Venus Prime Audit

OPENZEPPELIN SECURITY | OCTOBER 10, 2023

Security Audits

Table of Contents

Summary

Type
: DeFi

Timeline
: From 2023-08-21
: To 2023-09-13

Languages
: Solidity

Total Issues
: 14 (12 resolved)

Critical Severity Issues
: 1 (1 resolved)

High Severity Issues
: 0 (0 resolved)

Medium Severity Issues
: 2 (2 resolved)

Low Severity Issues
: 5 (4 resolved)

Notes & Additional Information
: 6 (5 resolved)

Scope

We audited the VenusProtocol/venus-protocol repository at commit f31a054. More specifically, we audited pull request #196.

```
contracts
├── Comptroller
│   ├── ComptrollerStorage.sol
│   └── Diamond
│       └── facets
```

```
│   ├── Prime
│       ├── IPrime.sol
│       ├── Prime.sol
│       ├── PrimeStorage.sol
│       └── libs
│           ├── Scores.sol
│           ├── FixedMath.sol
│           └── FixedMath0x.sol
│
└── XVSVault
    ├── XVSVault.sol
    └── XVSVaultStorage.sol
```

System Overview

Prime is a reward distribution smart contract system, tailored specifically to reward its holders (often referred to as "Prime holders"). The distribution of rewards is contingent upon a calculated score derived from an individual's staking activity and their associated balance in the Venus lending protocol.

Prime Token Qualification

In order to qualify for the Prime token and consequently partake in its reward system, a user must fulfill the criterion of staking no fewer than 1,000 XVS. This stake should be maintained for a minimum duration of 90 days.

Reward Distribution

All incoming revenue to the Prime system is distributed amongst Prime holders. This distribution is proportional, based on the calculated scores of each individual holder.

Score Calculation

The scoring function of the Prime system is expressed as:

$$xvs^{\alpha} * capital^{(1-\alpha)}$$

and their `vToken` balance up to a certain cap.

- "**α**" adjusts the weight of XVS compared to "capital" for score calculation purposes.

Interactions With External Systems

Prime closely interacts with the Venus lending protocol. The borrowed balance from Venus, as well as the `vToken` balance, play crucial roles in determining an individual's score, and thereby their reward from the Prime system.

`XVSVault` and the Comptroller's `PolicyFacet` were modified by adding hooks to alert rime on balance changes of staked XVS and `vToken` , respectively.

Adjustable Parameters

The Prime system has the ability to modify the `alpha` , `rewardMultiplier`, and `borrowMultiplier` parameters after deployment. The `rewardMultiplier` and `borrowMultiplier` set a cap for the amount of `VTokens` and borrowed balance respectively, at which point rewards stop increasing relative to the amount borrowed or loaned.

Security Model and Trust Assumptions

We trust the Protocol Share Reserve (PSR) to handle all accounting functionalities correctly. The system also interacts with the broader lending protocol by reading user’s vToken and borrowed balances which are used to compute scores.

Privileged Roles

Minimal modifications were made to the access control manager system. This system has been nearly unmodified. Therefore, the assumptions are nearly identical to those listed in the published Venus Protocol Oracles report. The `prime.sol` contract has an owner address that can set the access controller address. This controller can then set permissions for the following privileged functions:

- Update the alpha parameter used in the score formula
- Update the multipliers used in the score formula

Critical Severity

`updateAlpha` and `updateMultipliers` Break Rewards Accounting and Could Make Rewards Pool Insolvent

Every time interest is accrued, the total rewards are divided by the total sum of the user's scores to get the rewards entitled per score. This assumes that the total score for each user is the same when interest is collected as when it is calculated. This invariant is usually maintained as `executeBoost` collects the interest from users before any actions that change their XVS or VToken balances (which play a role in score calculation).

