CERTIK

Security Assessment

Venus - Oracle Update

CertiK Assessed on Jul 17th, 2023

CertiK Assessed on Jul 17th, 2023

Venus - Oracle Update

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 07/17/2023	N/A

CODEBASE

https://github.com/VenusProtocol/oracle/

View All in Codebase Page

COMMITS

base: 3851f8f414401fd694d2d9113e93fdbe0a08a3af

update1: bd5e433453f0515db7e065e69dcb1c43f984ed44

update2: 3a7f9e2d1e0e7d56dc0642f9b1ef3c43f88d61d6

View All in Codebase Page

Vulnerability Summary

5	3	2	0	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
2	Major	2 Mitigated	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
0	Medium		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.
2	Informational	2 Resolved	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 - ORACLE UPDATE

CODEBASE | VENUS - ORACLE UPDATE

Repository

https://github.com/VenusProtocol/oracle/

Commit

base: 3851f8f414401fd694d2d9113e93fdbe0a08a3af
update1: bd5e433453f0515db7e065e69dcb1c43f984ed44
update2: 3a7f9e2d1e0e7d56dc0642f9b1ef3c43f88d61d6
update3: 5b35b64174002be0ac909950828c69e706e584dd
update4: 82c4004090d248ad7816813f89093bed6b3c01a8

AUDIT SCOPE | VENUS - ORACLE UPDATE

10 files audited ● 6 files with Mitigated findings ● 1 file with Resolved findings ● 3 files without findings

ID	Repo	Commit	File	SHA256 Checksum
● BVV	VenusProtocol/oracle	3851f8f	contracts/oracles/BoundValidator.sol	7fca441f1a8bca4dfcf4d03ffa9fa9495eb30a8900a534796cf14fd00eebda14
● ROV	VenusProtocol/oracle	3851f8f	contracts/ResilientOracle.sol	5e4c186eba37c454b522db4fef1ff48fe9c1f98a42bbc2e48034427c2eb4e948
● BOV	VenusProtocol/oracle	3851f8f	contracts/oracles/BinanceOracle.sol	d20980b747d9c2fdc4e4723a170d5cc07da041b113085abbdb5a71f8e34d4d1b
● COV	VenusProtocol/oracle	3851f8f	contracts/oracles/ChainlinkOracle.sol	0b17d689586224f34d8524f47dfe2c2be852a95ebb3d67a30ee93999e2d36e97
● TOV	VenusProtocol/oracle	3851f8f	contracts/oracles/TwapOracle.sol	0c00e39ae346ebb221b3c0f4f69df71325918d4bf772fcb6b68e54bc01013015
● POV	VenusProtocol/oracle	3851f8f	contracts/oracles/PythOracle.sol	f8d3f83765ea363e774b18512b90f84e639621e3b5f2891270bfaeb3667f2f93
● BOP	VenusProtocol/oracle	5b35b64	contracts/oracles/BinanceOracle.sol	66e825e5090d04ceb98ca6e657f39d4120c658a2bc8532bc7611cbb90f186796
● FRV	VenusProtocol/oracle	5b35b64	contracts/interfaces/FeedRegistryInterface.sol	cad4841a41bb5d2016f025e0b9be401e980d7e7dd6a564a9c829ac092aab2574
● FRI	VenusProtocol/oracle	3851f8f	contracts/interfaces/FeedRegistryInterface.sol	83b30461f3429c23a060e7bb0c3f138292846d2f4a373187a2d1c1421eed81f0
● OIV	VenusProtocol/oracle	3851f8f	contracts/interfaces/OracleInterface.sol	e8ffce4d22e3aac883550e013f92ff1282d468f9a061bf725e1226f7484f4677

APPROACH & METHODS | VENUS - ORACLE UPDATE

This report has been prepared for Venus to discover issues and vulnerabilities in the source code of the Venus - Oracle Update 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 - ORACLE UPDATE

This audit concerns the changes implemented in the following PRs:

https://github.com/VenusProtocol/oracle/pull/65

The main change introduced in this PR is to get prices of an asset directly, as opposed to getting price. This is done by removing `getUnderlyingPrice()`, which takes a `vToken` as input, from the main, pivot, and fallback oracles and replacing it with `getPrice()`, which takes the asset as input. In addition, price feeds and bound validations also have been changed to accept assets directly as inputs. In `ResilientOracle`, there is now the added function `getPrice()`, which takes an asset as input. However, the function `getUnderlyingPrice()` is also still present and functions as it did previously.

https://github.com/VenusProtocol/oracle/pull/107

The main change in the PR is to revert back to using `latestRoundDataByName()` for the `BinanceOracle` which will fetch the round data by inputting a name as opposed to an address. However, the names may differ than the token symbols fetched from the token contracts, so there was functionality added to store an `overrideSymbol`, which can be used in such cases.

