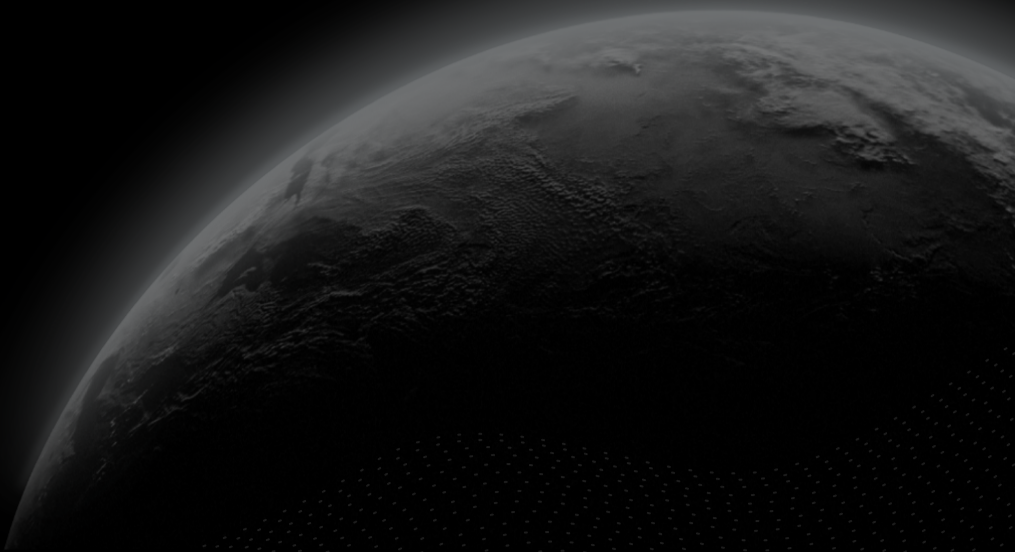




Security Assessment

Venus - VAI Peg

CertiK Assessed on May 24th, 2023





Certik Assessed on May 24th, 2023

Venus - VAI Peg

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

Executive Summary

TYPES

DeFi

ECOSYSTEM

Binance Smart Chain
(BSC)

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 05/24/2023

KEY COMPONENTS

N/A

CODEBASE

<https://github.com/VenusProtocol/venus-protocol>[...View All](#)

COMMITTS

base: [194c6064149451eb15fe9d22122eee9fec481e50](#)update1: [bc049d73a740971fc1d80661eb2df0f663b37fa6](#)update2: [e61dc01b31e40ce47b432da9ec38fdec5fc3eb5a](#)[...View All](#)

Vulnerability Summary



12

Total Findings

10

Resolved

2

Mitigated

0

Partially Resolved

0

Acknowledged

0

Declined

1 Critical

1 Resolved



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.

2 Medium

2 Resolved



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

2 Minor

2 Resolved



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

5 Informational

5 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 - VAI PEG

I **Summary**

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

I **Overview**

[PegStability.sol](#)

[swapVAIForStable\(\)](#)

[swapStableForVAI\(\)](#)

I **Dependencies**

[Third Party Dependencies](#)

[Out Of Scope Dependencies](#)

[Recommendations](#)

I **Findings**

[PSP-01 : No Approval Given For `safeTransferFrom\(\)`](#)

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

[GLOBAL-02 : Centralized Control of Contract Upgrade](#)

[PSP-02 : `swapVAIForStable\(\)` Does Not Meet Specification](#)

[PSP-03 : Possible To Be Unable To Repay Or Liquidate Borrow](#)

[PSP-04 : Incorrect Old Price Oracle](#)

[PSS-02 : `ReentrancyGuardUpgradeable` Is Not Initialized](#)

[PSP-05 : Assumption On Same Decimals](#)

[PSP-06 : Possible Underflow Revert](#)

[PSP-07 : Typos And Inconsistencies](#)

[PSP-10 : Comparison to Boolean Constant](#)

[PSP-01 : Fee Amount Not Checked In Initializer](#)

I **Optimizations**

[PSP-08 : Unchecked Blocks Can Optimize Contract](#)

[PSP-09 : Inefficient Event Parameter](#)

