Neuron Option Strategy Security Analysis

by Pessimistic

This report is public

August 15, 2022

Abstract

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

Disclaimer

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

Summary

In this report, we considered the security of Neuron Option Strategy smart contracts. We performed our audit according to the procedure described below.

The initial audit showed several issues of medium severity: ERC20 standard violation, Unprotectected asset address, Inaccurate token unwrap and Potential revert with safeApprove. Also, several low-severity issues were found. The overall code quality is above average.

After the initial audit, the codebase was updated. In this update most of the issues were fixed, however, one new issue was found.

With the next codebase update this issue was fixed.

General recommendations

We recommend adding more tests and providing extensive documentation for the project. We also recommend implementing CI to calculate test code coverage, and analyze code with linters and security tools.

Project overview

Project description

For the audit, we were provided with Neuron Option Strategy project on a public GitHub repository, commit 9e9f82f041881b8c109fff5598c959707a54891c.

The scope of the audit includes the contracts from the repository, besides the **tests** folder.

The project functionality is described via NatSpec comments in the code.

The total LOC of audited sources is 2872.

All 194 tests pass successfully. The code coverage is not measured.

Codebase update

After the initial audit, the codebase was updated. For the recheck, we were provided with commit 48f5cdf28c7d43556c6644846fdc84afbf683858. This update includes fixes for most of the issues mentioned in the initial audit. One new medium severity issue was discovered in this version.

Codebase update #2

Later, we were provided with commit 7bae57bae52c593b25ad5032fc1c54461bb93a11 which includes a fix for the mentioned issue. No new issues were discovered.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
  - We scan the project's codebase with the automated tool Slither.
  - We manually verify (reject or confirm) all the issues found by the tool.
- Manual audit
  - We manually analyze the codebase for security vulnerabilities.
  - We assess the overall project structure and quality.
- Report
  - We reflect all the gathered information in the report.

Inter alia, we verify that:

- Neuron Option Strategy properly implements intended functionality, especially for deposit and withdraw functions.
- Neuron Option Strategy correctly integrates with Neuron Pools, Neuron options parts, and the Gnosis Auction contracts.
- The project is resistant to reentrancy, flashloan, and front-running attacks.

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 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.

M01. ERC20 standard violation (addressed)

The return value of ERC20 token transfer is not handled at line 573 of the **NeuronCollateralVault** contract.

ERC-20 standard states:

> *Callers MUST handle `false` from returns `(bool success)`. Callers MUST NOT assume that `false` is never returned!*

However, returned value from `transfer` is not checked.

This issue is also on lines 422 and 367 of the **NeuronCollateralVault** contract.

> *Comment from the developers: We do not handle revert value in this transfer since the tokens which are transferred are written by us and follow OpenZeppelin ERC20 standard, which has strict rules and revert transaction in case of failed transfer.*

M02. Unprotectected asset address (fixed)

In the `withdrawIfDisabled`, `completeWithdraw` and `withdrawInstantly` functions of the **NeuronCollateralVault** contract the value of the argument `_withdrawToken` is not verified. This allows a possibility of passing a malicious contract as an argument. Note, that in this case all the user shares will burn, however, the asset transferred might not be from the list of allowed assets.

We recommend implementing the same token verification mechanism as in the case of deposit (utilizing `allowedDepositTokens` variable).

The issue has been fixed and is not present in the latest version of the code.

M03. Inaccurate token unwrapping (fixed)

All the withdraw functions of the **NeuronCollateralVault** contract utilize the `unwrapNeuronPool()` function (lines 455, 485 and 534). Note, that the exact amount of unwrapped tokens might differ from requested `withdrawAmount`. However, the contract does not handle the return value of the `unwrapNeuronPool` call.

We recommend passing the return value of the `unwrapNeuronPool()` function to the call of `transferAsset()` (lines 456, 486 and 535) to ensure the correctness of token transfer.

The issue has been fixed and is not present in the latest version of the code.

M04. Potential revert with safeApprove (fixed)

Incorrect usage of the `safeApprove()`:

- at line 57 in the `startAuction` function of **GnosisAuction** contract
- at line 59 in the `wrapToNeuronPool` function of **NeuronPoolUtils** contract
- at line 22 in the `safeApproveNonCompliant` function of **SupportsNonCompliantERC20** contract

We recommend setting the value to zero and then calling `safeApprove` with the corresponding value.