For more information that can be found in the previous audit see: https://skynet.certik.com/projects/venus. The previous audit can be found in the *Code Audit History* section under the title **Venus - Oracle**.

DEPENDENCIES | VENUS - ORACLE UPDATE

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:

- Chainlink Oracle
- Binance Oracle
- PythOracle
- AMM's Such As PancakeSwap

The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

Recommendations

We recommend constantly monitoring the third parties involved to mitigate any side effects that may occur when unexpected changes are introduced.

FINDINGS | VENUS - ORACLE UPDATE

5	0	2	0	1	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Venus - Oracle Update . Through this audit, we have uncovered 5 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
VPB-02	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	**Mitigated**
VPB-03	**Centralization Related Risks**	**Centralization**	**Major**	**Mitigated**
BOV-03	Missing Zero Address Validation	Volatile Code	Minor	Resolved
ROV-01	`fallbackPrice` Is Tested Against `mainPrice`	Logical Issue	Informational	Resolved
VPU-01	Typos And Inconsistencies	Inconsistency	Informational	Resolved

VPB-02 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	Major	contracts/ResilientOracle.sol (base): 142; contracts/oracles/BinanceOracle.sol (base): 61; contracts/oracles/BoundValidator.sol (base): 61; contracts/oracles/ChainlinkOracle.sol (base): 88; contracts/oracles/PythOracle.sol (base): 97; contracts/oracles/TwapOracle.sol (base): 110	Mitigated

Description

`BinanceOracle`, `BoundValidator`, `ChainlinkOracle`, `PythOracle`, `TwapOracle`, and `ResilientOracle` are upgradeable contracts. The owner can upgrade the contract without the community's commitment. If an attacker compromises the account, he can change the implementation of the contract and drain tokens from the contract as well as change the logic of the contract to return incorrect prices.

Recommendation

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

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

Short Term:

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

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

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

- Provide the deployed time-lock address.

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

Long Term:

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

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

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

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

Permanent:

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

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

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

Alleviation

`[Venus, 07/05/2023]`: The owner of these contracts was transferred to 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, these contracts will be upgraded only via a Normal VIP, involving the community in the process.

VPB-03 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	Major	contracts/ResilientOracle.sol (base): 152, 161, 199, 219, 304; contracts/oracles/BinanceOracle.sol (base): 47; contracts/oracles/BoundValidator.sol (base): 75; contracts/oracles/ChainlinkOracle.sol (base): 60, 104; contracts/oracles/PythOracle.sol (base): 86, 115; contracts/oracles/TwapOracle.sol (base): 155	Mitigated

Description

In the contract `BinanceOracle`, the `DEFAULT_ADMIN_ROLE` can grant access to the following functions:

- `setMaxStalePeriod()`
- `setSymbolOverride()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or access to the functions may allow a hacker to do the following:

- set the `maxStalePeriod` to any nonzero value. If they set the value to be very large, then this allows old prices to be valid. If the value is set to be very small, then reasonably recent prices will be considered invalid.
- set the override symbol for a symbol to another value. This can reference the wrong feed or cause the calls to revert.

In the contract `BoundValidator`, the `DEFAULT_ADMIN_ROLE` can grant access to the following functions:

- `setValidateConfigs()`
- `setValidateConfig()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or access to the functions may allow a hacker to do the following:

- set the upper and lower bound validation ratios for an asset. In particular this allows them to set the ratio to be a very small range, in which case most time the price will not be validated. Or they can set the ratio to a large range, allowing prices to be validated when they are not reasonably close to one another.