I **Appendix**

Disclaimer

CODEBASE | VENUS - VAI PEG

Repository

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

Commit

base: [194c6064149451eb15fe9d22122eee9fec481e50](#)


update1: [bc049d73a740971fc1d80661eb2df0f663b37fa6](#)

update2: [e61dc01b31e40ce47b432da9ec38fdec5fc3eb5a](#)

update3: [7f394e087828001b0c5b5e1022701b05c0150488](#)

AUDIT SCOPE | VENUS - VAI PEG

1 file audited ● 1 file with Resolved findings

ID	Repo	Commit	File	SHA256 Checksum
● PSP	VenusProtocol/venus-protocol	194c606	 PegStability.sol	87b94b35b863e682d4e1320039c8e8425ddeb9f45c2207d3190552bb01213a89

APPROACH & METHODS | VENUS - VAI PEG

This report has been prepared for Venus to discover issues and vulnerabilities in the source code of the Venus - VAI Peg 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.

OVERVIEW | VENUS - VAI PEG

■ PegStability.sol

This contract is designed to implement the two main mechanisms outlined in this proposal:

<https://community.venus.io/t/proposal-deploy-the-vai-peg-stability-module-and-supply-liquidity-to-pancakeswap-for-the-vai-usdt-pair/3461>. Here we give a summary of their implementation in the smart contract.

■ swapVAIForStable()

This function is designed to swap VAI for stable token. It has the inputs `receiver`, the address the stable token will be sent to, and `stableTknAmount`, the amount of stable token that the `receiver` will be transferred if the swap is successful. The amount of VAI is transferred from the `msg.sender` and is calculated using the amount in USD of the desired amount of stable token plus a fee. The amount in USD is determined by setting the price of the stable token to the maximum of 1\$ and the price of the stable token provided by the oracle. The fee is given by a percentage of the amount in USD that was calculated. The fee amount of VAI is transferred to the treasury and the amount in USD calculated of VAI is burned.

■ swapStableForVAI()

This function is designed to swap stable token for VAI. It has the inputs `receiver`, the address the stable token will be sent to, and `stableTknAmount`, the amount of stable token that the `msg.sender` will send to the PSM to swap. The amount of VAI minted to the `receiver` is calculated using the amount in USD of the received stable token minus a fee. This is determined by setting the price of the stable token to be the minimum of 1\$ and the price of the stable token provided by the oracle. The fee is given by a percentage of the amount in USD that was calculated. The fee amount of VAI is minted to the treasury and the amount in USD calculated minus the fee of VAI is minted to the `receiver`.

This function also checks for the `vaiMintCap`, which is the maximum amount of VAI that the PSM is allowed to mint. If the amount in USD calculated plus the amount of tokens already minted by the contract (minus any that have been burned through the use of `swapVAIForStable()`) exceeds this value, then it will revert.

DEPENDENCIES | VENUS - VAI PEG

Third Party Dependencies

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

- ERC20 Tokens (Stable Tokens)

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.

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:

- Oracles
- Comptroller

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

Manipulation of the oracle price directly affects the swapping logic and it should be ensured that the oracle always provides an accurate price. The comptroller is used to fetch the oracles address, so it should be ensured that it always returns the proper oracle address. Improper implementation of the AccessControlManager may allow users to call privileged functions when they have not been given access.

Recommendations

We recommend constantly monitoring the third parties involved to mitigate any side effects that may occur when unexpected changes are introduced. Additionally, we recommend all out-of-scope dependencies are carefully vetted to ensure they function as intended.

FINDINGS | VENUS - VAI PEG



12

Total Findings

1

Critical

2

Major

2

Medium

2

Minor

5

Informational

This report has been prepared to discover issues and vulnerabilities for Venus - VAI Peg. Through this audit, we have uncovered 12 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
PSP-01	No Approval Given For <code>safeTransferFrom()</code>	Logical Issue	Critical	● Resolved
GLOBAL-01	Centralization Related Risks	Centralization / Privilege	Major	● Mitigated
GLOBAL-02	Centralized Control Of Contract Upgrade	Centralization / Privilege	Major	● Mitigated
PSP-02	<code>swapVAIForStable()</code> Does Not Meet Specification	Logical Issue	Medium	● Resolved
PSP-03	Possible To Be Unable To Repay Or Liquidate Borrow	Coding Style	Medium	● Resolved
PSP-04	Incorrect Old Price Oracle	Logical Issue	Minor	● Resolved
PSS-02	<code>ReentrancyGuardUpgradeable</code> Is Not Initialized	Logical Issue	Minor	● Resolved
PSP-05	Assumption On Same Decimals	Logical Issue	Informational	● Resolved
PSP-06	Possible Underflow Revert	Coding Style	Informational	● Resolved
PSP-07	Typos And Inconsistencies	Coding Style	Informational	● Resolved
PSP-10	Comparison To Boolean Constant	Coding Style	Informational	● Resolved