The issue has been fixed and is not present in the latest version of the code.

M05. Revert on token deposit (fixed)

In the case of non-ether deposit in the `_depositWithToken` function, the **NeuronCollateralVault** contract calls to the `deposit` function of `_collateralToken` address with ether transfer (`{value: _amount}`) at line 276. Considering the previous checks, this `_amount` is required to be non-zero positive integer, while the actual `msg.value` of transaction is required to be zero. Thus, such transaction shall be reverted.

We recommend fixing this bug by sending the actual value in case of ether transaction.

The issue has been fixed and is not present in the latest version of the code.

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. Code duplication (fixed)

We recommend extracting repetitive code for token unwrapping and asset transfer in lines 452-457, 482-487, 531-536, 564-566 and 635-636 of **NeuronCollateralVault** into a private function.

The issue has been fixed and is not present in the latest version of the code.

L02. Pool integration (fixed)

`wrapToNeuronPool` function should utilize `.deposit` return value at line 66 of **NeuronPoolUtils** contract.

The issue has been fixed and is not present in the latest version of the code.

L03. Unused import (fixed)

Unused imports at lines 8 and 10 in **CollateralVaultLifecycle**.

The issue has been fixed and is not present in the latest version of the code.

L04. Redundant variable declaration (fixed)

The `PLACEHOLDER_UINT` variable is defined both in the **Vault** contract at line 16 and in the **ShareMath** library at line 10. It is sufficient to utilize this variable just from the Vault contract.

The issue has been fixed and is not present in the latest version of the code.

L05. Type miscast (fixed)

The variable `lastQueuedWithdrawAmount` is defined as `uint256` in the **NeuronCollateralVaultStorage** contract. However, it is explicitly casted to `uint128` at line 498 in `completeWithdraw` function of **NeuronCollateralVault** contract.

The issue has been fixed and is not present in the latest version of the code.

L06. Discrepancy with documentation (fixed)

The value of the DELAY constant is set to zero at line 39 in the **NeuronThetaVault** contract, however the NatSpec states it should be set to 15 minutes.

The issue has been fixed and is not present in the latest version of the code.

L07. Redundant protection (fixed)

The `nonReentrant` modificator is not actually necessary for the `initRounds` function of the **NeuronCollateralVault** contract. This also applies to the `setStrikePrice` function of the **NeuronThetaVault** contract.

The issue has been fixed and is not present in the latest version of the code.

L08. Missing zero-check (fixed)

In the `initialize` function of the **NeuronCollateralVault** contract at line 137 `allowedDepositTokens` mapping is initialized. Note that there is no protection against setting zero address here.

The issue has been fixed and is not present in the latest version of the code.

Notes

N01. Out of scope functionality

The contracts integrate with `OptionsPremiumPricer` and utilize a volatility oracle. Both of them lie out of the scope of this audit.

N02. Inconsistent auction integration (fixed)

The **VaultLifecycle** library inlucdes methods to start and finish an auction (`startAuction` and `settleAuction` functions). However, the project utilizes only the first function. Thus, there is functionality to start the auction but no functionality to end it.

Note that the function `claimAuctionONtokens` of the **GnosisAuction** library is not utilized as well.

The issue has been fixed and is not present in the latest version of the code.

N03. Dependence on an external actor (fixed)

Note that the `WETH` balance of **NeuronCollateralVault** contract is not handled inside the project, so the ether withdraw functionality depends on maintaining a sufficient `WETH` balance.

> *Comment from the developers: Using and passing `WETH` address was removed. All the functionalaty regarding `WETH` is now moved to **NeuronPools** the same as other collateral ERC20 tokens. Also added special address for ETH deposits and utilizing of sending `value` in transfers in deposit and withdraw functions.*

The issue has been fixed and is not present in the latest version of the code.

N04. Unused code (fixed)

The project implements integration with Uniswap (the `swap` function of the **VaultLifecycle** library); however, it is never utilized. Besides, the **Path** and the **DateTime** libraries are also never used in the codebase.

The issue has been fixed and is not present in the latest version of the code.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Vladimir Tarasov, Security Engineer
Vladimir Pomogalov, Security Engineer
Irina Vikhareva, Project Manager

August 15, 2022