Security Assessment

Neko Network

Aug 6th, 2021

Table of Contents

Summary

This report has been prepared for Maze Protocol to discover issues and vulnerabilities in the source code of the Neko Network project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Neko Network
Description	A Lending Platform with Mining
Platform	Ethereum, BSC
Language	Solidity
Codebase	https://github.com/maze-protocol-official/neko-core-protocol
Commit	1. 2c5a6984643f565a153f057bc3bfebfbeadac919 2. a8d3c160bf385b8cf8ac0e998c76f01a6921cf56

Audit Summary

Delivery Date	Aug 06, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
Critical	0	0	0	0	0	0
Major	7	0	4	3	0	0
Medium	0	0	0	0	0	0
Minor	7	0	1	3	3	0
Informational	9	0	3	4	2	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
NMM	configuration/NekoManagement.sol	298ffbbba43ba2638901f4353730b3507f618f78382d38f665b9f9fe92dba204
AMP	dependencies/openzeppelin/contracts/Address.sol	52667e08d9c66b4cdf84850d94a54e142d3375c2784882bc8bae6e72b7c1a8eb
BEP	dependencies/openzeppelin/contracts/BEP20.sol	60fe7f8f768e75cceec3d21067b18e35ffaa5d94be0c4136312a3c7eed9a406d
CMP	dependencies/openzeppelin/contracts/Context.sol	d4ee56f31d60432de7cc5c423c1958f188405bdaafc6481936de69239bd8970b
ESM	dependencies/openzeppelin/contracts/EnumerableSet.sol	e3714759569aa412612eef66b0315bb48f11b60f09b11add0faf5bf9dca8bd8a
IBE	dependencies/openzeppelin/contracts/IBEP20.sol	b1f7a0752b035961bdec8de9db6d656036fbcc718cf5e6368ab841fd4bac107a
IBP	dependencies/openzeppelin/contracts/IBEP20Detailed.sol	63f4c9642559547d62820ef2bf7822bddc4e97f9ddb8e06c035f5daf8c72d2bc
IUV	dependencies/openzeppelin/contracts/IUniswapV2BEP20.sol	f443c7e295eccfe273c38144921eb605a15971cbfc7636910864b37933ba89da
OMP	dependencies/openzeppelin/contracts/Ownable.sol	0cb30ddc7998dc6d04c8cdcc19f8b08164a2ea0219e95451feb992f9818d2e62
SBE	dependencies/openzeppelin/contracts/SafeBEP20.sol	a831e8b38a7fdd0c1cb1de0838cb186bb8bafe9feb5d628a7bd891392edb3cc9
SMM	dependencies/openzeppelin/contracts/SafeMath.sol	026d2efe05ac42afa8bd4b8bd3fcc5c7ffbcc6b1f16aeda06442e9035b36194a
IAI	interfaces/IAaveIncentivesController.sol	bf8c3f49be857cb8f24a738d882d981cebf53d687ab83f3063a71cef31e35baf
IDT	interfaces/IDebtToken.sol	9826dfbb5f77949e50b3005478d90e8d566a4a9c52598094538618a161c6f8e0
IMT	interfaces/IMToken.sol	1cfca7464146297eefaa01a179c091e3bf795919b507f6665ae2352ec298e95c