In the contract `ChainlinkOracle`, the `DEFAULT_ADMIN_ROLE` can grant access to the following functions:

- `setDirectPrice()`
- `setTokenConfigs()`
- `setTokenConfig()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or access to the functions may allow a hacker to do the following:

- change the forced prices for assets. If only this oracle is used as the main oracle, this would allow the hacker to set the exact price they want for an asset. If it is used as the pivot, then the value can be set to always validate the fallback or main oracle, even if the oracle is compromised and returns unreasonable prices. If it is used as the fallback, it can be used to get the best price that the pivot would validate or to validate the main oracles price, even if it is unreasonable. If this is the only oracle used, then this allows a hacker to set the price they want for an asset.
- set the feed address and `maxStalePeriod` for an asset. In particular a hacker could set the feed address of the asset to a feed that is not for the asset and USD and use the incorrect price to exploit funds from the protocol. The hacker can also set the `maxStalePeriod` to a small value, so that reasonably recent prices are invalid, or to a large value so that old prices may be used.

In the contract `PythOracle`, the `DEFAULT_ADMIN_ROLE` can grant access to the following functions:

- `setUnderlyingPythOracle()`
- `setTokenConfigs()`
- `setTokenConfig()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or access to the functions may allow a hacker to do the following:

- set the `underlyingPythOracle` to an address of a malicious contract that will return incorrect prices that can be used to exploit the protocol.
- set the `pythId` and `maxStalePeriod` for an asset. In particular a hacker could set the `pythId` of the asset to a feed that is not for the asset and USD and use the incorrect price to exploit funds from the protocol. The hacker can also set the `maxStalePeriod` to a small value, so that reasonably recent prices are invalid, or to a large value so that old prices may be used.

In the contract `TwapOracle`, the `DEFAULT_ADMIN_ROLE` can grant access to the following functions:

- `setTokenConfigs()`
- `setTokenConfig()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or access to the functions may allow a hacker to do the following:

- set the `baseUnit`, `pancakePool`, `isBnbBased`, `isReversedPool`, and `anchorPeriod` for any asset. A hacker can change these values to manipulate the price that is given for the asset to exploit funds from the protocol.

In the contract `ResilientOracle`, the `DEFAULT_ADMIN_ROLE` can grant access to the following functions:

- `pause()`
- `unpause()`

- `setOracle()`
- `enableOracle()`
- `setTokenConfigs()`
- `setTokenConfig()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or access to the functions may allow a hacker to do the following:

- pause the oracle, so that any call to `getUnderlyingPrice()` or `getPrice()` will revert. This can allow a hacker to perform a denial of service attack.
- unpause the oracle, allowing `getUnderlyingPrice()` or `getPrice()` to be called. This can allow the hacker to exploit the protocol if it was paused due to a bug.
- set the main, pivot, or fallback oracles for an asset. A hacker could change these addresses to malicious contracts that will return incorrect prices allowing the hacker to exploit funds from the protocol.
- set if the main, pivot, or fallback oracles are enabled for an asset. If a hacker has compromised the main oracle, they can disable the pivot so that the main price will be used and the hacker can use the incorrect price to exploit funds from the protocol. They can also perform a denial of service by disabling the oracles.

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 recommend carefully managing 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., multi-signature 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 of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key being 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 of 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, 07/05/2023]`: We'll use the AccessControlManager (ACM) deployed at https://bscscan.com/address/0x4788629abc6cfca10f9f969efdeaa1cf70c23555.

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

0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 is granted to execute the mentioned functions. Moreover, [a] (Fast-track) and [b] (Critical) are also granted to execute pause() and unpause() functions in the ResilientOracle. These are the Timelock contracts to execute VIP's with a shorter delay.

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

normal: 24 hours voting + 48 hours delay fast-track: 24 hours voting + 6 hours delay critical: 6 hours voting + 1 hour delay

Regarding the role, specifically, the sequence in the ACM was:

In [1] the ACM was created, and the address 0x55a9f5374af30e3045fb491f1da3c2e8a74d168d had the DEFAULT_ADMIN_ROLE.

In [2], 0x55a9f5374af30e3045fb491f1da3c2e8a74d168d gave the DEFAULT_ADMIN_ROLE to 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396.

In [3] 0x55a9f5374af30e3045fb491f1da3c2e8a74d168d renounced to the DEFAULT_ADMIN_ROLE.

Therefore, we consider this setup safe enough and don't plan to do any other changes.

