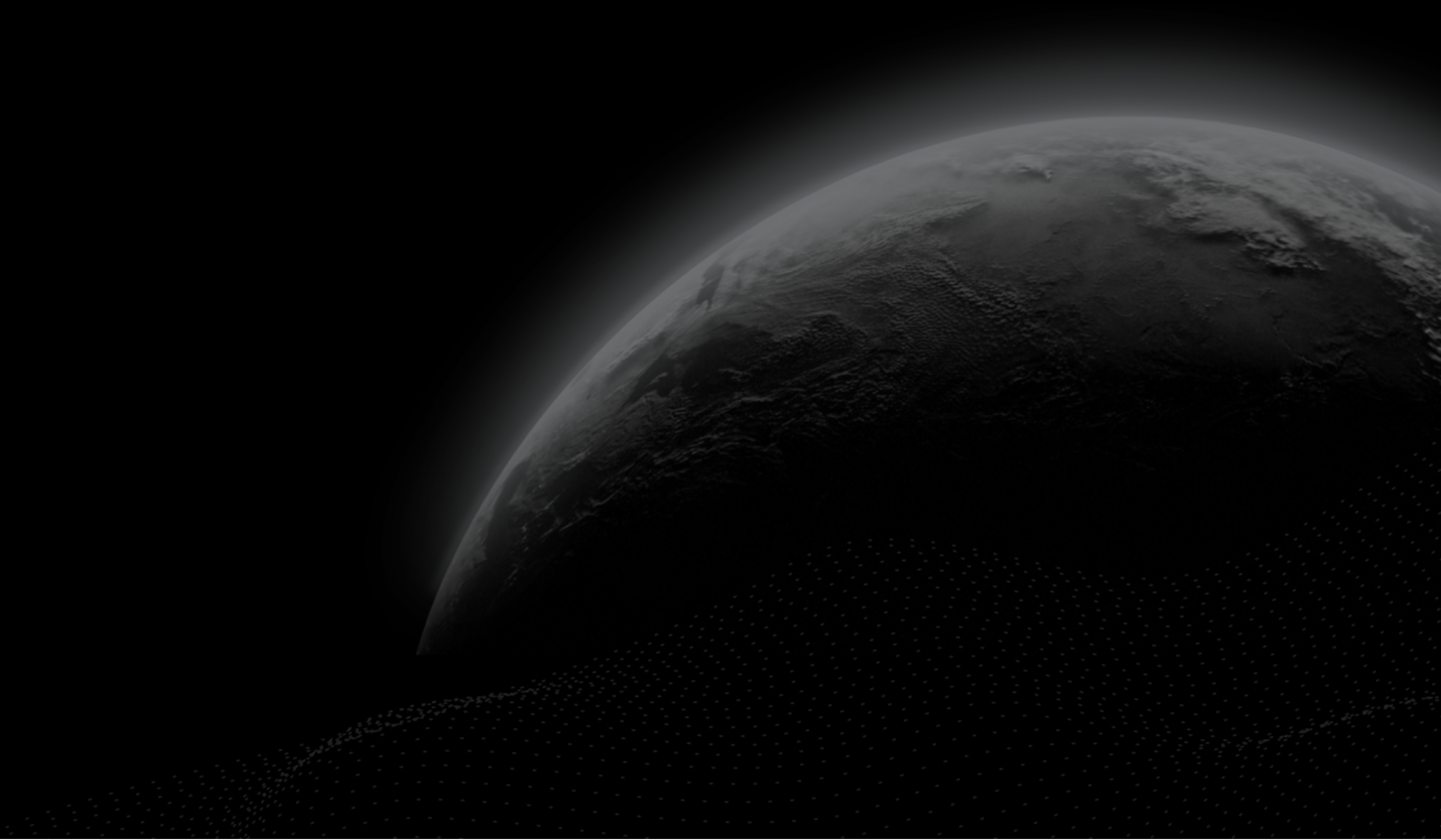
CERTIK

Security Assessment

Venus - Allocation of Income

CertiK Assessed on Sept 12th, 2023



CertiK Assessed on Sept 12th, 2023

Venus - Allocation of Income

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Binance Smart Chain (BSC)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 09/12/2023	N/A

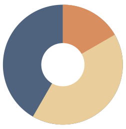
CODEBASE

https://github.com/VenusProtocol/isolated-pools
https://github.com/VenusProtocol/venus-protocol
https://github.com/VenusProtocol/protocol-reserve
View All in Codebase Page

COMMITS

protocol-reserve Base: 66537d61407287108b33057e4a5d572fcbff0c1f
protocol-reserve Update1:
dfb653d2e3fe163a248bbd9f8951cd6b96b06390
View All in Codebase Page

