (Base): 12~13	Acknowledged

Description

In each of the contracts cited, the role `admin` has the authority to update the implementation 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 therefore execute potential malicious functionality in the implementation contract.

Recommendation

We recommend that the team make efforts to restrict access to the admin of the proxy contract. 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, 03/19/2024]`: "The admin of the oracle contracts will be the Normal timelock contract [1] on BNB chain, so the upgrade will be doable only via Governance.

[1] https://bscscan.com/address/0x939bD8d64c0A9583A7Dcea9933f7b21697ab6396

In the rest of the networks, the admin of the oracle contracts will be initially the Guardian multisig wallets:

- Ethereum: https://etherscan.io/address/0x285960C5B22fD66A736C7136967A3eB15e93CC67
- opBNB: https://opbnbscan.com/address/0xC46796a21a3A9FAB6546aF3434F2eBfFd0604207

This role will be transferred to the Normal timelock contracts on each network as soon as we complete the deployment of the Multichain governance project."

`[CertiK, 03/19/2024]` : The client has provided all steps towards mitigation on the BSC chain. In order to mitigate the finding completely, please provide the relevant information corresponding the new networks when they are available.

POV-01 | MISSING VALIDATION OF `PendleOracle` PARAMETERS

Category	Severity	Location	Status
Logical Issue	Medium	contracts/oracles/PendleOracle.sol (Base): 40~42	Resolved

Description

In the contract `PendleOracle`, the cardinality of the observations is not checked to be able to handle the `TWAP_DURATION`. In addition, the oldest observation is not checked to have been at least the `TWAP_DURATION` in the past.

Recommendation

We recommend checking that the cardinality of observations can handle the `TWAP_DURATION` and that the oldest observation is at least the `TWAP_DURATION` in the past. See here for more information https://docs.pendle.finance/Developers/Integration/PTOracle#oracle-preparation.

Alleviation

`[CertiK, 03/19/2024]`: The client made the recommended changes in commit 90ea2b114e8d27a27ccf7258353504303fa092a3.

SBB-01 | POTENTIAL DIVIDE BY ZERO

Category	Severity	Location	Status
Logical Issue	Minor	contracts/oracles/StkBNBOracle.sol (Base): 38~39	Resolved

Description

Function `getUnderlyingAmount` in contract `StkBNBOracle` is missing a check ensuring that `poolTokenSupply` is nonzero before using it in division.

The expression `(exchangeRateData.totalWei * EXP_SCALE) / exchangeRateData.poolTokenSupply` may divide by zero.

Recommendation

We recommend the use of conditionals to rule out the possibility of a division-by-zero.

Alleviation

`[CertiK, 03/19/2024]`: The client made changes resolving the finding in commit 423c5520f2ffa11d4d5d0933b804c6ae98ba8c7e.

OJO-01 | DISCUSSION ON USE CASES OF `OneJumpOracle`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/oracles/OneJumpOracle.sol (Base): 34~36	Resolved

Description

The `OneJumpOracle` is designed to fetch the amount of underlying token for 1 correlated token via an intermediate oracle. For example, this could be used on Arbitrum to get the wstETH-ETH exchange rate via ChainLink assuming stETH is 1 to 1 with ETH. We would like to know the intended use cases so that they can be properly considered.

Recommendation

We recommend clarifying the design intent.

Alleviation

`[Venus, 03/19/2024]`: "The first use case of OneJumpOracle is to fetch LDO price on Ethereum. Chainlink has a feed for LDO-ETH

We also plan to use OneJumpOracle for sfrxETH, getting the sfrxETH/ETH price feed using the FRAX oracle, and then multiplying it by the ETH/USD price"

`[CertiK, 03/22/2024]`: Thank you for the clarifications.

Considering this, can you clarify the intended `INTERMEDIATE_ORACLE`s that may be used? In particular, it is essential that these oracles return the underlying token amount for 1 correlated token scaled by underlying token decimals. This can be a particular issue if tokens that do not have 18 decimals are used in the future.

`[Venus, 04/11/2024]`: The idea is to use as INTERMEDIATE_ORACLE oracles following the Compound oracle convention: returning values scaled to 36 - decimals(assets).