[a] https://bscscan.com/address/0x555ba73dB1b006F3f2C7dB7126d6e4343aDBce02

[b] https://bscscan.com/address/0x213c446ec11e45b15a6E29C1C1b402B8897f606d

[1] https://bscscan.com/tx/0x3eb2ef9b54b1ec3873e07fc9994d32de6fe6c9bc9277c17619c6fa6701340ae0

[2] https://bscscan.com/tx/0x66b32b0d8918b43e43e2b6104927273f012b81ad8ee30d1284c6067aa761b687

[3] https://bscscan.com/tx/0x2a4b3b21f5acd9fb73c9fa740d9a8a123780bdb01ec712baac639576df33d7d4

BOV-03 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/oracles/BinanceOracle.sol (base): 66, 67	Resolved

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities.

```
66          sidRegistryAddress = _sidRegistryAddress;
```

- `_sidRegistryAddress` is not zero-checked before being used.

```
67          WBNB = _WBNB;
```

- `_WBNB` is not zero-checked before being used.

Recommendation

We recommend adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

`[CertiK, 07/07/2023]`: The client made the recommended changes in commits:

- ad9cd99458b9ee30dc4e852d643de2d8c68f4a2e;
- 30da9b176b0cf7853afaec48dd53100bc55176dc.

ROV-01 | `fallbackPrice` IS TESTED AGAINST `mainPrice`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/ResilientOracle.sol (base): 357	Resolved

Description

In the function `_getPrice()`, if the validation of the `mainOraclePrice` vs. `pivotPrice` and `fallbackOraclePrice` vs. `pivotPrice` fails. Then the `fallbackPrice` is tested against the `mainPrice` and if the validation passes, then the `mainPrice` is returned. However, if the `mainPrice` is being returned then it should be the price that is tested.

Recommendation

We recommend calling `boundValidator.validatePriceWithAnchorPrice()` with `mainPrice` as the input `reportedPrice` and `fallbackPrice` as the input `anchorPrice`.

Alleviation

`[CertiK, 07/05/2023]`: The client made the recommended changes in commit: cd1a2b9273bfcc568d2c5e9b59d570d13c51f725.

VPU-01 | TYPOS AND INCONSISTENCIES

Category	Severity	Location	Status
Inconsistency	Informational	contracts/ResilientOracle.sol (base): 112~113, 240, 316, 366, 376, 407, 412; contracts/oracles/BinanceOracle.sol (base): 87; contracts/oracles/ChainlinkOracle.sol (base): 16, 46, 55, 57, 112, 113, 130, 133, 148, 150, 151; contracts/oracles/PythOracle.sol (base): 108, 123; contracts/oracles/TwapOracle.sol (base): 26, 115, 117, 171, 203; contracts/ResilientOracle.sol (update1): 183~184, 206, 299; contracts/oracles/BinanceOracle.sol (update1): 67	Resolved

Description

In the contract `BinanceOracle`, the comment above the function `getPrice()` states: "asset Address of the address". However, this should say this is the address of the asset.

In the contract `ChainlinkOracle`, the comment above the `constructor()` states: "Sets immutable variables". However, it no longer sets any immutable variables.

In the contract `ChainlinkOracle`, the comments above `getPrice()` and `_getPriceInternal()` do not reflect that the price that is returned could also be the manually set price.

In the contract `ChainlinkOracle`, the comment above the function `getPrice()` states: "asset Address of the address". However, this should say this is the address of the asset.

In the contract `ChainlinkOracle`, `PythOracle`, and `ResilientOracle`, "underlying" is used often, when it has been changed to be for any asset, not just the asset underlying a `vToken`. In particular, above the function `_getChainlinkPrice()` it states: "Get the Chainlink price for the underlying asset of a given vToken".

In the contract `PythOracle`, the comment above the function `getPrice()` states: "asset Address of the address". However, this should say this is the address of the asset.

In the contract `PythOracle`, the comment above `setTokenConfig()` mentions `vToken` when it could be any asset.

In the contract `ResilientOracle`, the comment above the function `_getFallbackOraclePrice()` states: "This function won't revert when the price is 0 because getUnderlyingPrice checks if price is > 0". However, it is more accurate to say that _getPrice checks if price is > 0.