ID	Title	Category	Severity	Status
PSV-01	Fee Amount Not Checked In Initializer	Coding Style	Informational	● Resolved

PSP-01 | NO APPROVAL GIVEN FOR `safeTransferFrom()`

Category	Severity	Location	Status
Logical Issue	● Critical	PegStability.sol (base): 155	● Resolved

Description

In the function `swapVAIForStable()`, `safeTransferFrom()` is used to transfer the `stableTknAmountUSD` from `address(this)` to the `receiver`. However, the contract does not give approval to itself, so that this will cause a revert preventing `swapVAIForStable()` from being called successfully.

Recommendation

We recommend using `safeTransfer()` instead of `safeTransferFrom()` to avoid giving unnecessary approvals.

Alleviation

[Certik]: The client made the recommended changes in commit: [9cbfb077aad0fa14386bf597ccffbc756388b362](#).

GLOBAL-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization / Privilege	● Major		● Mitigated

Description

PegStability.sol

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

- `pause()`
- `resume()`
- `setFeeIn()`
- `setFeeOut()`
- `setVaiMintCap()`
- `setVenusTreasury()`
- `setComptroller()`

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

- Pause the contract preventing anyone from swapping.
- Resume the contract, which may allow swaps to occur when the should not be allowed.
- Set the fees in BIPs to any percentage less than 100%.
- Set the `vaiMintCap` to any value, allowing any amount of VAI to be minted or if the value is set to 0 to prevent anyone swapping stable token to VAI.
- Set the treasury to a wallet they control to collect the fees for themselves.
- Set the comptroller address to a malicious contract that will return the address of a malicious oracle to provide inaccurate prices for the stable token.

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 ($\frac{2}{3}$, $\frac{3}{5}$) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

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

Long Term:

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

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

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

I Alleviation

[Venus] :

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 use during the Venus Improvement Proposals.

The idea is to grant 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 to execute the mentioned functions. Moreover, we'll allow [a] (Fast-track) and [b] (Critical) also to execute pause() and resume(). 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 is safe enough and don't plan to do any other change.

[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#eventlog>

GLOBAL-02 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization / Privilege	● Major		● Mitigated

Description

`PegStability` is an upgradeable contract. The owner can upgrade the contract without the community's commitment. If an attacker compromises the account, they can change the implementation of the contract and drain tokens from the contract.

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 ($\frac{2}{3}$, $\frac{3}{5}$) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

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

Long Term:

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

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

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

Permanent:

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

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

I Alleviation

[Venus] : The ownership of this contract will be transfer 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, this contract will be upgraded only via a Normal VIP, involving the community in the process.

PSP-02 | `swapVAIForStable()` DOES NOT MEET SPECIFICATION

Category	Severity	Location	Status
Logical Issue	● Medium	PegStability.sol (base): 155	● Resolved

Description

The function `swapVAIForStable()` is supposed to swap VAI for the input `stableTknAmount` of stable Token. However, it swaps VAI for the `stableTknAmountUSD` instead.

This does not meet the specification outlined in this proposal: <https://community.venus.io/t/proposal-deploy-the-vai-peg-stability-module-and-supply-liquidity-to-pancakeswap-for-the-vai-usdt-pair/3461>.

Recommendation

We recommend transferring the `receiver` the `stableTknAmount` as opposed to the `stableTknAmountUSD`.

Alleviation

[Certik]: The client made the recommended changes in commit: [32ba168d16a20098030935df65fc7749e1de57b2](#).