However, `updateAlpha` and `updateMultipliers` update the total interest for every market but do not first collect the interest for every individual user. This means that when users collect interest after the alpha or multiplier update, the new score is multiplied by the interest accrued since the last time `executeBoost` was called. However, that score is different from the score used to calculate the interest at the time. Additionally, the `sumOfScores` now does not equal the `sumOfScores` during interest calculation.

If the update increases the sum total of the user's scores, there would not be enough funds in the reserve to pay out all the accounted-for reward amounts. This would mean that users who collect interest early on will get a larger reward than they should, while later claimants will have zero rewards as the reward pool will become insolvent.

Update: *Resolved in pull request #196 at commit f5e3221.The Venus team stated:*

> *With this change, we'll always accrue interest before updating the user score, so the rewards will be allocated using the right score*

Medium Severity

Maximum XVS Staked Does Not Match the Documentation

The documentation states:

> The maximum XVS cap taken into account when calculating the score of a user is 100,000 XVS.

Update: *Resolved in pull request #196 at commit 860c959.*

Irrevocable Token Cannot Be Minted to Revocable Token Holder

If the Venus team attempts to mint an irrevocable token for a user already holding a revocable Prime token, the minting process fails. This is due to the internal function `_mint`, which requires that the recipient must not hold any form of Prime token, whether it is revocable or irrevocable.

Given that irrevocable Prime tokens are more desirable as they remain even when the user unstakes, this behavior suggests an unintended restriction in the minting process.

Update: *Resolved at commit a19e29c, pull request #196 at commit b98dc82.*

Low Severity

Missing Check for Setting Alpha Values

`alphaDenominator` should never exceed `alphaNumerator`, as specified by the audit documentation as well as this error which would be thrown in the `calculateScore` function:

```
assert(alphaNumerator <= alphaDenominator);
```

Consider ensuring that `alphaNumerator <= alphaDenominator` in the `initialize` and `updateAlpha` functions.

Update: *Resolved at commit f7d463b, at commit a7e913f, at commit 5a04aa0.*

Missing Docstrings

Throughout the codebase, there are several parts that do not have docstrings:

- Line 572 in `Prime.sol`
- Line 105 in `Prime.sol`

Consider thoroughly documenting all functions (and their parameters) that are part of any contract's public API. Functions implementing sensitive functionality, even if not public, should be

2abb751.

Unsafe Typecasting in `uintMul()` and `uintDiv()`

The divisor argument of `uintDiv` and the multiplier argument of `uintMul` in the `FixedMath` library are both signed integers.

The results of multiplication and division operations involving these arguments will therefore be signed, and the results of both the `uintDiv` and `uintMul` functions are downcast to `uint256`.

While this may produce the correct results when the multiplier and divisor arguments are properly-encoded fixed-point numbers, it will produce erroneous results when the high bit of either of these arguments is set.

Consider adding a check to ensure that the multiplier argument of `uintMul` and the divisor argument of `uintDiv` are well-formed fixed-point numbers.

Update: *Resolved at commit 61ad1e7.*

Old Floating Pragma Version

Pragma directives should be fixed to clearly identify the Solidity version with which the contracts will be compiled.

The file `IPrime.sol` has the `solidity ^0.5.16` floating pragma directive which is an outdated Solidity version.

Consider using the same fixed pragma version used in `Prime.sol`.

Update: *Acknowledged, not resolved. The Venus team stated:*

> *It is done so that our core pool contracts can import the interface.*

Missing Check that `PrimePoolId` Is a Valid `poolId`

have this check.

Consider adding a check in `setPrimeToken` to ensure that it matches a pre-existing `poolId`.

Update: *Resolved at commit ad5b11a.*

Notes & Additional Information

Non-Explicit Imports Are Used

The use of non-explicit imports in the codebase can decrease the clarity of the code, and may create naming conflicts between locally defined and imported variables. This is particularly relevant when multiple contracts exist within the same Solidity files or when inheritance chains are long.