ID	File	SHA256 Checksum
IMM	interfaces/IMine.sol	8e8f7c921fa6bdcf7b7b305c7762b73761388975e0d0f403bca92a298f0497e7
IMP	interfaces/IMiningToken.sol	600bf5687c882051cc0465c57fc642321c639be2dce6587c54c03a33aa9cd457
INM	interfaces/INekoManagement.sol	cfd035b18c92d73522b92466e3b448714350ca28702bf121572a897d63fb3572
IPA	interfaces/IPancakeAssetPrice.sol	c739c954d2f3af992cc58806c19ae9019288a430c4472bcaaf5ba65b8803ff83
IPP	interfaces/IPancakePair.sol	65e67dbe3c160d58163189bec2ed105526f6bb0c886f2a07bb4ff46346696e77
IPM	interfaces/IPool.sol	4f9baae169230c827f01e4c097ab28b1e1f2d04162ad28c96296d77b70fd76ac
IPC	interfaces/IPoolCollateralManager.sol	1d9549fc351660f2e8fd408b89b3b1975607aa6f310c5afa56131d1c3e2814b2
IPK	interfaces/IPoolParam.sol	eb4ab005deec90ba08e4c35171b9c40cbbf416efb586c052045d996229640c23
IPO	interfaces/IPriceOracleGetter.sol	e58da6662835baa5c3b12fecba83f60d382169ff08a4911c90cb7816bf706e7e
IRI	interfaces/IReserveInterestRateStrategy.sol	9d814ac478de12ff3632c60f5fa7b1f9681a4a68fbb2575c71a1fb62594e7ebe
IVT	interfaces/IVoteToken.sol	9edb14c3137cc75a34db0bfdfeab94e0e20b53abb391f7416bea8a174ef2ab8c
RCM	libraries/configuration/ReserveConfiguration.sol	bf908897c799cd0ecacb7ad2e0ee9894ca0b0369c84423257ab903fe3923b674
UCM	libraries/configuration/UserConfiguration.sol	faa40fef98594d91ad16049c25ada130d946b18c434d7b2a83298a6310be885b
EMP	libraries/helpers/Errors.sol	17d4d2b40e2d160f309a1c7f0409f0f302a8aa0116577804a47a393b8946ac5b
HMP	libraries/helpers/Helpers.sol	baf9d2a9a058cb6e477bdedc5d6cd4b2a4695dddc1c8fb87c09f362ff6ab5359

ID	File	SHA256 Checksum
GLM	libraries/logic/GenericLogic.sol	6adcd0424e1b2ab8565118e0ead860618efa3dc0827797ad615f8a6860526797
RLM	libraries/logic/ReserveLogic.sol	7456a5b8d0d66ede1a5b84229390c12849792821aade9f3ec768d8886db78f76
VLM	libraries/logic/ValidationLogic.sol	89806e848e4da18b723b2d75431c459d547e7544c06564dcfc7af6c75537c5fb
FPM	libraries/math/FixedPoint.sol	f2a979a74f86cf0f85413ab6cdfcc938c880619922aa7b1e5d72d8325e6adf84
MUM	libraries/math/MathUtils.sol	f87438192e2feaa3eb37ca05b6a5a0c3ceedf9db09577b63922530da6cc7173d
PMM	libraries/math/PercentageMath.sol	d11ad3e07bb8b723de053c93fcaef0429162b8f1e90bd55cd0f8b315dfc8ef0f
WRM	libraries/math/WadRayMath.sol	e61864e7bff3d1e1c52ee89f99c6ee7ec50b601685f70dc21713671b4401d23a
WRE	libraries/math/WadRayMathExternal.sol	9b1bd005bf172afaea5cb3c2eec7ee6f9455be41bd2c371f6be8a0218c9ea455
PLM	libraries/pancake/PancakeLibrary.sol	e7beeec229b15468cfaf833bf2e14f720265d4fff6e1d4a574b707ffeaaf2a3b
POL	libraries/pancake/PancakeOracleLibrary.sol	5341a7d4bcaa7acc6309ee0a65cde08aea1c33e4f29799bd2e8d2a86dbfc52b7
DTM	libraries/types/DataTypes.sol	4a4d67135f6cc3fdd9944893f208c1ff62126895cd176fed40bae9aa3a436471
VIM	libraries/VersionedInitializable.sol	3993eb062af0df31c8867a2ae0b8df3a88e012fc551f6ca0142070a00c7b42d8
IWB	misc/interfaces/IWBNB.sol	ea3e754bfb4a3e7fa3f638b15464865ab06ddc639957bddada1a0270fb6aa40d
IWN	misc/interfaces/IWBNBGateway.sol	6327ac8d95ae18428e650341e7f9ef91c70a1a3e6becf07f41cb712cb15dd69d
WBN	misc/interfaces/WBNB.sol	622cfefe4088bb24f5bdcf8b4dcaf857ab9050c181df2685d18a1e6f8e31de1a

ID	File	SHA256 Checksum
PAP	misc/PancakeAssetPrice.sol	9fc4aa8c0344d4c35584da73584584083f82d10c39a3e3b8af56d4b3d394d112
PAM	misc/PancakeAssetPrice2.sol	2aacc49aa691ab0b95f61a1128b47484e60ef38394a0758f61bdfdf4e3455a16
POM	misc/PriceOracleManager.sol	86261840ab52b383fb2828aae36bc82de90f1acd05bccd75620aa7546e15578f
WBB	misc/WBNBGateway.sol	4ccdeeb41f934767cd015ddebf6aea634d37f3295a84b016aa0c21d4bf6065fe
LPP	pool/LPPool.sol	f298fed1e4d3f31bf06f82f9df573bfa40618730859e8e71df42e458e0224302
MMM	pool/MainPool.sol	9dea2d0729f1e3155e03108e7d58e11a08693f7ba24ce0c328b593ed6eb76464
PPP	pool/Mine.sol	e14a6e63d4b262b29fa8db2d587b7b6c3a5f2e392609cc4daffdac3fed4e0234
PCM	pool/PoolCollateralManager.sol	598e417fc9d3de98860d6fdd7e8857f6a112879a6c87806ebea0b271cfb29e0b
PSM	pool/PoolStorage.sol	67d324ddd9e85d3c3053ea393538da2f91433025cdb9799b827e29a89ccd2873
DTB	tokenization/base/DebtTokenBase.sol	e72deddc5a5cf949165bd7919816ac168ca9779ff849a8993c8ceb50bfafb85c
DTP	tokenization/DebtToken.sol	39a24d38d6c6ae5cb909b6d00cfbb3a7c47b2f9478c9cf27c37583ac257c0768
FTM	tokenization/FundingToken.sol	57323c63337f64750cb3b8f26edd4799fea82e6a204312a8f628d310e7d1c87f
IBM	tokenization/IncentivizedBEP20.sol	0085879b9cbc4aa718b65aef682b908ab175f2aca75098cd4fa78d84446874ff
MTM	tokenization/MiningToken.sol	dbaf2da52b2cd6453d253def52097edeb7c160cdce7482a846aced4fceb7e1ea
RTM	tokenization/ReserveToken.sol	553272b8d97304914447d6b5af413a016ef6567f581adaea8d81e6af0d33d715

ID	File	SHA256 Checksum
VTM	tokenization/VoteToken.sol	9e6c814a30ea694efe62320c432aec4f385c7aba84eddc2de3fa5460683de3c7

It should be noted that the system design includes a number of economic arguments and assumptions. These were explored to the extent that they clarified the intention of the code base, but we did not audit the mechanism design itself. Note that financial models of blockchain protocols need to be resilient to attacks. It needs to pass simulations and verifications to guarantee the security of the overall protocol. The correctness of the financial model is not in the scope of the audit.

To bridge the gap of trust between owner and users, the owner needs to express a sincere attitude regarding the considerations of the administrator team's anonymity.

The owner of `NekoManagement` has the responsibility to notify users with the following capability in `NekoManagement`:

- set the address of the main pool through `setMainPool()`
- set the address of the lp pool through `setLPPool()`
- set the address of the mining pool through `setMinePool()`
- set address of the price oracle through `setPriceOracle()`
- set address of the collateral manager through `setCollateralManager()`
- set the address of the gateway through `setGateway()`
- set the pool administrator through `setPoolAdmin()`

The pool administrator of `LPPool` has the responsibility to notify users with the following capability in `PoolStorage`:

- initialize a reserve through `initPool()`

The pool administrator has the responsibility to notify users with the following capability in `PoolStorage`:

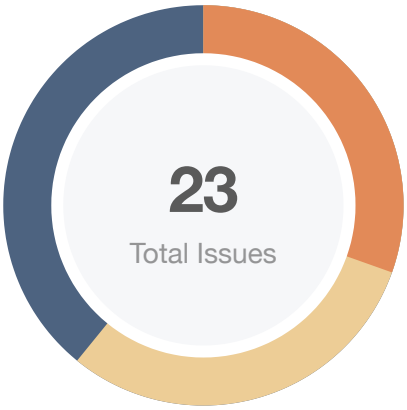
- initialize a reserve through `initPoolReserves()`
- set degrade flag through `setConfiguration()`
- set reserve withdrawal fee through `setConfiguration()`
- set funding withdrawal fee through `setConfiguration()`
- enable and disable borrowing through `setConfiguration()`
- activate and deactivate a reserve through `setConfiguration()`
- activate and deactivate funding through `setConfiguration()`
- activate and deactive reserving through `setConfiguration()`
- set loan to value through `setConfiguration()`
- set liquidation threshold through `setConfiguration()`
- set liquidation bonus through `setConfiguration()`

The owner of `FundingToken`/`ReserveToken`/`MiningToken`/`DebtToken`/`VoteToken` has the responsibility to notify users with the following capability in respective contracts:

- Owner may mint and burn uncapped tokens.
- Except `FundingToken`, the owner may transfer at will.

The contract is serving as the underlying entity to interact with third-party PancakeSwap protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties may be compromised thus leading to assets being lost or stolen, eg. flashloan.

Findings



Critical	**0** (0.00%)
Major	**7** (30.43%)
Medium	**0** (0.00%)
Minor	**7** (30.43%)
Informational	**9** (39.13%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unlocked Compiler Version	Language Specific	Informational	Partially Resolved
GLOBAL-02	Incompatibility With Deflationary Tokens	Logical Issue	Minor	Resolved
GLOBAL-03	Owner Privilege of Tokens	**Centralization / Privilege**	**Major**	**Partially Resolved**
DTM-01	Typo in Variable Name	Language Specific	Informational	Partially Resolved
FTM-01	Unrestricted Token Transfer	Logical Issue	Major	Resolved
IBM-01	Set `immutable` to Variables	Gas Optimization	Informational	Resolved
IBM-02	Lack of Input Validation	Logical Issue	Minor	Resolved
INM-01	Missing Emit Events	Logical Issue	Informational	Resolved
LPP-01	Unsafe Access to the First LP Pool	Volatile Code	Major	Resolved
MMC-01	Unused Library	Gas Optimization	Informational	Partially Resolved
MMP-01	Redundant Interface	Gas Optimization	Informational	Resolved
NMM-01	Owner Privilege of `NekoManagement`	**Centralization / Privilege**	**Minor**	**Acknowledged**
PAM-01	Third Party Dependencies	Control Flow	Minor	Acknowledged

ID	Title	Category	Severity	Status
PAP-01	Duplicate `PancakeAssetPrice` Contracts	Logical Issue	Informational	Resolved
POM-01	Owner May Set Asset Price	**Centralization / Privilege**	**Major**	**Resolved**
PSM-01	Inaccurate Error Messages	Logical Issue	Minor	Resolved
PSM-02	Variable Tight Packing	Gas Optimization	Informational	Acknowledged
PSM-03	Owner Privilege of `PoolStorage`	**Centralization / Privilege**	**Major**	**Partially Resolved**
PSM-04	Unrestricted Fee Setting	**Centralization / Privilege**	**Major**	**Partially Resolved**
PSM-05	Unrestricted Loan-To-Value Setting	**Centralization / Privilege**	**Major**	**Partially Resolved**
RCM-01	Unused Constants	Logical Issue	Minor	Acknowledged
RTM-01	Transfer of `ReserveToken`	Logical Issue	Informational	Acknowledged
WBB-01	Emergency Token Transfer in WBNBGateway	**Centralization / Privilege**	**Minor**	**Partially Resolved**

GLOBAL-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	Global	Partially Resolved

Description

The contract has unlocked compiler versions. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be dificult to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The client locked all contracts to 0.8.0 except the pancake libraries in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

GLOBAL-02 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	Global	Resolved

Description

During `deposit()` or `withdraw()`, the lending pool's internal asset balances are designed to always be consistent with actual token balances maintained in individual ERC20 token contracts. However, deflationary tokens that charge a certain fee for every transfer or rebasing tokens may not meet the assumption behind these low-level asset-transferring routines. This may introduce unexpected balance inconsistencies when comparing internal asset records with external ERC20 token contracts.

Recommendation

If deflationary/rebase currencies are to be supported, additional checks need to be added before and after transfers to ensure the book-keeping amount is accurate. Otherwise, we advise avoiding such currencies during deployment.

Alleviation

The client added checks and resolved this issue in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

GLOBAL-03 | Owner Privilege of Tokens

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Partially Resolved

Description

To bridge the gap of trust between owner and users, the owner needs to express a sincere attitude regarding the considerations of the administrator team's anonymity. The owner of `FundingToken`/`ReserveToken`/`MiningToken`/`DebtToken`/`VoteToken` has the responsibility to notify users with the following capability in respective contracts:

- Owner may mint, burn, transfer uncapped tokens and set or remove other owners.

Recommendation

We would like to enquire about precautions against abuse. Consider renouncing ownership when it is the right timing, or gradually migrating to a timelock plus multi-signature governing procedure and letting the community monitor in respect of transparency considerations.

Alleviation

The client removed the capability to set and remove other owners and added a validator to ensure owners are contracts but reserved the capability to mint, burn, and transfer in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

DTM-01 | Typo in Variable Name

Category	Severity	Location	Status
Language Specific	Informational	libraries/types/DataTypes.sol: 25	Partially Resolved

Description

“fronzens” should be “frozens” judging from its context and functionality.

Recommendation

We advise client to use "frozens" in this variable and the related functions.

Alleviation

The client fixed the typo in this variable but "fronzen" remain in related functions in `UserConfiguration` in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

FTM-01 | Unrestricted Token Transfer

Category	Severity	Location	Status
Logical Issue	Major	tokenization/FundingToken.sol	Resolved

Description

Funding tokens serve as proof of the user's asset deposit in the main pool and are not supposed to be freely transferred. Such restrictive design is reflected in the (re)implementation of several transfer methods in token contracts where the approval by `Owner` the main pool is required. However, `transferFrom()` is still inherited from `IncentivizedBEP20` and can be accessed to freely transfer tokens, making the aforementioned restriction ineffective. This may create token imbalance and cause further issues during withdrawal/repayment in the main pool.

Recommendation

We advise the client to review the functionality of `ftoken` and `rtoken` and disable unrestricted transfer methods.

Alleviation

The client reimplemented `transferFrom()` in `FundingToken` in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

IBM-01 | Set `immutable` to Variables

Category	Severity	Location	Status
Gas Optimization	Informational	tokenization/IncentivizedBEP20.sol: 23, 24	Resolved

Description

`_decimals` and `_owner` are never changed throughout the contract and could be declared `immutable` in the first place for gas optimization.

Recommendation

We advise the client to declare the aforementioned variables as `immutable`.

Alleviation

The client heeded our advice and resolved this issue in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

IBM-02 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Minor	tokenization/IncentivizedBEP20.sol: 36	Resolved

Description

The assigned values to `_owner` should be verified as non-zero values to prevent being mistakenly assigned as `address(0)` in the `constructor()` function.

Recommendation

We advise the client to add zero check for `owner`.

Alleviation

The client added checks in other contracts that inherits `IncentivizedBEP20` in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

INM-01 | Missing Emit Events

Category	Severity	Location	Status
Logical Issue	Informational	interfaces/INekoManagement.sol	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `setMinePool()`
- `setCollateralManager()`
- `setGateway()`
- `setPoolAdmin()`

Recommendation

We advise the client to add events for sensitive actions and emit them in the functions.

Alleviation

The client added the missing emit events and fixed this issue in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

LPP-01 | Unsafe Access to the First LP Pool

Category	Severity	Location	Status
Volatile Code	Major	pool/LPPool.sol: 89, 95, 113, 134, 156, 105	Resolved

Description

When the input `asset` does not match with any address index in `_poolAddress` mapping, one will get zero as `_pid`. This gives precarious access to the first pool. For example, a malicious user may deploy a mock token and feed its address to `deposit()` to mint `lpNeko` and `pNeko` for free in the first pool. He/she may then withdraw the real token through `withdraw()`.

Recommendation

We advise the client to leave `poolInfos[0]` alone and start with `poolInfos[1]` like in `Mine`.

Alleviation

The client heeded our advice and fixed this issue in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

MMC-01 | Unused Library

Category	Severity	Location	Status
Gas Optimization	Informational	libraries	Partially Resolved

Description

Several libraries are never used in the project:

- `MathUtils`
- `WayRayMathExternal`
- `ReserveLogic`

Recommendation

We advise the client to review their functionalities and remove them if there is no plan for further use.

Alleviation

The client removed `MathUtils` and `WayRayMathExternal` while `ReserveLogic` is preserved in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

MMP-01 | Redundant Interface

Category	Severity	Location	Status
Gas Optimization	Informational	interfaces	Resolved

Description

Several interfaces are never used in the project:

- `IReserveInterestRateStrategy`
- `IPoolParam`

Recommendation

We advise the client to review their functionality and remove them if there is no plan for further use.

