DeHacker

Code Security Assessment

Venus - RewardsDistributor

July 14 th, 2023

Contents

CONTENTS 1
SUMMARY 2
ISSUE CATEGORIES 3
OVERVIEW 4
PROJECT SUMMARY 4
VULNERABILITY SUMMARY 4
AUDIT SCOPE 5
FINDINGS 6
MAJOR 7
RDR-04|CENTRALIZED CONTROL OF CONTRACT UPGRADE 7
DESCRIPTION 7
RECOMMENDATION 7
MAJOR 9
RDR-05|CENTRALIZATION RISKS 9
DESCRIPTION 9
RECOMMENDATION 10
Medium 10
RDR-01|POTENTIAL DENIAL OF SERVICE ATTACK 11
DESCRIPTION 11
RECOMMENDATION 11
MINOR 12
RDR-02|EXCESS REWARDS GIVEN IF REWARDS RESTARTED 12
DESCRIPTION 12
RECOMMENDATION 12
Informational 13
RDR-03|TYPOS AND INCONSISTENCIES 13
DESCRIPTION 13
RECOMMENDATION 13
DISCLAIMER 14
APPENDIX 15
ABOUT 16

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

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

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	Venus - RewardsDistributor
Platform	Binance Smart Chain
Website	https://venus.io/
Type	DeFi
Language	Solidity

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	1
Major	2	2	0	1	0	0
Medium	1	0	0	0	0	1
Minor	1	0	0	1	0	1
Informational	1	0	0	0	0	1
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
RDR	RewardsDistributor.sol	e628742f940d1c9e8c5058d3d2d9497cf244b2e229616fb8e481a44e56674821
PLL	PoolLens.sol	d723429b6dea59c2380d9abda3a449333b84e793da5bd692a74f8de3c2b8fdba

Findings

ID	Category	Severity	Status
RDR-04	Centralization	Major	Mitigated
RDR-05	Centralization	Major	Mitigated
RDR-01	Logical Issue	Medium	Resolved
RDR-02	Logical Issue	Minor	Resolved
RDR-03	Inconsistency	Informational	Resolved

MAJOR

RDR-04|CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	Major	RewardsDistributor.sol(baseRewards): 29	Mitigated

Description

RewardsDistributor is an upgradeable contract. The owner can upgrade the contract without the community'scommitment. If an attacker compromises the account, he can change the implementation of the contract and drain tokensfrom 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 atime-lock and a multi-signature (,) wallet can be used to prevent a single point of failure due to a private keycompromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrateto a new implementation contract.

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

Short Term:
A combination of a time-lock and a multi signature (,) wallet mitigate the risk by delaying the sensitive operation andavoiding 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 keycompromised;
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.

Recommendation

Long Term:
A combination of a time-lock on the contract upgrade operation and a DAO for controlling the upgrade operation mitigate thecontract 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 userinvolvement;
AND
A medium/blog link for sharing the time-lock contract, multi-signers addresses, and DAO information with thecommunity.

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 shallmake a decision based on the current state of their project, timeline, and project resources.

MAJOR

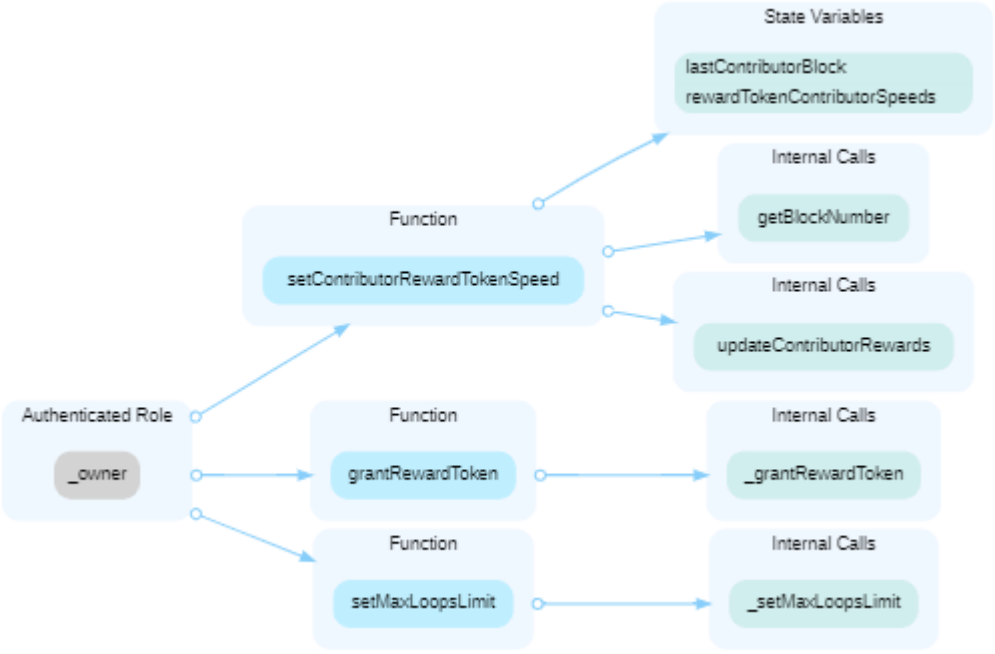
RDR-05|CENTRALIZATION RISKS

Category	Severity	Location	Status
Centralization	Major	RewardsDistributor.sol (baseRewards): 155, 194, 202, 212, 218, 233, 256, 273, 287, 30	Mitigated

Description

In the contract RewardsDistributor the role _owner has authority over the functions shown in the diagram below. Anycompromise to the _owner account may allow the hacker to take advantage of this authority and do the following:

Change the contributor reward token speed to any value;
Change the max loops, which limits that amount of vToken that claimRewardToken() can be called on at onetime;
Grant any amount of reward tokens, provided enough are held by the contract, to any user.



