CERTIK

Nerve Finance

Core Contracts

Security Assessment

April 6th, 2021

Audited By:
Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org
Reviewed By:
Camden Smallwood @ CertiK
camden.smallwood@certik.org

Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Nerve Finance - Core Contracts
Description	A SushiSwap and Saddle Finance based DeFi implementation
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. 64fbeaf95c67641e2650f35415634014975102bf 2. 01e524a8b91c4867b3a8dbf4dbc959878c16c5a1

Audit Summary

Delivery Date	April 6th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	March 24th, 2021 - March 25th, 2021

Vulnerability Summary

Total Issues	7
Total Critical	0
Total Major	0
Total Medium	0
Total Minor	2
Total Informational	5

Executive Summary

We were tasked with auditing the codebase of the Nerve Finance team composed of multiple contracts adapted from various protocols such as SushiSwap and SaddleFinance.

The contract adjustments were minimal and mostly related to adjustments in the way the contracts are deployed, the naming conventions they utilize or a new functionality in the form of a fee.

The `NerveToken`, `xNerve` and `MasterMind` contracts derive from the SushiSwap core implementation with adjustments to the `MasterMind` containing a setter for the reward per block as well as the removal of its migration functionality. The `NerveToken` itself also became burnable in contrast to its parent implementation.

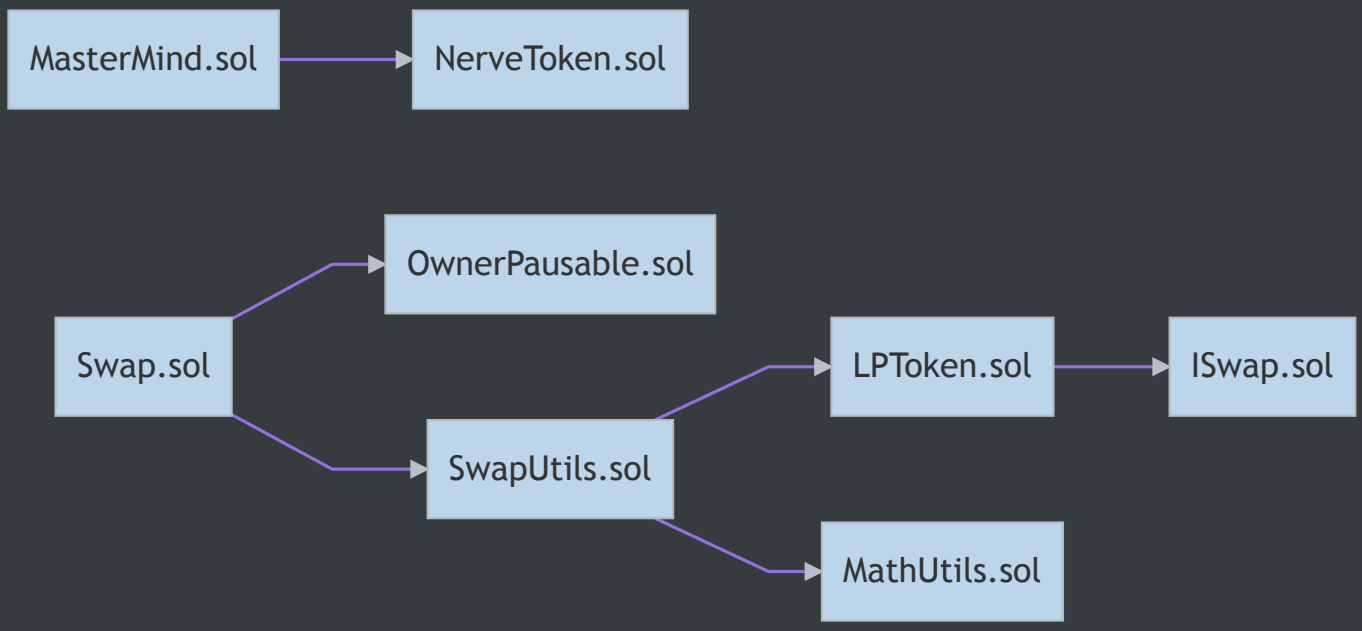
The `LPToken`, `MathUtils`, `OwnerPausable`, `Swap` and `SwapUtils` contracts derive from the SaddleFinance core implementation with adjustments to the swap contracts introducing a deposit fee that is acquired during liquidity provision. This fee is subtracted from the final minted amount and does not otherwise affect the sane operation of the contracts.

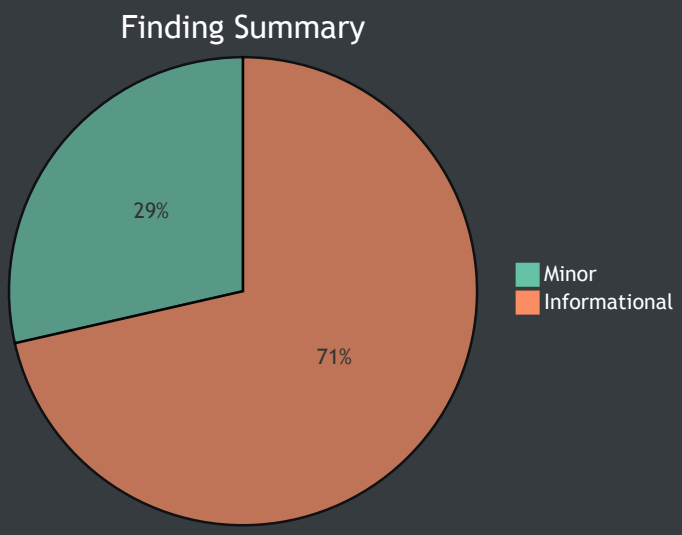
Overall, the changes made were done so conforming to the original style guidelines and security principles of the base implementations and have not introduced any vulnerability to the codebase. We noted a few optimizations that can be carried out as well as certain sanitizations that should be applied to render the contracts secure.