Vulnerability Summary



12	9	2	0	1	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

🟥 0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧 2	Major	2 Mitigated	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨 0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨 5	Minor	4 Resolved, 1 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦 5	Informational	5 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | VENUS - ALLOCATION OF INCOME

CODEBASE | VENUS - ALLOCATION OF INCOME

Repository

https://github.com/VenusProtocol/isolated-pools

https://github.com/VenusProtocol/venus-protocol

https://github.com/VenusProtocol/protocol-reserve

Commit

protocol-reserve Base: 66537d61407287108b33057e4a5d572fcbff0c1f

protocol-reserve Update1: dfb653d2e3fe163a248bbd9f8951cd6b96b06390

protocol-reserve Update 2: 92cc2491a0e1b928b2b968f5c9e729461dcca891

protocol-reserve Update 3: 762c054693c39a693fbaa2bdb1680635146f79c1

PR:262 Base: 0ea2384b0602080f3483ebe6f4a4a485f6365d92

PR:262 Update1: 2e0f5f64f49aba3150e49c7a7e34a28826726f20

PR262 Update2: fc1178a42f5179b3ce379900fd86a883bd8b2329

PR289 Base: 08cd99c3e13f7a82c78e58c81dc2de6b11db0104

PR289 Update1: b11d2972dbbf9855a7560f26745fae783bc15e7e

PR289 Update 2: 348955dae2ba4728fd995a2abdf04ff6db0d3914

PR207 Base: 86d6de62c3787ce24ce6c85cde160c5b19fe9979

PR207 Update 1: 92353cf35ddbbfdd67ee255ac095ee861d6bb7fb

AUDIT SCOPE | VENUS - ALLOCATION OF INCOME

15 files audited ● 1 file with Acknowledged findings ● 14 files without findings

ID	Repo	Commit	File	SHA256 Checksum
● PSR	VenusProtocol/protocol-reserve	66537d6	contracts/ProtocolReserve/ProtocolShareReserve.sol	f4123d5a2dba3e0b1e6e45b175727d51cd85ccece6daa927f7beb3528de9cb8e
● VBV	VenusProtocol/venus-protocol	0ea2384	VBep20.sol	b16dd47e6390e68edb75783cd48b61972771a74eb10dfbec8d752608e3ec709d
● VTV	VenusProtocol/venus-protocol	0ea2384	VToken.sol	c37c6f89c6024d9473df0dd5fa58a9bb411a7652705c88d82b7394527aa4e7a9
● VTI	VenusProtocol/venus-protocol	0ea2384	VTokenInterfaces.sol	443840180642bdbf445b3b5268eeaa4460d44145609b28371565cfb1748ed7ec
● ERU	VenusProtocol/venus-protocol	0ea2384	ErrorReporter.sol	2df048755d3379a73ed00f9e3937342ca9362ed9b1ae636d837f5fe5213e6463
● VBN	VenusProtocol/venus-protocol	08cd99c	VBNBAdmin.sol	74a1eedf4c1cc22df52516328156f6b1e32021d3e641e26732e9cc87e7026b5b
● VBB	VenusProtocol/venus-protocol	08cd99c	VBNBAdminStorage.sol	be7bc3ad37db5ba165e3284226d9e4c407e34cbb54cd3620b5d702a62eca98ba
● VTP	VenusProtocol/isolated-pools	86d6de6	VToken.sol	296ba63357978c4e181fc5ac29b1beba4fd7d98a8db4d10063e71131f6a1d313
● VIV	VenusProtocol/isolated-pools	86d6de6	VTokenInterfaces.sol	c45d15730b5be1e2f3a74c5ce90241989ee0a95d1feeeba680a2a5add8059f90
● CII	VenusProtocol/protocol-reserve	66537d6	contracts/Interfaces/ComptrollerInterface.sol	49deaf3a00d630650dca5a156bba1a445a944d63fcfbb9b02aee6be2f74c6fcb
● IID	VenusProtocol/protocol-reserve	66537d6	contracts/Interfaces/IIncomeDestination.sol	0cff4535841b5ff94ef400c30105d1605d5c1ef81940c78925b4f35a95c2dcf7

ID	Repo	Commit	File	SHA256 Checksum
IPI	VenusProtocol/protocol-reserve	66537d6	contracts/Interfaces/IPrime.sol	87477d74668df1643d895bfbaa700779d1423305f4e3905c7d285bd8f9e9138c
IPS	VenusProtocol/protocol-reserve	66537d6	contracts/Interfaces/IProtocolShareReserve.sol	dca738307ff5e99057fdad2d8d1e9f07a0561cde7298612a3dbf99ed7e7e94e2
IVT	VenusProtocol/protocol-reserve	66537d6	contracts/Interfaces/IVToken.sol	1d07e692bd5355b79e7974a56c238889a0c49516ec7f59c0f2b8fa32ec3bb3f0
PRI	VenusProtocol/protocol-reserve	66537d6	contracts/Interfaces/PoolRegistryInterface.sol	786148bbb7d6e4b161b4cb9b915aae2cb0227b913f3f9fe1a3196cc45a7f0998

APPROACH & METHODS | VENUS - ALLOCATION OF INCOME

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

- Testing the smart contracts against both common and uncommon attack vectors;
- 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.