PSP-03 POSSIBLE TO BE UNABLE TO REPAY OR LIQUIDATE BORROW

Category	Severity	Location	Status
Coding Style	● Medium	PegStability.sol (base): 151~153 , 154	● Resolved

Description

When VAI is swapped for stable tokens, if `vaiMinted` is zero it will burn VAI that has been borrowed. This can cause the total amount of VAI in circulation to be less than the amount of VAI borrowed, which can make it impossible to repay or liquidate all the borrows. Note that this is possible because if USDT is over peg more VAI is burned than USDT is transferred out and if USDT is under peg less VAI is minted than USDT is transferred in.

Scenario

Assume for simplicity that there is 1000 VAI minted from borrows, that `vaiMinted` in the PSM is zero, the PSM holds 200 USDT, and the `vaiMintCap` is 100.

- Users take advantage of arbitrage and in total burn 200 VAI for 200 USDT. Leaving a circulating supply of 800 VAI.
- Market prices drop and there is a rush to repay borrows.
- Users repay as much VAI as possible, but there is currently only 800 in supply.
- Users mint VAI with the PSM to also help repay their borrow, however the maximum they can mint is 100.
- Thus there is still 100 VAI that is borrowed, but there is no way to mint more VAI except through borrowing it.
- This prevents users from repaying their loans and also prevents liquidators from liquidating positions that borrowed VAI.

This demonstrates how if the amount of VAI in circulation is less than the amount of VAI borrowed that it can prevent borrows from being repayed or liquidated.

Recommendation

We recommend ensuring that the amount of VAI in circulation is never less than the amount of VAI borrowed. One solution would be to check that `stableTknAmountUSD` is less than or equal to `vaiMinted` in the function `swapVAIForStable()`. This would ensure that only VAI that at most the VAI minted by the PSM could be burned.

Alleviation

[Certik]: The client added a check that the amount of VAI to be burned is less than or equal to the amount minted in commit: [bc049d73a740971fc1d80661eb2df0f663b37fa6](#).

PSP-04 | INCORRECT OLD PRICE ORACLE

Category	Severity	Location	Status
Logical Issue	Minor	PegStability.sol (base): 329	Resolved

Description

In the function `setPriceOracle()`, the `oldPriceOracleAddress` is set to be the input `priceOracle_`. However, it should be the current `priceOracle`.

Recommendation

We recommend setting `oldPriceOracleAddress` to `priceOracle` as opposed to `priceOracle_`.

Alleviation

[Certik]: The client made the recommended changes in commit: [e4dae48c58c2865b51d52f189f7aedef36ff17ad2](#).

In a later commit they removed this function and changed the functionality to fetch the oracle address from the comptroller: [7f394e087828001b0c5b5e1022701b05c0150488](#).

PSS-02 | ReentrancyGuardUpgradeable IS NOT INITIALIZED

Category	Severity	Location	Status
Logical Issue	● Minor	PegStability.sol (update1): 32 , 99~128	● Resolved

Description

Contract `PegStability` extends `ReentrancyGuardUpgradeable`, while `__ReentrancyGuard_init()` is not called in the initialize function. Generally, the initializer function of an upgradeable contract should always call all the initializer functions of the contracts that it extends.

Recommendation

We recommend initializing `ReentrancyGuardUpgradeable`.

Alleviation

[Certik]: The client made the recommended changes in commit: [c4c158e78e2b3e92b6c8a0e58e0592a0ffefca07](#).

PSP-05 | ASSUMPTION ON SAME DECIMALS

Category	Severity	Location	Status
Logical Issue	● Informational	PegStability.sol (base): 149 , 154 , 155	● Resolved

Description

Note that on BSC all tokens USDT, USDC, and VAI have 18 decimals so there is no issue with the logic. However, if stable tokens with other decimals are used, then the logic must be adjusted to account for the difference in decimals.

Recommendation

We recommend ensuring only tokens with 18 decimals will be used or to change the logic to account for different decimal places.

Alleviation

[Venus] : "As of now we do not plan to interact with stable tokens with different decimals value than 18. We plan to use only USDT and in the future maybe USDC."