In the contract RewardsDistributor the role DEFAULT_ADMIN_ROLE of the Access Control Manager can grant addressesthe privilege to call the following functions:
setRewardTokenSpeeds()
setLastRewardingBlocks()

Any compromise to the DEFAULT_ADMIN_ROLE or these privileged functions may allow the hacker to take advantage of thisauthority and do the following:
change the reward token speed to any value;
set the last rewarding blocks to either stop rewards early or to lengthen the amount of blocks rewards are given for.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level ofdecentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefullymanage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommendcentralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accountswith enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that wouldalso mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:
Timelock and Multi sign (2/3,3/5) 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 keycompromised;
AND
A medium/blog link for sharing the timelock contract and multi-signers addresses information with the publicaudience.

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 publicaudience.

Permanent:
Renouncing the ownership or removing the function can be considered fully resolved.
Renounce the ownership and never claim back the privileged roles.
ORRemove the risky functionality.

MEDIUM

RDR-01|POTENTIAL DENIAL OF SERVICE ATTACK

Category	Severity	Location	Status
Logical Issue	Medium	RewardsDistributor.sol (baseRewards): 514~516, 518, 549~551, 553	Resolved

Description

Assume that for a market the supplyState.lastRewardingBlock is currently 300, the current block is 200, andsupplyState.block = 199 .
A entity that has access to setLastRewardingBlocks() updates supplyState.lastRewardingBlock to be 100 forthis market.
Any other user then attempts an action that will call the preMintHook() , preRedeemHook() , preSeizeHook() , orpreTransferHook() of the comptroller.
These hooks will then call rewardsDistributor.updateRewardTokenSupplyIndex which will then perform thefollowing logic:

```
        uint32 blockNumber = safe32(getBlockNumber(), "block number exceeds 32
bits");

        if (supplyState.lastRewardingBlock > 0 && blockNumber >
supplyState.lastRewardingBlock) {
            blockNumber = supplyState.lastRewardingBlock;
        }

        uint256 deltaBlocks = sub_(uint256(blockNumber),
uint256(supplyState.block));
```

As the supplyState.lastRewardingBlock was set to be 100 , which is less than the current block number of200 , blockNumber will be set to 100 .

Thus deltaBlocks will take 100 minus the supplyState.block = 199 and revert due to underflow.

This demonstrates how a user with access to setLastRewardingBlocks() can perform a denial of service. For examplethis could be done to prevent accounts from becoming liquidated, which could cause the protocol to incur bad debt.

Similarly this can be done for borrowState.lastRewardingBlock to perform a denial of service on actions that call thecomptrollers preBorrowHook() or preRepayHook() .

Recommendation

We recommend checking that the input supplyLastRewardingBlock and borrowLastRewardingBlock are greater than thecurrent block in the _setLastRewardingBlock () function.

MINOR

RDR-02|EXCESS REWARDS GIVEN IF REWARDS RESTARTED

Category	Severity	Location	Status
Logical Issue	Minor	RewardsDistributor.sol (baseRewards): 514~518	Acknowledged

Description

If the last reward blocks for a market are reached and the last reward blocks are updated to restart giving rewards, then anyuser that remained a borrower or supplier in the market will receive rewards for the time the rewards were not active.

Recommendation

We recommend ensuring that if the lastRewardingBlock is reached and rewards are to be restarted, that no rewards willbe given for the blocks between the lastRewardingBlock and the block the lastRewardingBlock is updated to restart rewards.

INFORMATIONAL

RDR-03|TYPOS AND INCONSISTENCIES

Category	Severity	Location	Status
Inconsistency	Informational	RewardsDistributor.sol (baseRewards): 247	Resolved

Description

In the comments above setLastRewardingBlocks() , it states "The markets whose REWARD TOKEN rewarding block toupdate". When it is more accurate to say "The markets whose REWARD TOKEN last rewarding block to update".

Recommendation

We recommend fixing the typos/inconsistencies mentioned above.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. 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. Blockchain technology remains under development and is subject to unknown risks and flaws. 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. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. 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. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

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.

Appendix

Finding Categories

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.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

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.

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.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum
Cosmos
Eos
Substrate

TECH STACK

Python
Solidity
Rust
c++

CONTACTS

https://dehacker.io
https://twitter.com/dehackerio
https://github.com/dehacker/audits_public
https://t.me/dehackerio
https://blog.dehacker.io/

DeHacker

July 2023