Alleviation

The client fixed `IPoolParam` to accommodate other changes in the project and removed `IReserveInterestRateStrategy` in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

NMM-01 | Owner Privilege of `NekoManagement`

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	configuration/NekoManagement.sol	**Acknowledged**

Description

To bridge the trust gap between owner and users, the owner needs to express a sincere attitude in regard to the considerations of the administrator team's anonymity. The owner has the responsibility to notify users with the following capability in `NekoManagement`:

- set the address of the main pool through `setMainPool()`
- set the address of the lp pool through `setLPPool()`
- set the address of the mining pool through `setMinePool()`
- set address of the price oracle through `setPriceOracle()`
- set address of the collateral manager through `setCollateralManager()`
- set the address of the gateway through `setGateway()`
- set the pool administrator through `setPoolAdmin()`

Recommendation

We would like to enquire about precautions against abuses. Consider renouncing ownership when it is the right timing, or gradually migrating to a timelock plus multi-signature governing procedure and letting the community monitor in respect of transparency considerations.

Alleviation

The client added a few new contracts to manage contract ownership while reserving these privileges in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

PAM-01 | Third Party Dependencies

Category	Severity	Location	Status
Control Flow	Minor	misc/PancakeAssetPrice2.sol	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party pancakeSwap protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties may be compromised thus leading to assets being lost or stolen, eg. flashloan.

Recommendation

We understand that the business logic requires interaction with pancakeSwap for price feeds. We encourage the team to constantly monitor the status of those 3rd parties to mitigate the side effects when unexpected activities are observed.

PAP-01 | Duplicate `PancakeAssetPrice` Contracts

Category	Severity	Location	Status
Logical Issue	Informational	misc/PancakeAssetPrice.sol	Resolved

Description

`PancakeAssetPrice` and `PancakeAssetPrice2` provide the same price feed functionality that only differs in the address of `PancakePair`. We would like to enquire on the reason for such design.

Alleviation

The client kept `PancakeAssetPrice2` in `PancakeAssetPrice.sol` and removed `PancakeAssetPrice` in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

POM-01 | Owner May Set Asset Price

Category	Severity	Location	Status
Centralization / Privilege	Major	misc/PriceOracleManager.sol: 75~78	Resolved

Description

`Owner` of the oracle manager may set asset price with `setAssetPrice()`. Such privilege gives owner immense control over users as all major operations, namely deposit/withdraw/borrow/repay/liquidate, depends on the price feed. We would like to enquire on precautions against potential manipulative abuses such as over-collateralize assets, force liquidation etc.

Alleviation

The client removed `setAssetPrice()` function thus resolving this issue in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

PSM-01 | Inaccurate Error Messages

Category	Severity	Location	Status
Logical Issue	Minor	pool/PoolStorage.sol	Resolved

Description

The set functions first validate that a reserve has been initialized before changing one of its parameters. Therefore the error message should be `ASSET_NOT_INIT` rather than `ASSET_ALREADY_INIT`. Besides, a nonzero `ftoken`, not `fmtoken`, is the preferable flag of initialized reserve like in `initPoolReserves()`.

The incorrect require statement is used in:

- setDegrade()
- setRWithdrawFee()
- setFWithdrawFee()
- setBorrowingEnabled()
- setActive()
- setFActive()
- setRActive()
- setLtv()
- setLiqThreshold()
- setLiqBonus()

Recommendation

We advise the client to review their functionalities and change the require statements to make them consistent with function logic.

Alleviation

The client removed all the set parameter functions in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

PSM-02 | Variable Tight Packing

Category	Severity	Location	Status
Gas Optimization	Informational	pool/PoolStorage.sol	Acknowledged

Description

The uint8 variable `MAX_DATAVALUE` is slotted between two uint256 variables unoptimized.

Recommendation

We advise that it is instead relocated under the `_management` address to ensure tight packing.

Alleviation

The client did not address this issue but may revise it in the future.

PSM-03 | Owner Privilege of `PoolStorage`

Category	Severity	Location	Status
Centralization / Privilege	Major	pool/PoolStorage.sol: 98, 140, 149, 158, 167	Partially Resolved

Description

To bridge the trust gap between owner and users, the owner needs to express a sincere attitude in regard to the considerations of the administrator team's anonymity. The pool administrator has the responsibility to notify users with the following capability in `PoolStorage`:

- initialize a reserve through `initPoolReserves()`
- set degrade flag through `setDegrade()`
- set reserve withdrawal fee through `setRWithdrawFee()`
- set funding withdrawal fee through `setFwithdrawFee()`
- enable and disable borrowing through `setBorrowingEnabled()`
- activate and deactivate a reserve through `setActive()`
- activate and deactivate funding through `setFActive()`
- activate and deactive reserving through `setRActive()`
- set loan to value through `setLtv()`
- set liquidation threshold through `setLiqThreshold()`
- set liquidation bonus through `setLiqBonus()`

Recommendation

The aforementioned privileges grant the administrator huge leverages over the users' asset deposit and we would like to enquire about precautions against potential abuses. Consider renouncing ownership when it is the right timing, or gradually migrating to a timelock plus multi-signature governing procedure and letting the community monitor in respect of transparency considerations.

Alleviation

The client removed the set functions but added `setConfiguration()` which still allows modification of all the aforementioned sensitive parametersin commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

PSM-04 | Unrestricted Fee Setting

Category	Severity	Location	Status
Centralization / Privilege	**Major**	pool/PoolStorage.sol	**Partially Resolved**

Description

The pool administrator is capable of setting withdrawal fees of a reserve through `setFWithdrawFee()` and `setRWithdrawFee()` to any `uint16` values. This allows the administrator to either in effect lock the user's assets by setting fee rate above one thus reverting ensuing transfer or take control of them by transferring an arbitrarily high fraction to a designated `rewardAddress`.

Recommendation

We advise the client to add a below one upper limit check to the fee setting functions and would like to further enquire about other precautions against abuse. Consider renouncing ownership when it is the right timing, or gradually migrating to a timelock plus multi-signature governing procedure and letting the community monitor in respect of transparency considerations.

Alleviation

The client removed these functions but still allows administrator to set fee through the newly added `setConfiguration()` in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

PSM-05 | Unrestricted Loan-To-Value Setting

Category	Severity	Location	Status
Centralization / Privilege	**Major**	pool/PoolStorage.sol	**Partially Resolved**

Description

The pool administrator is capable of setting the loan-to-value of a reserve to any `uint16` value. A high ltv would allow risky under-collateralized loans while a drastic lowering of ltv could potentially force users to liquidate.

Recommendation

We advise the client to restrict ltv below one and would like to enquire on precautions against abuse. Consider renouncing ownership when it is the right timing, or gradually migrating to a timelock plus multi-signature governing procedure and letting the community monitor in respect of transparency considerations.

Alleviation

The client removed these functions but still allows the administrator to set the loan-to-value through the newly added `setConfiguration()` in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

RCM-01 | Unused Constants

Category	Severity	Location	Status
Logical Issue	Minor	libraries/configuration/ReserveConfiguration.sol: 18, 20, 33, 35, 44, 47	Acknowledged

Description

The aforementioned constants seem to be vestiges of previous repository and are not used throughout the project.

Recommendation

We advise the client to remove unused constants.

Alleviation

The client did not address this issue but may revise it in the future.

RTM-01 | Transfer of `ReserveToken`

Category	Severity	Location	Status
Logical Issue	Informational	tokenization/ReserveToken.sol	Acknowledged

Description

The client did not reimplement `transferFrom()` in `ReserveToken` and we wonder if this suits their design.

Alleviation

The client explains that reserve tokens are designed to support the transfer of token entitlements across chains. For example, a user deposit USDT into a BSC reserve to acquire `rUSDT` and `rmUSDT`. He/she may then transfer those rtokens across chains to withdraw USDT from another reserve on Polygon.

WBB-01 | Emergency Token Transfer in WBNBGateway

Category	Severity	Location	Status
Centralization / Privilege	Minor	misc/WBNBGateway.sol: 167~174, 182~184	Partially Resolved

Description

To bridge the trust gap between owner and users, the owner needs to express a sincere attitude regarding the considerations of the administrator team's anonymity. The pool administrator has the responsibility to notify users with the following capability in `PoolStorage`:

- The Owner may transfer at will through `emergencyTokenTransfer()` and `emergencyBNBTransfer()`.

Recommendation

The aforementioned privileges grant the administrator huge leverages over the users' asset deposit and we would like to enquire about precautions against potential abuses. Consider renouncing ownership when it is the right timing, or gradually migrating to a timelock plus multi-signature governing procedure and letting the community monitor in respect of transparency considerations.

Alleviation

The client added a few new contracts to manage contract ownership while reserving these privileges in commit `a8d3c160bf385b8cf8ac0e998c76f01a6921cf56`.

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.

Gas Optimization

Gas Optimization findings 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.

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.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

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.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.