Files In Scope

ID	Contract	Location
LPT	LPToken.sol	LPToken.sol
MMD	MasterMind.sol	MasterMind.sol
MUS	MathUtils.sol	MathUtils.sol
NTN	NerveToken.sol	NerveToken.sol
OPE	OwnerPausable.sol	OwnerPausable.sol
SWA	Swap.sol	Swap.sol
SUS	SwapUtils.sol	SwapUtils.sol
XNE	xNerve.sol	xNerve.sol

File Dependency Graph




Finding Summary
29%
71%
Minor
Informational

Manual Review Findings

ID	Title	Type	Severity	Resolved
LPT-01	Variable Mutability Optimization	Gas Optimization	Informational	✓
NTN-01	Redundant Visibility Specifier	Coding Style	Informational	✓
SWA-01	Inconsistent Input Sanitization	Logical Fault	Minor	✓
SUS-01	Unchecked `address` Input	Logical Fault	Minor	✓
SUS-02	Redundant Duplication of Statement	Gas Optimization	Informational	✓
XNE-01	Variable Mutability Optimization	Gas Optimization	Informational	✓
XNE-02	Leftover Comment	Coding Style	Informational	✓

LPT-01: Variable Mutability Optimization

Type	Severity	Location
Gas Optimization	Informational	LPToken.sol L20

Description:

The `swap` variable of the `LPToken` contract is only assigned to once during the `constructor` of the contract.

Recommendation:

We advise it to be set to `immutable` greatly benefitting from the gas optimizations it brings.

Alleviation:

The `swap` variable was properly set to `immutable` optimizing the codebase.

NTN-01: Redundant Visibility Specifier

Type	Severity	Location
Coding Style	Informational	NerveToken.sol L12

Description:

The `MINTER_ROLE` variable is declared as `public` yet is `constant`.

Recommendation:

We advise it to be set to `internal` as it has no use outside of the contract.

Alleviation:

The Nerve token contract is already deployed and as such, this non-security related change will not be reflected in the live token deployment.

SWA-01: Inconsistent Input Sanitization

Type	Severity	Location
Logical Fault	Minor	Swap.sol L160, L162, L166, L170

Description:

The linked fee variables are evaluated during the `constructor` of the contract to be strictly less than the specified variable, however, their respective setters in `SwapUtils` permit them to be inclusive of the limit.

Recommendation:

We advise the checks to be adjusted either on the `Swap` constructor or the `SwapUtils` setters to ensure consistency within the codebase.

Alleviation:

The Nerve Finance team stated that for future deployment of pools, this finding will be implemented in the codebase but for now it will remain as is.

SUS-01: Unchecked `address` Input

Type	Severity	Location
Logical Fault	Minor	SwapUtils.sol L1322-L1325

Description:

The `setDevAddress` function does not validate the `_devaddr` argument which can lead to a misconfiguration of the system.

Recommendation:

We advise a zero-address check to be imposed here to ensure that the system is configured properly at all times.

Alleviation:

A zero-address check was set in the codebase and will be in effect for future deployments.

SUS-02: Redundant Duplication of Statement

Type	Severity	Location
Gas Optimization	Informational	SwapUtils.sol L879, L886, L916, L937, L940, L960

Description:

The `totalSupply` getter function of `self.lpToken` is invoked numerous times across the execution of the `addLiquidity` function whilst it remains unchanged.

Recommendation:

We advise its result to be cached into an in-memory variable that is consequently utilized. For this purpose, we also suggest the `AddLiquidity` event emitted to contain the cached total supply variable with the `toMint` added to it.

Alleviation:

This optimization was applied to the codebase and will be reflected in future deployments of pools.

XNE-01: Variable Mutability Optimization

Type	Severity	Location
Gas Optimization	Informational	xNerve.sol L12

Description:

The `nerve` variable of the `xNerve` contract is only assigned to once during the `constructor` of the contract.

Recommendation:

We advise it to be set to `immutable` greatly benefitting from the gas optimizations it brings.

Alleviation:

The Nerve Finance team acknowledged the optimization this finding presents but cannot redeploy the xNerve token solely for optimizations and as such, the live deployment will remain as is.

XNE-02: Leftover Comment

Type	Severity	Location
Coding Style	Informational	xNerve.sol L19

Description:

The linked comment states that the user must pay `SUSHI` but the functionality within acquires `NRV`.

Recommendation:

We advise this comment to be updated to reflect the latest state of the codebase.

Alleviation:

The comment has been updated in the codebase of the project for clarity of future developers and readers of the codebase.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.