We realized that is incompatible with the requirement from the CorrelatedTokenOracle: OneJumpOracle must return a value scaled to the underlying asset decimals.

We have pushed the following commits (code + natspec) to fix this issue:

https://github.com/VenusProtocol/oracle/commit/9efcae374301d2218e1f53e3f3df638f35817241
https://github.com/VenusProtocol/oracle/commit/eeab989a9c6e2f44555cba629b944fb99835f193

`[CertiK, 04/12/2024]`: Considering these changes we mark this finding as resolved. However, it should be noted that the `INTERMEDIATE_ORACLE` should always return a value scaled to 36 - decimals(assets).

SFO-01 | PARAMETER ORDER CAN BE SWAPPED

Category	Severity	Location	Status
Coding Style	Informational	contracts/oracles/SFraxOracle.sol (Base): 16~17, 17~18, 19~20	Resolved

Description

The `constructor` of the `SFraxOracle` contract places input `_frax` as the first parameter and `_sFrax` as the second parameter, then swaps their order when used in the `constructor` of `CorrelatedTokenOracle`.

Recommendation

We recommend placing input parameter `_sFrax` first in the `constructor` of `SFraxOracle` in order to maintain code readability and avoid any confusion.

Alleviation

`[CertiK, 03/19/2024]`: The client made changes resolving the finding in commit 12e9d524f4ebc02319a03b34f231fbb26c1c03cd.

SFO-02 | THIRD PARTY'S RATE MAY BE MANIPULATED

Category	Severity	Location	Status
Logical Issue	Informational	contracts/oracles/SFraxOracle.sol (Base): 26	Acknowledged

Description

Some of the referenced tokens utilize ERC4626 vaults, whose rate can be manipulated. Additionally, depending on the implementation of the other tokens and staking contracts, their rates may also be manipulatable.

For example `sFrax` is an ERC4626 vault utilizing an internal balance for the `totalAssets()`. In this case shares are converted to assets by multiplying by `totalAssets()` and dividing by the total supply of shares. However, this can be manipulated via multiple calls to `deposit()` and `redeem()` by depositing an amount that would cause the maximum rounding error. The rounded amount is then distributed amongst the shares, increasing their value and thus allowing for subsequent deposit calls to take advantage of a greater rounding error.

Note that the `sFrax.deposit()` function makes the following check

```
require((shares = previewDeposit(assets)) != 0, "ZERO_SHARES");
```

Which, is why the exploiter always chooses the maximum amount of assets to deposit, that will give 1 share.

For example assume that `totalSupply = 1` and `storedTotalAssets = 2` and `_rewardToDistribute = 0` so that `totalAssets() = 2`, so the initial rate is 1 share to 2 assets.

- Exploiter calls `deposit(3)` to receive 3*1/2 = 1 share and then `totalSupply = 2` and `totalAssets() = 5`.
- Exploiter then calls `redeem(1)` to receive 5 * 1 /2 = 2 assets and `totalSupply = 1` and `totalAssets() = 3`. Rate is now 3 assets to 1 share.
- Exploiter calls `deposit(5)` to receive 5*1/3 = 1 share and then `totalSupply = 2` and `totalAssets() = 8`.
- Exploiter then calls `redeem(1)` to receive 8 * 1 /2 = 4 assets and `totalSupply = 1` and `totalAssets() = 4`. Rate is now 4 assets to 1 share.
- Exploiter calls `deposit(7)` to receive 7*1/4 = 1 share and then `totalSupply = 2` and `totalAssets() = 11`.
- Exploiter then calls `redeem(1)` to receive 11 * 1 /2 = 5 assets and `totalSupply = 1` and `totalAssets() = 6`. Rate is now 6 assets to 1 share.
- This continues until the attacker reaches the desired manipulated rate for each share.

Note, that the cost of this manipulation is at least the desired rate manipulation times the `totalSupply` they do not own. So that if the `totalSupply` not held by a single entity is great enough, then the price manipulation is infeasible.

Recommendation

We recommend identifying the correlated token oracles interacting with tokens that may have their rates manipulated under certain scenarios. For those oracles, we recommend always using a secondary oracle that is not affected by the rate manipulation to validate against.

Alleviation

`[CertiK, 04/12/2024]`: The client acknowledged the issue and opted to not make any changes.

VPB-01 | DISCUSSION ON EXCHANGE RATE CHOICE

Category	Severity	Location	Status
Inconsistency	Informational	contracts/oracles/WeETHOracle.sol (Base): 26; contracts/oracles/WstETHOracle.sol (Base): 26	Resolved

Description

In some of the oracle contracts, there are multiple methods to fetch the exchange rate.

- In `WeETHOracle`, it gets the underlying amount via the call `IWeETH(CORRELATED_TOKEN).getEETHByWeETH(1 ether)`. However, this makes the external call `liquidityPool.amountForShare(_weETHAmount);` so that instead `amountForShare()` could be called directly on the `liquidityPool`.
- In `WstETHOracle`, it gets the underlying amount via the call `IStETH(UNDERLYING_TOKEN).getPooledEthByShares(1 ether)`. However, this could also be obtained by calling `getStETHByWstETH()` in the `wstETH` contract, which then makes an external call `stETH.getPooledEthByShares()`.

Recommendation

We recommend clarifying the choice of method.

Alleviation

`[CertiK, 03/19/2024]`: The client stated they would use the method that avoids an external call to be gas efficient. They made changes resolving this finding in commit ac18ad7bdf47290555b2040cb6e29b6d49f12458.

VPB-03 | BNB ADDRESS INPUT CAN BE CONSTANT

Category	Severity	Location	Status
Logical Issue	Informational	contracts/oracles/AnkrBNBOracle.sol (Base): 17; contracts/oracles/BNBxOracle.sol (Base): 24; contracts/oracles/SlisBNBOracle.sol (Base): 23; contracts/oracles/StkBNBOracle.sol (Base): 25	Resolved

Description

Many of the oracle contracts take the BNB address as an input. However, other contracts in the codebase use the constant address of 0xbBbBBBBbbBBBbbbBbbBbbbbBBbBbbbbBbBbbBBbB.

Recommendation

We recommend using the constant address to be consistent throughout the codebase.

Alleviation

`[CertiK, 03/19/2024]`: The client made changes resolving the finding in commit 085463688294f8c1774a9673a9669025d07f035d.

VPB-04 | INCONSISTENT INPUT NAMING CONVENTION

Category	Severity	Location	Status
Inconsistency	Informational	contracts/oracles/AnkrBNBOracle.sol (Base): 16~18; contracts/oracles/BNBxOracle.sol (Base): 22~25; contracts/oracles/OneJumpOracle.sol (Base): 21~24; contracts/oracles/PendleOracle.sol (Base): 29~34; contracts/oracles/SFraxOracle.sol (Base): 16~18; contracts/oracles/SFrxETHOracle.sol (Base): 16~18; contracts/oracles/SlisBNBOracle.sol (Base): 21~24; contracts/oracles/StkBNBOracle.sol (Base): 23~26; contracts/oracles/WBETHOracle.sol (Base): 16~18; contracts/oracles/WeETHOracle.sol (Base): 16~18; contracts/oracles/WstETHOracle.sol (Base): 16~18	Resolved

Description

The usage of a leading underscore or not for input names is inconsistent.

Recommendation

We recommend choosing a convention and making the contracts consistent with that convention.

Alleviation

`[CertiK, 03/19/2024]`: The client made changes resolving the finding in commit f0f5ed00983a99e452f69ca65349c1e90b523ade.

VPB-05 | ORACLES ASSUME TOKENS HAVE 18 DECIMALS

Category	Severity	Location	Status
Logical Issue	Informational	contracts/oracles/AnkrBNBOracle.sol (Base): 26; contracts/oracles/BNBxOracle.sol (Base): 36; contracts/oracles/SFraxOracle.sol (Base): 26; contracts/oracles/SFrxETHOracle.sol (Base): 26; contracts/oracles/SlisBNBOracle.sol (Base): 35; contracts/oracles/StkBNBOracle.sol (Base): 38; contracts/oracles/WBETHOracle.sol (Base): 26; contracts/oracles/WeETHOracle.sol (Base): 26; contracts/oracles/WstETHOracle.sol (Base): 26	Acknowledged

Description

While the tokens used all have 18 decimals, this is implicitly assumed in the `getUnderlyingAmount()` functions where `1 ether` of `EXP_SCALE` is used. To be more explicit, the token decimals can be used to ensure future oracles take into consideration that this value is dependent on the token decimals.

Recommendation

We recommend explicitly handling the token decimals in `getUnderlyingAmount()`. Note, that if it is assumed the token decimals will not change, an immutable variable can be added and set to the token decimals upon deployment. Note, that if this option is chosen and the token is upgradeable, then it should be monitored in case the token is upgraded and the token decimals are changed.

Alleviation

`[CertiK, 03/19/2024]`: The client made changes in commit b14b83278d88d64f12a9b433f272c449968118e1.

However, the substitution of `1 ether` for `EXP_SCALE` still assumes that all oracles are for tokens with 18 decimals.

`[Venus, 04/11/2024]`: We'll add simulations and fork tests before using the new oracles, so the mentioned potential issue should be discovered (if we don't take into account the token decimals). So, we prefer to keep it as it is now.

VPU-01 | TYPOS AND INCONSISTENCIES

Category	Severity	Location	Status
Coding Style	Informational	contracts/oracles/OneJumpOracle.sol (Base): 13~14, 21~22, 35~36; contracts/oracles/PendleOracle.sol (Base): 11~12; contracts/oracles/common/CorrelatedTokenOracle.sol (Base): 37~38, 38~39, 39~40, 44~45, 53~54, 58~59; contracts/oracles/OneJumpOracle.sol (Update1): 14, 34~36	Resolved

Description

CorrelatedTokenOracle.sol

- The comment above function `getPrice()` misspells “scaled” as “scalked.”
- The comment within function `getPrice()` states that the return value is `underlyingAmount (for 1 liquid staked token) * underlyingUSDPrice / 1e18` however, the product is divided by `10**decimals` which may not be `1e18` in general.
- Comments throughout both functions `getPrice()` and `getUnderlyingAmount()` refer to a “liquid staked token” when it is more accurate to refer to a “correlated token” since the correlated token may not be a liquid staked token in every case of use.

PendleOracle.sol

- The comment above the contract `PendleOracle` uses article “an” instead of article “a” in the notice: `This oracle fetches the price of an pendle token`

Recommendation

We recommend correcting the typos and inaccuracies above.

Alleviation

`[CertiK, 04/11/2024]`: The client made changes resolving the finding in commits

- c167681a7b1bf52a194183483604d07d74b5d955;
- c167681a7b1bf52a194183483604d07d74b5d955;
- 6f846481fab25b82bda9591ac57c2fa0d8e58f62;
- 0e7f175b47b84708c79dbb842e9b34e15b9f4672;

- fa59be36c299479a76e69872a146696214e18eaa;

During remediations another inconsistency was found. In `OneJumpOracle`,

- In the case of example given, LDO-ETH, LDO is not a correlated token to underlying ETH.

`[Venus, 04/11/2024]`: "Regarding the LDO-ETH example, even not being correlated tokens, the OneJumpOracle can take advantange of the CorrelatedTokenOracle code to provide the prices of assets when the available price feed is not directly to USD. We prefer to keep it as it is now"

OPTIMIZATIONS | VENUS - CORRELATED TOKEN ORACLES

ID	Title	Category	Severity	Status
CTO-01	Custom Errors Can Be Used	Gas Optimization	Optimization	Resolved

CTO-01 | CUSTOM ERRORS CAN BE USED

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/oracles/common/CorrelatedTokenOracle.sol (Base): 42~43	Resolved

Description

Function `getPrice()` within contract `CorrelatedTokenOracle` uses a string error message for the check that the input `asset` is `CORRELATED_TOKEN`.

Recommendation

We recommend considering the use of custom errors to reduce gas costs.

Reference: https://blog.soliditylang.org/2021/04/21/custom-errors/.

Alleviation

`[CertiK, 03/19/2024]`: The client made changes which resolve the finding in commit 5c4a5316e57bf3c10c2470b78f6255bae6b7930a.

APPENDIX | VENUS - CORRELATED TOKEN ORACLES

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.
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.