Throughout the codebase, global imports are used:

- Line 3 of `Prime.sol`
- Line 4 of `Prime.sol`
- Line 5 of `Prime.sol`
- Line 6 of `Prime.sol`
- Line 6 of `FixedMath.sol`
- Line 7 of `FixedMath.sol`
- Line 5 of `Scores.sol`
- Line 6 of `Scores.sol`
- Line 9 of `XVSVault.sol`

Following the principle that clearer code is better code, consider using named import syntax (`import {A, B, C} from "X"`) to explicitly declare which contracts are being imported.

Update: *Resolved at commit b2fa6d5, and commit adb8535.*

Using `int/uint` Instead of `int256/uint256`

- Line 9 of `Prime.sol`
- Line 10 of `Prime.sol`
- Line 11 of `Prime.sol`
- Line 118 of `Prime.sol`
- Line 185 of `Prime.sol`
- Line 190 of `Prime.sol`
- Line 208 of `Prime.sol`
- Line 223 of `Prime.sol`
- Line 249 of `Prime.sol`

In favor of explicitness, consider replacing all instances of `int/uint` with `int256/uint256`.

Update: *Resolved at commit e9269c7.*

Unused State Variable

The `MAXIMUM_BPS` state variable in the `PrimeStorageV1_contract` is never used.

To improve the overall clarity, intentionality, and readability of the codebase, consider removing any unused state variables.

Update: *Resolved. The Venus team stated:*

> *We are finally using it in the function `_calculateUserAPR`. See commit ec2f4fd.*

Lack of SPDX License Identifiers

Throughout the codebase, there are files that lack SPDX license identifiers. For instance:

- `IPrime.sol`
- `Prime.sol`
- `PrimeStorage.sol`

To avoid legal issues regarding copyright and follow best practices, consider adding SPDX license identifiers to files as suggested by the Solidity documentation.

Providing a specific security contact, such as an email or ENS, within a smart contract significantly simplifies the process for individuals to communicate if they identify a vulnerability in the code. This practice proves beneficial as it permits the code owners to dictate the communication channel for vulnerability disclosure, eliminating the risk of miscommunication or failure to report due to a lack of knowledge on how to do so. Additionally, if the contract incorporates third-party libraries and a bug surfaces in these, it becomes easier for the creators of those libraries to make contact, inform the code owners about the problem, and provide mitigation instructions.

Neither of the contracts in scope has a security contact.

Consider adding a NatSpec comment on top of the contracts' definitions with a security contact. Using the `@custom:security-contact` convention is recommended as it has been adopted by the OpenZeppelin Wizard and the ethereum-lists.

Update: *Resolved at commit 405f962. The Venus team stated:*

> *We have added a link to our github repo. At the end of the readme file in that repo we say this: For any concerns with the protocol, open an issue or visit us on Telegram to discuss. For security concerns, please contact the administrators of our telegram chat.*

Multiple Instances of Missing Named Parameters in Mappings

Since Solidity 0.8.18, developers can utilize named parameters in mappings. This means mappings can take the form of `mapping(KeyType KeyName? => ValueType ValueName?)`. This updated syntax provides a more transparent representation of the mapping's purpose.

Throughout the codebase, there are multiple mappings without named parameters:

- The `tokens` state variable in the `PrimeStorageV1` contract.
- The `stakedAt` state variable in the `PrimeStorageV1` contract.
- The `markets` state variable in the `PrimeStorageV1` contract.
- The `interests` state variable in the `PrimeStorageV1` contract.
- The `isScoreUpdated` state variable in the `PrimeStorageV1` contract.

Consider adding named parameters to the mappings to improve the readability and maintainability of the code.

Update: *Acknowledged, not resolved. The Venus team stated:*

> *We are using solidity* `0.8.13`*, and named parameters were included in* `0.8.18`

Conclusion

The addition of Prime tokens had a relatively low impact on the existing codebase. The audit uncovered 1 critical-severity and 2 medium-severity issues. Several recommendations were proposed to follow best practices.

Related Posts

Beefy Zap Audit

BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

Security Audits

OpenBrush Contracts Library Security Review

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Security Audits

Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits