



Venus VAIController Upgrade

AUDIT REPORT

Version 1.0.0

Serial No. 2024041800012024

Presented by Fairyproof

April 18, 2024

www.fairyproof.com

01. Introduction

This document includes the results of the audit performed by the Fairyproof team on the Venus VAIController Upgrade project.

Audit Start Time:

April 15, 2024

Audit End Time:

April 18, 2024

Audited Source File's Address:

<https://github.com/VenusProtocol/venus-protocol/pull/467>

Audited Code's Github Repository:

<https://github.com/VenusProtocol/venus-protocol/pull/467/commits>

Audited Code's Github Commit Number When Audit Started:

6b7636bd45184aac804b16e3217764ecb9d1594e

Audited Code's Github Commit Number When Audit Ended:

a5569976c6b88c2fb82f9a9c5343817144b558b4

Audited Source Files:

The source files audited include all the files as follows:

```
contracts/Tokens/VAI/VAIController.sol
```

The goal of this audit is to review Venus's solidity implementation for its VAIController Upgrade function, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

We make observations on specific areas of the code that present concrete problems, as well as general observations that traverse the entire codebase horizontally, which could improve its quality as a whole.

This audit only applies to the specified code, software or any materials supplied by the Venus team for specified versions. Whenever the code, software, materials, settings, environment etc is changed, the comments of this audit will no longer apply.

— Disclaimer

Note that as of the date of publishing, the contents of this report reflect the current understanding of known security patterns and state of the art regarding system security. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk.

The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from offchain sources are not extended by this review either.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit is recommended after the issues covered are fixed.

To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

— Methodology

The above files' code was studied in detail in order to acquire a clear impression of how the its specifications were implemented. The codebase was then subject to deep analysis and scrutiny, resulting in a series of observations. The problems and their potential solutions are discussed in this document and, whenever possible, we identify common sources for such problems and comment on them as well.

The Fairyproof auditing process follows a routine series of steps:

1. Code Review, Including:

- Project Diagnosis

Understanding the size, scope and functionality of your project's source code based on the specifications, sources, and instructions provided to Fairyproof.

- Manual Code Review

Reading your source code line-by-line to identify potential vulnerabilities.

- Specification Comparison

Determining whether your project's code successfully and efficiently accomplishes or executes its functions according to the specifications, sources, and instructions provided to Fairproof.

2. Testing and Automated Analysis, Including:

- Test Coverage Analysis

Determining whether the test cases cover your code and how much of your code is exercised or executed when test cases are run.

- Symbolic Execution

Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.

3. Best Practices Review

Reviewing the source code to improve maintainability, security, and control based on the latest established industry and academic practices, recommendations, and research.

— Structure of the document

This report contains a list of issues and comments on all the above source files. Each issue is assigned a severity level based on the potential impact of the issue and recommendations to fix it, if applicable. For ease of navigation, an index by topic and another by severity are both provided at the beginning of the report.

— Documentation

For this audit, we used the following source(s) of truth about how the token issuance function should work:

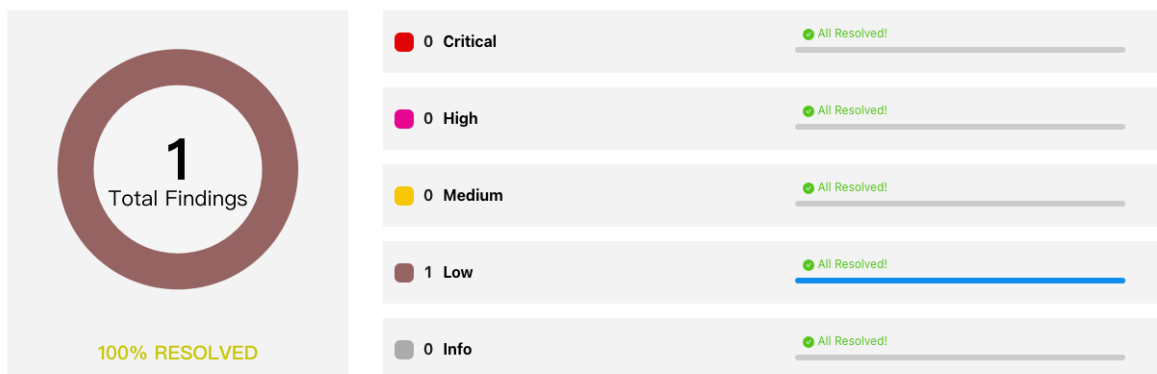
Website: <https://venus.io/>

Source Code: <https://github.com/VenusProtocol/venus-protocol/pull/467>

These were considered the specification, and when discrepancies arose with the actual code behavior, we consulted with the Venus team or reported an issue.

— Comments from Auditor

Serial Number	Auditor	Audit Time	Result
2024041800012024	Fairproof Security Team	Apr 15, 2024 - Apr 18, 2024	Passed



Summary:

The Fairyproof security team used its auto analysis tools and manual work to audit the project. During the audit, one issue of low-severity was uncovered. The Venus team fixed this issue.

02. About Fairyproof

[Fairyproof](#) is a leading technology firm in the blockchain industry, providing consulting and security audits for organizations. Fairyproof has developed industry security standards for designing and deploying blockchain applications.

03. Introduction to Venus

Venus Protocol ("Venus") is an algorithmic-based money market system designed to bring a complete decentralized finance-based lending and credit system onto Ethereum, Binance Smart Chain, opBNB and Arbitrum.

The above description is quoted from relevant documents of Venus.

04. Major functions of audited code

The audited code includes some changes on `VAIController.sol` as follows:

- Fixing the issue of incorrect `repaidAmount` returned by `repayVAIFresh`. This issue would lead to a decrease in the liquidator's profits, thus the liquidator lacks the motivation to carry out the liquidation operation.

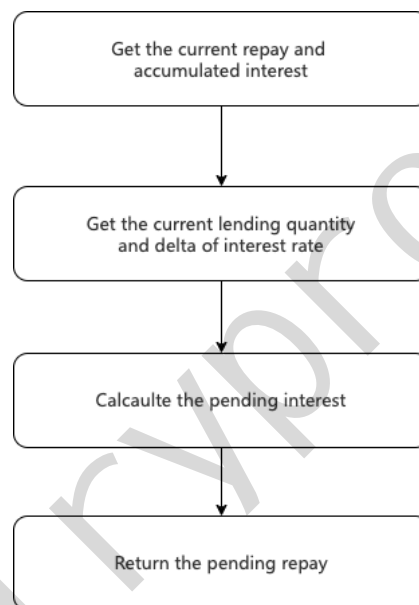
- Adding a `repayVAIBehalf` function to repay on behalf of other borrowers.
- Some misc changes

`VAIController` is a contract responsible for managing the lending, repayment, and liquidation of VAI by users. After borrowing VAI, users are required to pay interest, and each borrowing updates the accumulated interest and interest rate. Users have the option to repay either the entire borrowed VAI or a portion of it along with the corresponding interest. In cases where a user becomes insolvent, other users can initiate the liquidation process.

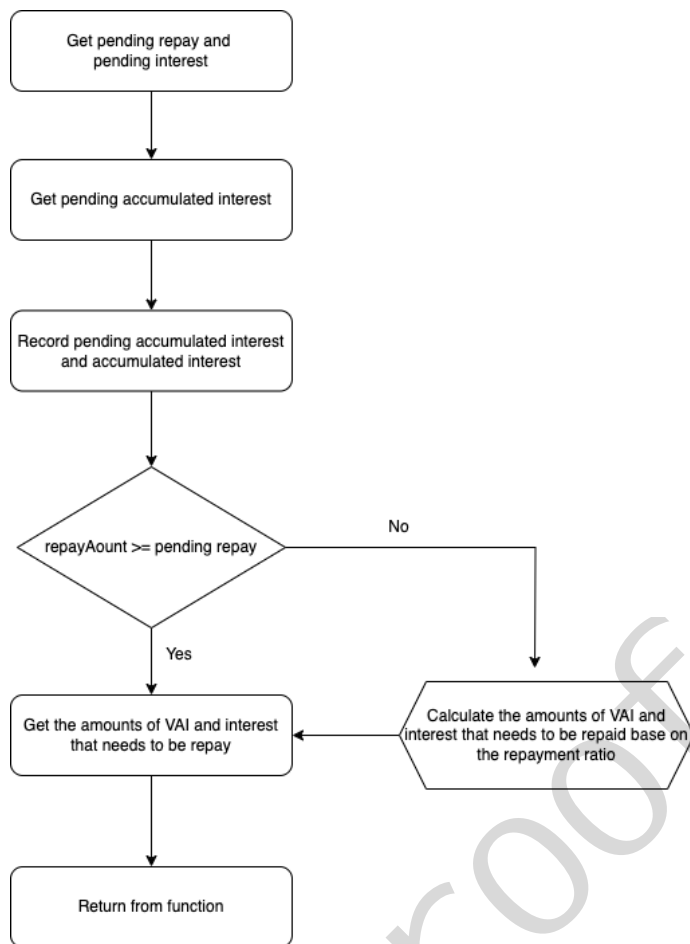
The previous version of the `repayVAIFresh` function did not include the paid interest in its return value. It only included the principal amount borrowed by the user. However, the liquidator needs to repay both the principal and the interest. As a result, the calculated quantity of obtainable collateral becomes less. The new version of the `repayVAIFresh` function has fixed this issue.

Here are the relevant functions:

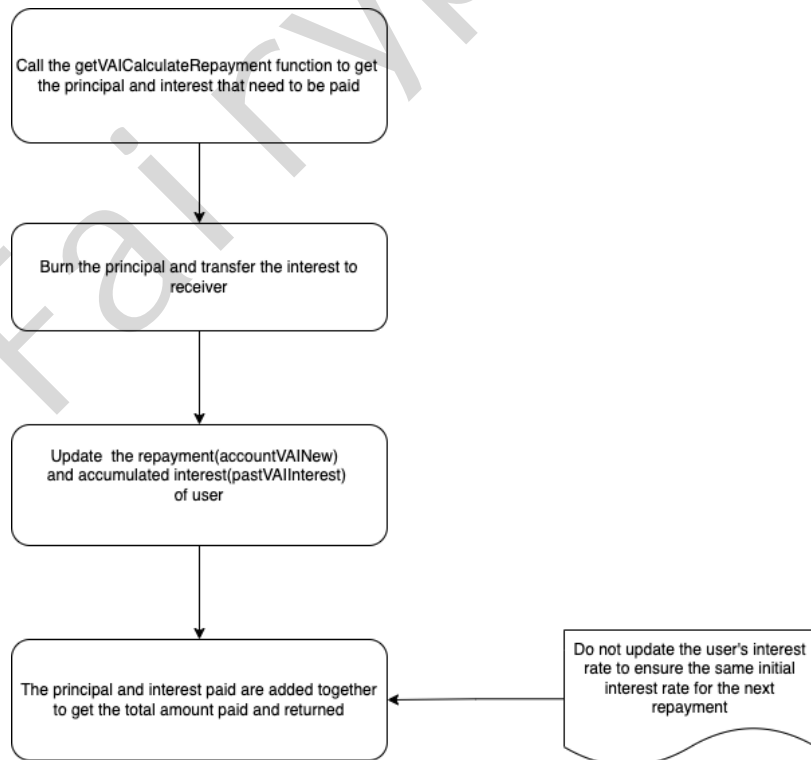
- `getVAIRepayAmount` function:



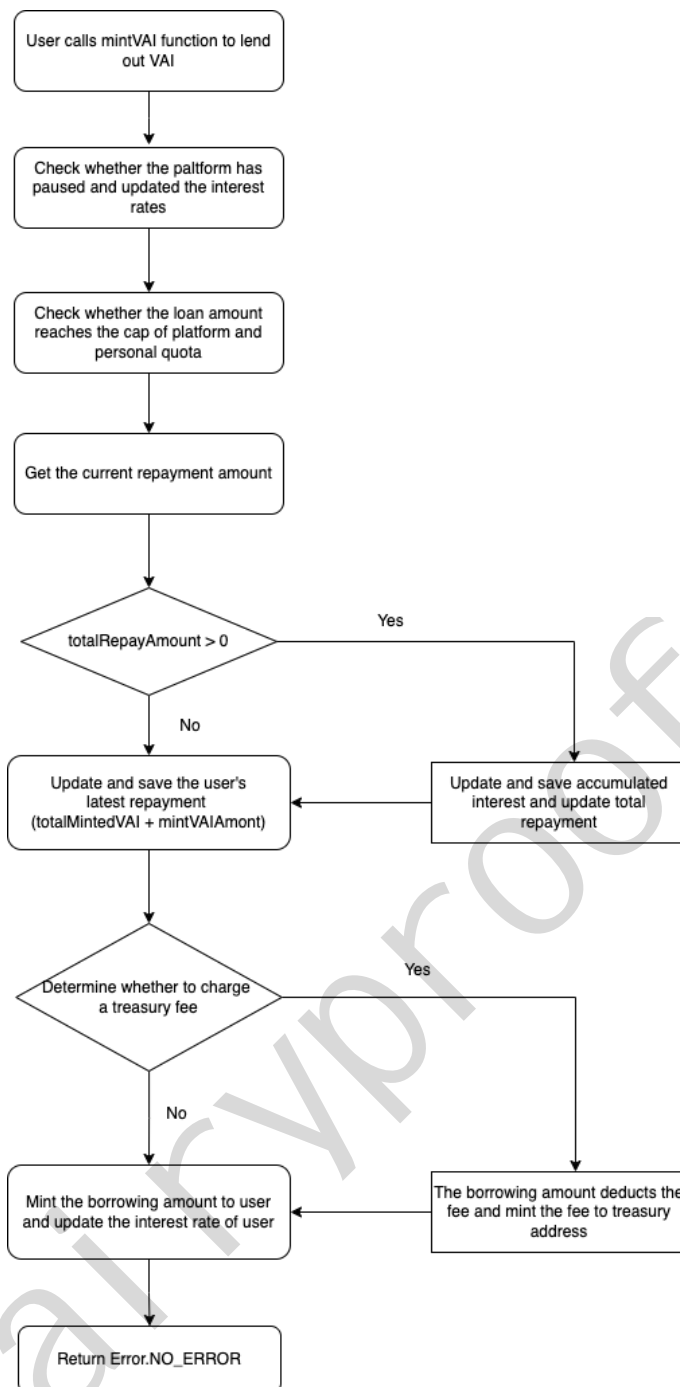
- `getVAICalculateRepayAmount` function



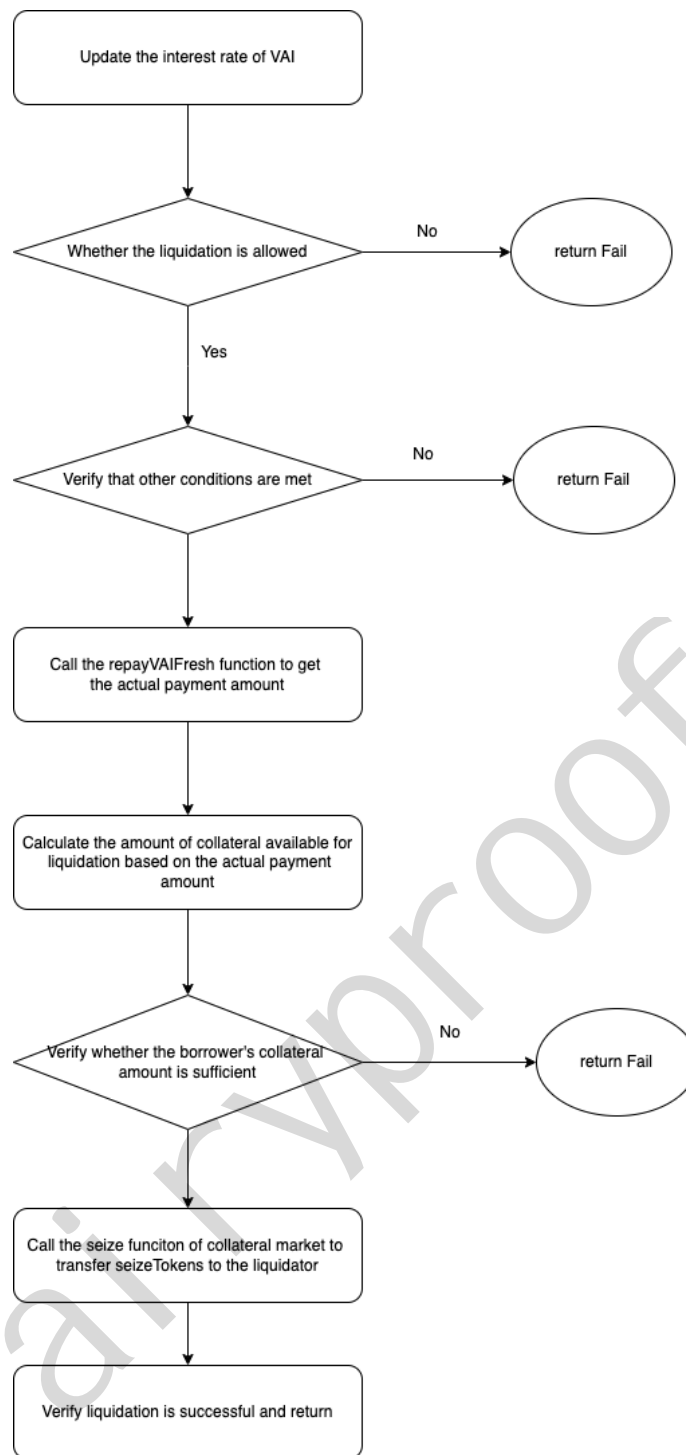
- `repayVAIFresh` function



- `mintVAI` function



- `liquidateVAIFresh` function

**note:**

`VAIController` is designed to be upgradeable. The current upgrade doesn't involve adding/removing state variables and does not involve any existing data issues.