PSP-06 | POSSIBLE UNDERFLOW REVERT

Category	Severity	Location	Status
Coding Style	● Informational	PegStability.sol (base): <u>151~153</u>	● Resolved

Description

As VAI can also be minted by borrowing against assets, there is more VAI in circulation than the value stored in `vaiMinted`. If a user attempts to call `swapVAIForStable()` so that the `stableTknAmountUSD` is greater than `vaiMinted` and `vaiMinted` is nonzero, then the call will revert with a non-descriptive error.

Recommendation

We recommend checking that `stableTknAmountUSD` is less than or equal to `vaiMinted` and reverting with a descriptive error if it is not. This recommendation is written to follow that of the finding *Possible To Be Unable To Repay Or Liquidate Borrow*.

Alleviation

[Certik]: The client made the recommended changes in commit: [bc049d73a740971fc1d80661eb2df0f663b37fa6](#).

PSP-07 | TYPOS AND INCONSISTENCIES

Category	Severity	Location	Status
Coding Style	● Informational	PegStability.sol (base): 145 , 148 , 167	● Resolved

Description

In the function `swapVAIForStable`, "enough" is misspelled as "enought".

The error message "Amount must be greater than zero" does not use a period as all other error messages do.

Recommendation

We recommend fixing the typos and inconsistencies mentioned above.

Alleviation

[Certik]: The client made the recommended changes in commits:

- [56f9adcfac63dd0712cb7fac35e5705df8b15034](#);
- [e61dc01b31e40ce47b432da9ec38fdec5fc3eb5a](#).

PSP-10 | COMPARISON TO BOOLEAN CONSTANT

Category	Severity	Location	Status
Coding Style	● Informational	PegStability.sol (base): 85 , 247 , 259	● Resolved

Description

Boolean constants can be used directly and do not need to be compared to true or false.

```
85         require(isPaused == false, "Contract is paused.");
```

```
247         require(isPaused == false, "PSM is already paused.");
```

```
259         require(isPaused == true, "PSM is not paused.");
```

Recommendation

We recommend removing the equality to the boolean constant.

Alleviation

[certik]: The client made the recommended changes in commit: [da4a91bd65427bb483cbd109041a0d0dde0049a3](#).

PSV-01 | FEE AMOUNT NOT CHECKED IN INITIALIZER

Category	Severity	Location	Status
Coding Style	● Informational	PegStability.sol (06073f3): 91-92	● Resolved

Description

The input `feeIn_` and `feeOut_` of `initialize()` function are not checked to be less than `1000`.

Recommendation

We recommend checking these values are less than `1000`.

Alleviation

[certik]: The client made the recommended changes in commit: [9976024ccf70a447cdac742e4fe3b9c9c79c210d](#).

OPTIMIZATIONS | VENUS - VAI PEG

ID	Title	Category	Severity	Status
PSP-08	Unchecked Blocks Can Optimize Contract	Gas Optimization	Optimization	● Resolved
PSP-09	Inefficient Event Parameter	Gas Optimization	Optimization	● Resolved

PSP-08 | UNCHECKED BLOCKS CAN OPTIMIZE CONTRACT

Category	Severity	Location	Status
Gas Optimization	● Optimization	PegStability.sol (base): 177	● Resolved

Description

In the function `swapStableForVAI`, the check

```
require(vaiMinted + actualTransferAmtInUSD <= vaiMintCap, "VAI mint cap reached.");
```

is performed before incrementing `vaiMinted` by the `actualTransferAmtInUSD`. As the check adds `vaiMinted` and `actualTransferAmtInUSD` it already checks for overflow. Thus it does not need to be checked when incrementing.

Recommendation

We recommend incrementing in an unchecked block to save gas.

Alleviation

[Certik]: The client made the recommended changes in commit: [4120a202fddf3f0bb87cbe6253be1e14987ac9df](#).

PSP-09 | INEFFICIENT EVENT PARAMETER

Category	Severity	Location	Status
Gas Optimization	● Optimization	PegStability.sol (base): 299 , 303	● Resolved

Description

The event `VaiMintCapChanged` uses the storage variable `vaiMintCap`. However, gas can be saved by using `vaiMintCap_`.

Recommendation

We recommend using `vaiMintCap_` instead of `vaiMintCap` to save gas.

Alleviation

[Certik]: The client made the recommended changes in commit: [022f0c657425111690fe11b7995d24d11e464a49](#).

APPENDIX | VENUS - VAI PEG

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
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.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

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.