SUMMARY | VENUS - ALLOCATION OF INCOME

This audit concerns the changes made in files outlined in the Audit Scope section within the following PR's up to the listed commit:

- Core pool - markets spread: PR https://github.com/VenusProtocol/venus-protocol/pull/262. Last commit considered: 2e0f5f64f49aba3150e49c7a7e34a28826726f20.
- Harvesting BNB income: PR https://github.com/VenusProtocol/venus-protocol/pull/289. Last commit considered: b11d2972dbbf9855a7560f26745fae783bc15e7e.
- Isolated pools - Liquidations & markets spread: PR https://github.com/VenusProtocol/isolated-pools/pull/207. Last commit considered: 92353cf35ddbbfdd67ee255ac095ee861d6bb7fb.

In addition, the `ProtocolShareReserve` contract from this repository: https://github.com/VenusProtocol/protocol-reserve was also in scope.

DEPENDENCIES | VENUS - ALLOCATION OF INCOME

Third Party Dependencies

The protocol is serving as the underlying entity to interact with third party protocols. The third parties that the contracts interact with are:

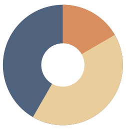
- ERC20 Tokens;

The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. Moreover, updates to the state of a project contract that are dependent on a read of the state of external third party contracts may make the project vulnerable to read-only reentrancy. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

Recommendations

We recommend constantly monitoring the third parties involved to mitigate any side effects that may occur when unexpected changes are introduced, as well as vetting any third party contracts used to ensure no external calls can be made before updates to its state.

FINDINGS | VENUS - ALLOCATION OF INCOME



12	0	2	0	5	5
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Venus - Allocation of Income. Through this audit, we have uncovered 12 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
VPB-02	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	● **Mitigated**
VPB-04	**Centralization Related Risks**	**Centralization**	**Major**	● **Mitigated**
PSR-09	Potential Reentrancy	Volatile Code	Minor	● Resolved
PSR-10	Lack Of Access Control	Logical Issue	Minor	● Acknowledged
VBN-01	Unprotected Initializer	Coding Issue	Minor	● Resolved
VPB-05	Incorrect `ReservesReduced` Event	Inconsistency	Minor	● Resolved
VPI-01	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
PSR-04	Access Check Does Not Follow Convention	Inconsistency	Informational	● Resolved
PSR-05	Using Both `uint` And `uint256`	Inconsistency	Informational	● Resolved
PSR-08	Incomplete NatSpec Comments	Inconsistency	Informational	● Resolved
VBN-02	Code Does Not Follow Specification	Inconsistency	Informational	● Resolved

ID	Title	Category	Severity	Status
VPB-03	Typos And Inconsistencies	Inconsistency	Informational	● Resolved

VPB-02 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	● **Major**	**VToken.sol (PR207-VToken): 93~122; contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 124~130; VBNBAdmin.sol (PR289-VBNBAdmin): 21~33; VBep20.sol (PR262-VBep20): 10**	● **Mitigated**

Description

`VBep20` , `Vtoken` , `VBNBAdmin` , and `ProtocolShareReserve` are upgradeable contracts and the `admin` of the proxy has the ability to update the implementation contract behind the proxy contract.

Any compromise to the `admin` account may allow a hacker to take advantage of this authority and change the implementation contract which is pointed to by proxy and steal all tokens held by the contracts or implement and execute other potentially malicious functionality.

Recommendation

We recommend that the team make efforts to restrict access to the admin of the proxy contract. A strategy of combining a time-lock and a multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

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

Short Term:

A combination of a time-lock and a multi signature (⅔, ⅗) wallet mitigate the risk by delaying the sensitive operation and avoiding 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 key compromised;
  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.

Long Term:

A combination of a time-lock on the contract upgrade operation and a DAO for controlling the upgrade operation mitigate the contract 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 user involvement;
  AND
- A medium/blog link for sharing the time-lock contract, multi-signers addresses, and DAO 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.

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

Alleviation

`[Venus, 08/25/2023]` : The admin of these contracts was or will be transferred to 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396, which is the Timelock contract used to execute the normal Venus Improvement Proposals (VIP).

For normal VIPs, the time config is: 24 hours voting + 48 hours delay before the execution.

So, these contracts will be upgraded only via a Normal VIP, involving the community in the process.

VPB-04 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	● **Major**	**VToken.sol (PR207-VToken): 634~635; contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve-Update1): 147, 157, 168; VBNBAdmin.sol (PR289-VBNBAdmin): 83~88, 90~99; VToken.sol (PR262-VToken): 211~212, 228~232, 248~249, 348~352, 561~562; VToken.sol (PR262-VToken-Update1): 336; VBNBAdmin.sol (PR289-VBNBAdmin-Update1): 58**	● **Mitigated**

Description

The centralization risks indicated here are only related to those within the scope of the audit. CertiK has not audited the `venus-protocol` repository before and we recommend users to carefully review the handling of privileged roles throughout the codebase. CertiK has audited the `isolated-pools` repository and more information regarding the centralization risks can be found in our previous audits: https://skynet.certik.com/projects/venus.

ProtocolShareReserve

The `owner` has the privilege to call the following functions:

- `setPoolRegistry()`
- `setPrime()`

Any compromise to the `owner` may allow the hacker to take advantage of this authority and do the following:

- Set `poolRegistry` to a malicious contract.
- Set `prime` to a malicious contract.

The role `DEFAULT_ADMIN_ROLE` can grant addresses the privilege to call the following functions:

- `addOrUpdateDistributionConfigs()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or these privileged functions may allow the hacker to take advantage of this authority and do the following:

- Add or update distribution configs to have assets sent to themselves.

PR262 VToken

The following privileges were added:

The role `DEFAULT_ADMIN_ROLE` can grant addresses the privilege to call the following functions:

- `_setReserveFactor()`
- `_reduceReserves()`
- `_setInterestRateModel()`
- `setReduceReservesBlockDelta()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or these privileged functions may allow the hacker to take advantage of this authority and do the following:

- Set the `reserveFactorMantissa` to any value up to the `reserveFactorMaxMantissa` .
- Reduce the reserves by transferring them to the `protocolShareReserve` .
- Set the interest rate model to a malicious contract so that rates are much higher or lower than expected.
- Change the `reduceReservesBlockDelta` to have reduce reserves be called more or less frequently than expected.

The role `admin` has authority over the following functions:

- `setAccessControlManager()`
- `setProtocolShareReserve()`

Any compromise to the `admin` account may allow a hacker to take advantage of this authority and do the following:

- Change the `accessControlManager` to a malicious contract to allow them to call any privileged functions that the `DEFAULT_ADMIN_ROLE` can grant privilege to.
- Change the `protocolShareReserve` to an address they control to have reserves transferred to themselves when reserves are reduced.

PR289 VBNBAdmin

The role `owner` has authority over the following functions:

- `setProtocolShareReserve()`
- `fallback()`

Any compromise to the `owner` account may allow a hacker to take advantage of this authority and do the following:

- Change the `protocolShareReserve` to a contract they control so that they receive `WBNB` when the reserves are reduced.
- Call the fallback function, which will call the `vBNB` contract, allowing them to execute any functions the `VBNBAdmin` contract has access to. In particular, as `VBNBAdmin` will be made the admin of `vBNB` , they will be able to execute any of the external or public functions as well as any functions only executable by the `admin` role.

The role `vBNB` has authority over the following functions:

- `receive()`

Any compromise to the `vBNB` account may allow a hacker to take advantage of this authority and do the following:

- Send native tokens directly to the contract.

PR207 VToken

The role `DEFAULT_ADMIN_ROLE` can grant addresses the privilege to call the following functions:

- `setReduceReservesBlockDelta()`

Any compromise to the `DEFAULT_ADMIN_ROLE` or these privileged functions may allow the hacker to take advantage of this authority and do the following:

- Change the `reduceReservesBlockDelta` to have reduce reserves be called more or less frequently than expected.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) 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 key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

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 public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

`[Venus, 08/25/2023]` :

ProtocolShareReserve

- The owner of the contract will be the Normal Timelock contract used via VIP's.
- Regarding the DEFAULT_ADMIN_ROLE, we'll use the AccessControlManager (ACM) deployed at https://bscscan.com/address/0x4788629abc6cfca10f9f969efdeaa1cf70c23555. In this ACM, only 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 (Normal Timelock) has the DEFAULT_ADMIN_ROLE. And this contract is a Timelock contract used during the Venus Improvement Proposals. We'll allow Normal, Fast-track and Critical timelock contracts to execute the function addOrUpdateDistributionConfigs().

PR262 VToken

- Regarding the DEFAULT_ADMIN_ROLE, we'll use the AccessControlManager (ACM) deployed at https://bscscan.com/address/0x4788629abc6cfca10f9f969efdeaa1cf70c23555. In this ACM, only 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 (Normal Timelock) has the DEFAULT_ADMIN_ROLE. And this contract is a Timelock contract used during the Venus Improvement Proposals. We'll allow Normal, Fast-track and Critical timelock contracts to execute the functions
  - `_setReserveFactor()`
  - `_reduceReserves()`
  - `_setInterestRateModel()`
  - `setReduceReservesBlockDelta()`

- The role admin will be the same Normal Timelock contract.

PR289 VBNBAdmin

- The owner of the contract will be the Normal Timelock contract used via VIP's.
- The role vBNB will be the vBNB contract deployed at https://bscscan.com/address/0xA07c5b74C9B40447a954e1466938b865b6BBea36. We'll set it in the VIP where the upgrade is proposed.

PR207 VToken

- Regarding the DEFAULT_ADMIN_ROLE, we'll use the AccessControlManager (ACM) deployed at https://bscscan.com/address/0x4788629abc6cfca10f9f969efdeaa1cf70c23555. In this ACM, only 0x939bd8d64c0a9583a7dcea9933f7b21697ab6396 (Normal Timelock) has the DEFAULT_ADMIN_ROLE. And this contract is a Timelock contract used during the Venus Improvement Proposals. We'll allow Normal, Fast-track and Critical timelock contracts to execute the function setReduceReservesBlockDelta()

Extra information

Current config for the three Timelock contracts:

- Normal: 24 hours voting + 48 hours delay
- Fast-track: 24 hours voting + 6 hours delay
- Critical: 6 hours voting + 1 hour delay

Addresses of the Timelock contracts:

- Normal timelock: https://bscscan.com/address/0x939bD8d64c0A9583A7Dcea9933f7b21697ab6396
- Fast-track timelock: https://bscscan.com/address/0x555ba73dB1b006F3f2C7dB7126d6e4343aDBce02
- Critical timelock: https://bscscan.com/address/0x213c446ec11e45b15a6E29C1C1b402B8897f606d

PSR-09 | POTENTIAL REENTRANCY

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 318	● Resolved

Description

The function `releaseFunds()` is callable by anyone and will eventually call `safeTransfer()` . The token contract that is called may implement hooks or make other external calls, allowing for reentrancy. In addition, it will make a call to `prime` , which is out of scope of this audit, but which may also make other external calls potentially allowing reentrancy.

Recommendation

We recommend adding a reentrancy guard to all user facing functions and to follow the check-effect-interaction pattern wherever possible.

Alleviation

`[CertiK, 08/28/2023]` : The client made the recommended changes in commits:

- 789151704b35cd79b236d9e7873f2ea6a0a271c4;
- 10472241b503438891b784e7f62a776282240e62.

PSR-10 | LACK OF ACCESS CONTROL

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 201, 243~247	● Acknowledged

Description

The following functions have no access control and can be called by anyone:

- `releaseFunds()` , this may be called to release funds prior to an action that will be executed by one of the receiving contracts. This may allow for potential rate manipulations if the contracts logic depends on the amount of tokens held by the contract.
- `updateAssetsState()` , this may be called after a user sends funds directly to the contract to increase the reserves for particular markets. In combination with `releaseFunds()` this can be used to send a large amount of tokens to one of receiving contracts. This may allow for potential rate manipulations if the contracts logic depends on the amount of tokens held by the contract.

Recommendation

We recommend considering restricting access to these functions or to ensure that all receiving contracts are safe from manipulations in their token balances.

Alleviation

`[Venus, 08/25/2023]` : The contracts targets (the contracts that will received the funds after executing releaseFunds) are identified in PSR-03:

- Venus Treasury
- Prime
- RiskFund and RiskFundConverter
- XVSVaultConverter

It shouldn't be any problem if these contracts receive funds.

This is the expected behavior, we want to avoid centralization issues, allowing anyone to invoke the releaseFunds and the updateAssetsState functions.

VBN-01 | UNPROTECTED INITIALIZER

Category	Severity	Location	Status
Coding Issue	● Minor	VBNBAdmin.sol (PR289-VBNBAdmin): 21	● Resolved

Description

`VBNBAdmin` does not protect its initializer. An attacker can call the initializer and assume ownership of the logic contract, whereby they can perform privileged operations that trick unsuspecting users into believing that they are the owner of the upgradeable contract.

Recommendation

We recommend calling `_disableInitializers()` in the constructor or giving the constructor the `initializer` modifier to prevent the intializer from being called on the logic contract.

Reference: https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable#initializing_the_implementation_contract

Alleviation

`[CertiK, 08/28/2023]` : The client made the recommended changes in commit: bc6fd4e27232562a80265e4575418e4c5fb8536f.

VPB-05 | INCORRECT `ReservesReduced` EVENT

Category	Severity	Location	Status
Inconsistency	● Minor	VToken.sol (PR207-VToken): 1332; VToken.sol (PR262-VToken): 1555	● Resolved

Description

In both the `venus-protocol` and `isolated-pools` repository it defines the event `ReservesReduced`

```
/**
 * @notice Event emitted when the reserves are reduced
 */
event ReservesReduced(address admin, uint reduceAmount, uint newTotalReserves);
```

Where the `admin` indicated the address that funds were sent to. However, both now send the funds to the protocol share reserve. In addition, in `venus-protocol`, the input `msg.sender` is used instead of `protocolShareReserve`.

Recommendation

We recommend updating the event so that it describes the updated functionality and in `venus-protocol` changing the input `msg.sender` to `protocolShareReserve`.

Alleviation

`[CertiK, 08/28/2023]`: The client made the recommended changes in commits:

- df0c07bbead2b7204873ff64719a6617503bfba9;
- a3dadf1bea14aac80fdd2d9bcf44564f3e86953e.

VPI-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	VBNBAdmin.sol (PR289-VBNBAdmin): 27~29; VToken.sol (PR262-VToken): 354	● Resolved

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent loss of those tokens.

In the contract `VToken` :

- `protcolShareReserve_` is not zero-checked before being used in the function `setProtocolShareReserve()` .

In the contract `VBNBAdmin` :

- `_vBNB` , `_protocolShareReserve` , and `_WBNB` are not zero-checked before being used in the function `initialize()` .

Recommendation

We recommend adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

`[CertiK, 08/28/2023]` : The client made the recommended changes in commits:

- e699b139ffe3bd2bb037b48df659811656f98a31;
- c2656c6f0b43a457366d64a4ea364044ed6f8e47.

PSR-04 | ACCESS CHECK DOES NOT FOLLOW CONVENTION

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 161	● Resolved

Description

The function `addOrUpdateDistributionConfigs()` , takes an array of `DistributionConfig` structs as input. However, the access check does not include brackets for the input, which is the convention used if an input is an array.

Recommendation

We recommend adding brackets to have the access check remain consistent.

Alleviation

`[CertiK, 08/28/2023]` : The client made the recommended changes in commit: 64e4d6ae569fcd5fa81ae917bf0b4693b955f492.

PSR-05 | USING BOTH `uint` AND `uint256`

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 167, 172, 204, 222, 277, 290, 298, 298, 301, 310, 310, 312, 324, 324, 325, 355, 355, 357, 362	● Resolved

Description

Although `uint` and `uint256` are the same, we recommend keeping the consistancy of the code style and using the explicit version of `uint` .

Recommendation

We recommend changing `uint` at the aforementioned lines to `uint256` .

Alleviation

`[CertiK, 08/28/2023]` : The client made the recommended changes in commit: 1320fff4f7d92f3da41a4d5f99edde6e02722012.

PSR-08 | INCOMPLETE NATSPEC COMMENTS

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 110, 197~200, 238~242, 296, 340~344, 353, 368	● Resolved

Description

Some of the NatSpec comments are missing parameters and many functions do not have any NatSpec comments.

Recommendation

We recommend adding in the missing NatSpec comments to improve the readability of the codebase.

Alleviation

`[CertiK, 09/08/2023]` : The client made the recommended changes in commits:

- 95b2ff4902b30434cef21954b2ce04a61d275564;
- 92cc2491a0e1b928b2b968f5c9e729461dcca891.

VBN-02 | CODE DOES NOT FOLLOW SPECIFICATION

Category	Severity	Location	Status
Inconsistency	● Informational	VBNBAdmin.sol (PR289-VBNBAdmin): 27	● Resolved

Description

In the documentation provided it states the following for `VBNBAdmin` :

Storage attributes:

- `ProtocolShareReserve` address. Upgradable only by the admin
- `vBNB` address. Upgradable only by the admin

However, there is no setter function for `vBNB` and the setter function for `protocolShareReserve` is restricted via the `accessControlManager` .

Recommendation

We recommend ensuring the code meets the desired specifications.

Alleviation

`[Venus, 08/25/2023]` : ProtocolShareReserve address. Upgradable only by the admin. Fixed here: https://github.com/VenusProtocol/venus-protocol/commit/04d5e1c252801e14a7f056f554b556e2abcd5e63.

vBNB address. Upgradable only by the admin: we finally decided to make it immutable. We updated internally the documentation.

VPB-03 | TYPOS AND INCONSISTENCIES

Category	Severity	Location	Status
Inconsistency	● Informational	VToken.sol (PR207-VToken): 801~804; contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 22~23; VBep20.sol (PR262-VBep20): 230; VToken.sol (PR262-VToken): 243, 332, 345, 410~414, 519~522, 557	● Resolved

Description

In the contract `ProtocolShareReserve` :

- The comment above the `enum Schema` states the first schema is for spread income from prime markets in the core protocol. However, the first entry is `DEFAULT` which should be the schema for all the other sources of income.

In the contract `VToken` of `venus-protocol` :

- The comment above the function `_setInterestRateModel()` states: "Admin function to accrue interest and update the interest rate model". However, it is no longer an admin function and access is given by the access control manager.
- The comment above the functions `setReduceReservesBlockDelta()` and `setProtocolShareReserve()` use "A admin", which should be "An admin".
- The comment above the function `accrueInterest()` does not reflect the added functionality of psuedo-automatically reducing the reserves.
- The comment above the function `_reduceReserves()` states: "Accrues interest and reduces reserves by transferring to admin", however they are now transferred to the protocol share reserve contract.
- The `setReduceReservesBlockDelta()` was changed so that access must be granted through the ACM. However, the comment still refers to it as an admin function as opposed to a governance function.

In the contract `VBep20` of `venus-protocol` :

- In the function `doTransferOut()` , "complaint" is misspelled and should be "compliant".

In the contract `VBNBAdmin`

- The `onlyOwner` modifier is used for the `fallback()` function, however, a require statement is used for the `setProtocolShareReserve()` function instead of the `onlyOwner` modifier.

Inconsistency Between Repos

- The check for `reduceReservesBlockDelta` is inconsistent between `isolated-pools` and `venus-protocol`. The check in `isolated-pools` uses greater than, while `venus-protocol` uses greater than or equal to.

Recommendation

We recommend fixing the typos and inconsistencies mentioned above.

Alleviation

`[CertiK, 08/28/2023]`: The client made the recommended changes in commits:

- 11b5787e33cb9b973c64bd15ba0183b067c0ca71;
- e8fdf1b580a176229cebe743d84d26ec79eeeb8b;
- 9e0c1510e0703cb0376fd2ac4b4bc4cdb8f11d58;
- fc1178a42f5179b3ce379900fd86a883bd8b2329;
- 348955dae2ba4728fd995a2abdf04ff6db0d3914.

OPTIMIZATIONS | VENUS - ALLOCATION OF INCOME

ID	Title	Category	Severity	Status
PSP-01	Unnecessary Storage Read Access In For Loop	Coding Issue	Optimization	● Resolved
PSR-06	Unchecked Blocks Can Optimize Contract	Gas Optimization	Optimization	● Resolved
PSR-07	Can Use Strict Inequality To Save Gas	Gas Optimization	Optimization	● Resolved
PSR-11	Potential Out-Of-Gas Exception	Logical Issue	Optimization	● Resolved
VBB-01	Variables That Could Be Declared As Immutable	Code Optimization	Optimization	● Resolved
VPB-01	Emitted Events Can Be Optimized	Gas Optimization	Optimization	● Resolved
VTV-02	Check Can Optimize `_reduceReservesFresh()`	Code Optimization	Optimization	● Resolved

PSP-01 | UNNECESSARY STORAGE READ ACCESS IN FOR LOOP

Category	Severity	Location	Status
Coding Issue	● Optimization	contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve-Update2): 185, 249, 276, 442	● Resolved

Description

The for loop contains repeated storage read access in the condition check. Given that the ending condition does not change in the for loop, the repeated storage read is unnecessary, and its associated high gas cost can be eliminated.

```
185            for (uint256 j = 0; j < distributionTargets.length; ) {
```

Loop condition `j < distributionTargets.length` accesses the `length` field of a storage array.

```
249        for (uint256 i = 0; i < distributionTargets.length; ) {
```

Loop condition `i < distributionTargets.length` accesses the `length` field of a storage array.

```
276        for (uint256 i = 0; i < distributionTargets.length; ) {
```

Loop condition `i < distributionTargets.length` accesses the `length` field of a storage array.

```
442        for (uint256 i = 0; i < distributionTargets.length; ) {
```

Loop condition `i < distributionTargets.length` accesses the `length` field of a storage array.

Recommendation

Storage access costs substantially more gas than memory and stack access. We recommend caching the variable used in the condition check of the for loop to avoid unnecessary storage access.

Alleviation

`[CertiK]` : The team made changes resolving the finding in commit 762c054693c39a693fbaa2bdb1680635146f79c1.

PSR-06 | UNCHECKED BLOCKS CAN OPTIMIZE CONTRACT

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 167, 172, 204, 222, 277, 290, 301, 312, 325, 357, 362	● Resolved

Description

In general, the counter in a for loop can be incremented or decremented in an unchecked block.

Recommendation

We recommend adding these unchecked blocks to save gas.

Alleviation

`[CertiK, 08/28/2023]` : The client made the recommended changes in commits:

- f02e8ca0b4686e44bdcd45eada3bf57c9b23846b;
- 3e4e0135b2f9ff959cdd619b2640a006fe4fd91e.

PSR-07 | CAN USE STRICT INEQUALITY TO SAVE GAS

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 362	● Resolved

Description

In the function `ensurePercentages()` , the for loop iterates provided `schemaValue <= totalSchemas - 1` . However, this can instead check that the `schemaValue` is strictly less than `totalSchemas` to save gas.