05. Coverage of issues

The issues that the Fairproof team covered when conducting the audit include but are not limited to the following ones:

- Access Control
- Admin Rights
- Arithmetic Precision
- Code Improvement
- Contract Upgrade/Migration
- Delete Trap
- Design Vulnerability
- DoS Attack
- EOA Call Trap
- Fake Deposit
- Function Visibility
- Gas Consumption
- Implementation Vulnerability
- Inappropriate Callback Function
- Injection Attack
- Integer Overflow/Underflow
- IsContract Trap
- Miner's Advantage
- Misc
- Price Manipulation
- Proxy selector clashing
- Pseudo Random Number
- Re-entrancy Attack
- Replay Attack
- Rollback Attack
- Shadow Variable
- Slot Conflict
- Token Issuance
- Tx.origin Authentication
- Uninitialized Storage Pointer

06. Severity level reference

Every issue in this report was assigned a severity level from the following:

Critical severity issues need to be fixed as soon as possible.

High severity issues will probably bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Informational is not an issue or risk but a suggestion for code improvement.

07. Major areas that need attention

Based on the provided source code the Fairyproof team focused on the possible issues and risks related to the following functions or areas.

- Function Implementation

We checked whether or not the functions were correctly implemented.

We didn't find issues or risks in these functions or areas at the time of writing.

- Access Control

We checked each of the functions that could modify a state, especially those functions that could only be accessed by owner or administrator

We didn't find issues or risks in these functions or areas at the time of writing.

- Token Issuance & Transfer

We examined token issuance and transfers for situations that could harm the interests of holders.

We didn't find issues or risks in these functions or areas at the time of writing.

- State Update

We checked some key state variables which should only be set at initialization.

We found one issue, for more details please refer to [FP-1] in "09. Issue description".


- Asset Security

We checked whether or not all the functions that transfer assets were safely handled.
We didn't find issues or risks in these functions or areas at the time of writing.

- Miscellaneous

We checked the code for optimization and robustness.
We didn't find issues or risks in these functions or areas at the time of writing.

08. List of issues by severity

Index	Title	Issue/Risk	Severity	Status
FP-1	State Variable Not Reset	Misc	Low	 Fixed

09. Issue descriptions

[FP-1] State Variable Not Reset

Misc

Low

✓ Fixed

Issue/Risk: Misc

Description:

In function `mintVAI`, there was a following code snippet:

```
uint error = comptroller.setMintedVAIOf(minter, vars.accountMintVAINew);
if (error != 0) {
    return error;
}
```

Here, if the `comptroller` returned an error in setting minter's debt(repayment), the function would return directly.

However, the user's accumulated interest had been updated before this. The relevant code is as follows:

```
(vars.mathErr, pastVAIInterest[minter]) = addUInt(pastVAIInterest[minter],
remainedAmount);
if (vars.mathErr != MathError.NO_ERROR) {
    return failOpaque(Error.MATH_ERROR, FailureInfo.MINT_FEE_CALCULATION_FAILED,
uint(vars.mathErr));
}
```

This is easy to cause data disorder when the `comptroller` rejects the `setMintedVAIOf` calling.

Recommendation:

Consider replacing the relevant code in the `mintVAI` function with:

```
uint error = comptroller.setMintedVAIOf(minter, vars.accountMintVAINew);
// we have to revert upon error since side-effects already happened at this point
require(error == 0, "comptroller rejection");
```

Update/Status:

The Venus team has fixed the issue.

10. Recommendations to enhance the overall security

We list some recommendations in this section. They are not mandatory but will enhance the overall security of the system if they are adopted.

- Consider managing the admin's access control with great care and transferring it to a multi-sig wallet or DAO when necessary.

11. Appendices

- N/A

11.2 External Functions Check Points

- N/A

Fairyproof



<https://medium.com/@FairyproofT>



<https://twitter.com/FairyproofT>



<https://www.linkedin.com/company/fairyproof-tech>



https://t.me/Fairyproof_tech



Reddit: <https://www.reddit.com/user/FairyproofTech>

