VENUS

Venus Security Analysis

by Pessimistic

This report is public

May 3, 2024

Abstract

In this report, we consider the security of smart contracts of Venus project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

Although a security audit is one of the most crucial steps to overall project security, it does not give any guarantees on the security of the code. A single audit cannot be considered enough. We strongly encourage our customers to take additional measures to achieve maximum project security. One can check out our public documentation on how to protect a project beyond the code audit. Besides, a security audit is not investment advice.

On code diff security analysis

Our general policy is that we do not audit code diffs and do the full audits of the codebase instead.

For this project, we did not have a possibility to audit the full codebase, only the pull request. We can not provide our usual quality of work within these limitations, but the security review of this limited scope still increases the confidence level. This approach is not enough on its own, but it may be used as a supplement for the rest of the security measures. It must be explicitly stated that during this work, we assume that the initial codebase is completely secure and the update does not break any system invariants.

Generally, auditing only the updated parts of the code without the team being familiar with the rest of the codebase is often counterproductive and creates a dangerous illusion of proper risk management. Many aspects and security invariants of the system fall out of the scope of such security review. To provide the work of our usual quality standard and the same level of confidence, we need to audit the whole project, which means the full review of the unchanged parts of the code, too, unless we have audited that code in the last six months.

However, in some cases, there is a time or budget constraint on the side of our customer, which makes the full audit impossible. In such cases, we provide a best-effort security analysis within this limited time to increase the confidence level of some limited scope (usually some code update). This analysis can not provide the same quality level as our standard procedure.

Summary

In this report, we considered the security of Venus smart contracts. We described the audit process in the section below.

The initial audit showed one issue of low severity. It does not endanger project security. All the tests passed.

After the initial audit, the codebase was updated. The developers added NatSpec comments. We did not find any new issues.

General recommendations

We have no further recommendations.

Project overview

Project description

For the audit, we were provided with the following pull request:

Venus-Protocol, commit 8cb3def9cf4d44f9956f5f2ea98add98bcedf925.

The scope of the PR included **contracts/Tokens/VAI/VAIController.sol**.

The description of this PR was provided as:

- The public documentation;
- The private document **a3cf2004_7f02_43f1_aee6_4a8d6ad78a1b_VAIController_upgrade_Audit.pdf**, checksum: `6e5c934684630f61d697a944daefb06b669a02981998cd8c2b88015e3fbdca60`.

All 689 tests pass successfully. The code coverage is 84.41%.

The PR includes 437 insertions(+) in the current scope.

Codebase update

After the initial audit, the codebase was updated. For the recheck, we were provided with commit f1a45fd34cd503fe4e3f24551be64e54a2e183cf. In this update, NatSpec comments were added and no new issues were found.

All 689 tests pass successfully. The code coverage is 84.46%.

Audit process

We started the audit on April 24, 2024 and finished on May 3, 2024. It was not a full audit, just the pull request of the scope from the Project Description. Since we did not audit the whole project, there was a big chance to miss bugs. That is why we wrote the Disclaimer.

We inspected the materials provided for the audit, manually analyzed the contract from the scope of the audit and checked its changes. Among other, we verified the following properties of the contracts:

- Whether the refactoring changes are consistent with the previous version of the code;
- Whether all improvements are corresponded to the provided description;
- Whether the logical modifications are correct.

We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts. We also sent the results to the developers in the text file.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

On April 29, 2024 the developers provided us with an updated version of the code. In this update, they completed the NatSpec comments.

We did not find any issues in the provided code and commented on the code changes related to the described issue. Also, we rescanned the contract with Slither, Slitherin and Semgrep tools, the results did not reveal any new issues as the code had insignificant changes. In this version of the code, all tests pass.

Finally, we updated the report.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

The audit showed no issues of medium severity.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Unhandled the comptroller's zero address (commented)

The `mintVAI` and `_repayVAI` functions do not revert or return an error if the `comptroller` value is equal to `address(0)`. They return default values. At the same time, there are the following NatSpecs: "0 on success, otherwise an error code" at line 99 and "An error code (`0=success`, otherwise a failure, see **ErrorReporter.sol**), and the actual repayment amount" at line 187.

This means that anyone calling these functions cannot properly handle the returned values since, according to NatSpecs, it looks like a success. However, in the `_repayVAI` function it can be handled by checking the second returned value.

> *The developers have updated NatSpec comments at the commit [70798a28556c344e7a18e956a646a72442ff9e85](), now it is in line with the code. Meanwhile, the `mintVAI` and `_repayVAI` functions may still return `0` error code, requiring anyone calling these functions to separately handle the return value of `0` in case `comptroller == address(0)`.*

This analysis was performed by Pessimistic:

Daria Korepanova, Senior Security Engineer
Evgeny Bokarev, Junior Security Engineer
Konstantin Zherebtsov, Business Development Lead
Irina Vikhareva, Project Manager
Alexander Seleznev, Founder

May 3, 2024