In the contract `ResilientOracle`, the comment above the function `updateAssetPrice()` states: "This function should always be called before calling getUnderlyingPrice". However, it is more accurate to say that it should be called before calling getPrice.

In the contract `ResilientOracle`, the comment above the function `getOracle()` states: "Gets oracle and enabled status by vToken address". However, it gets it by asset address.

In the contract `ResilientOracle`, the comment above the modifier `checkTokenConfigExistence` uses `vToken`, when it can be used for any asset.

Recommendation

We recommend fixing the typos/inconsistencies mentioned above.

Alleviation

`[CertiK, 07/07/2023]`: The client made the recommended changes in commits:

- 9677dd613f24fc8ca6fff220301e647657ba4b74;
- bd5e433453f0515db7e065e69dcb1c43f984ed44;
- d5ea65ca554467bd4045d903837204c18bf7a58a;
- 37238cc629e1c67431489ceba3d6fb38d2540966;

OPTIMIZATIONS | VENUS - ORACLE UPDATE

ID	Title	Category	Severity	Status
BOP-01	`WBNB` Can Be Made A Constant	Coding Style	Optimization	Resolved
BOP-02	Inefficient `memory` & `storage` Management	Gas Optimization	Optimization	Resolved
BOV-01	Unchecked Blocks Can Optimize Contract	Gas Optimization	Optimization	Resolved
BOV-02	Unused Function	Code Optimization	Optimization	Resolved

BOP-01 | `WBNB` CAN BE MADE A CONSTANT

Category	Severity	Location	Status
Coding Style	Optimization	contracts/oracles/BinanceOracle.sol (update1): 30~31	Resolved

Description

A public variable `WBNB` was added to `BinanceOracle`, to store the address of `wBNB`. As the contract address is known it can be set as a constant to save gas. In addition, if `WBNB` does not need to be initialized, then `initialize()` does not need to be called again if the same `_sidRegistryAddress` and `_accessControlManager` are to be used. If this is the case, then the `initialize()` function can be adjusted to remain consistent with the previous version as it does not need to be called.

Recommendation

We recommend considering making `WBNB` a constant to save gas.

Alleviation

`[CertiK, 07/10/2023]`: The client made `WBNB` an immutable variable in commits:

- 3a7f9e2d1e0e7d56dc0642f9b1ef3c43f88d61d6;
- 07995d07982ae2f195084a8900c80197a95fc12b.

BOP-02 | INEFFICIENT `memory` & `storage` MANAGEMENT

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/oracles/BinanceOracle.sol (update3): 120	Resolved

Description

In the function `getPrice()`, the location specifier `storage` is used for `overrideSymbol`. However, it is only ever used to read from storage and instead the value can be stored in `memory` to reduce gas costs by reading from memory as opposed to storage.

Recommendation

We recommend using `memory` instead of `storage` to reduce gas costs.

Alleviation

`[CertiK, 07/17/2023]`: The client resolved the finding by changing the specifier to `memory` in the commit 82c4004090d248ad7816813f89093bed6b3c01a8

BOV-01 | UNCHECKED BLOCKS CAN OPTIMIZE CONTRACT

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/oracles/BinanceOracle.sol (base): 114	Resolved

Description

In the contract `BinanceOracle`, the function `_getPrice()` checks that `block.timestamp < updatedAt` and then makes the calculation `block.timestamp - updatedAt`. The check prevents the possibility of underflow, so that `deltaTime` can be declared and then inside an unchecked block assigned to be `block.timestamp - updatedAt`.

Recommendation

We recommend using unchecked blocks when overflow/underflow is not possible to save gas.

Alleviation

`[CertiK, 07/05/2023]`: The client made the recommended changes in commit: d0035d46725e59a477d4055d1fefac67cb998f44.

BOV-02 | UNUSED FUNCTION

Category	Severity	Location	Status
Code Optimization	Optimization	contracts/oracles/BinanceOracle.sol (base): 121~129	Resolved

Description

The function `compare()` is no longer used in the contract `BinanceOracle`.

Recommendation

We recommend removing the unused function to reduce the size of the deployed bytecode.

Alleviation

`[CertiK, 07/05/2023]`: The client made the recommended changes in commit: ef3b5d7904f96a4bc0f12189640224df41e9dc8f.

APPENDIX | VENUS - ORACLE UPDATE

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.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

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.