Recommendation

We recommend using a strict inequality to save gas and to remain consistent.

Alleviation

`[CertiK, 08/25/2023]` : The client made the recommended changes in commit: e9ae778b5d6580ac46f8dfaccf4342f567004e16.

PSR-11 | POTENTIAL OUT-OF-GAS EXCEPTION

Category	Severity	Location	Status
Logical Issue	● Optimization	contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 167, 172, 204, 222, 277, 290, 312, 357	● Resolved

Description

When a loop allows an arbitrary number of iterations or accesses state variables in its body, the function may run out of gas and revert the transaction.

```
167         for (uint i = 0; i < configs.length; ++i) {
```

Function `ProtocolShareReserve.addOrUpdateDistributionConfigs` contains a loop and its loop condition depends on parameters: `configs` .

```
172             for (uint j = 0; j < distributionTargets.length; ++j) {
```

Function `ProtocolShareReserve.addOrUpdateDistributionConfigs` contains a loop and its loop condition depends on state variables: `distributionTargets` .

```
204         for (uint i = 0; i < assets.length; ++i) {
```

Function `ProtocolShareReserve.releaseFunds` contains a loop and its loop condition depends on parameters: `assets` .

```
222         for (uint i = 0; i < distributionTargets.length; ++i) {
```

Function `ProtocolShareReserve.getUnreleasedFunds` contains a loop and its loop condition depends on state variables: `distributionTargets` .

```
277         for (uint i = 0; i < markets.length; ++i) {
```

Function `ProtocolShareReserve._accrueAndReleaseFundsToPrime` contains a loop and its loop condition depends on external calls: `IPrime(prime).allMarkets` .

```
290         for (uint i = 0; i < markets.length; ++i) {
```

Function `ProtocolShareReserve._accruePrimeInterest` contains a loop and its loop condition depends on external calls: `IPrime(prime).allMarkets`.

```
        for (uint i = 0; i < distributionTargets.length; ++i) {
```

Function `ProtocolShareReserve._releaseFund` contains a loop and its loop condition depends on state variables: `distributionTargets`.

```
        for (uint i = 0; i < distributionTargets.length; ++i) {
```

Function `ProtocolShareReserve._ensurePercentages` contains a loop and its loop condition depends on state variables: `distributionTargets`.

Recommendation

We recommended either:

1. Placing limitations on the loop's bounds.
2. Ensuring these loops will not exceed the gas limit with the expected (current and future) state of the protocol.

Alleviation

`[Certik, 08/25/2023]`: The client added some limits for the loops in commits:

- 24a956611704ef455361b09bede8df5abcda4bfe;
- e3146d143956e992c25b811d65873f7279cb9400;

In addition, they provided the following statement:

`[Venus, 08/25/2023]`: The approach we have followed to mitigate this potential issue was:

- when the for loop is on a parameter, the transaction's sender is responsible for providing a param small enough to avoid out-of-gas issues. Adding an extra check and reverting the transaction if the param is too big would have a similar effect.
- when the for loop is on a state variable (like distributionTargets), we have added checks to avoid having a state variable with too many items (invoking _ensureMaxLoops(distributionTargets.length) at the end of addOrUpdateDistributionConfigs).
- when the for loop depends on an external call (like IPrime(prime).allMarkets), we should add checks in the origin, to avoid having a too big collection.

VBB-01 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Code Optimization	● Optimization	VBNBAdminStorage.sol (PR289-VBNBAdminStorage): 28~32	● Resolved

Description

Both `vBNB` and `WBNB` will be known prior to deployment and are only assigned when initializing the contract. These can be made immutable variables and instead initialized in the constructor.

Recommendation

We recommend making these variables immutable or adding methods to change them.

Alleviation

`[CertiK, 08/25/2023]` : The client made the recommended changes in commit: bc6fd4e27232562a80265e4575418e4c5fb8536f.

VPB-01 | EMITTED EVENTS CAN BE OPTIMIZED

Category	Severity	Location	Status
Gas Optimization	● Optimization	VToken.sol (PR207-VToken): 636~638; contracts/ProtocolReserve/ProtocolShareReserve.sol (ProtocolShareReserve): 141, 153, 177~180; VBNBAdmin.sol (PR289-VBNBAdmin): 43~45	● Resolved

Description

In the contract `ProtocolShareReserve` :

In the function `setPoolRegistry()` , the event `PoolRegistryUpdated` uses `poolRegistry` when `_poolRegistry` can be used in its place to save gas. Alternatively, the event can be emitted before setting `poolRegistry` to `_poolRegistry` and the temporary variable `oldPoolRegistry` can be removed. The latter is slightly more gas efficient, but may not be consistent with the typical convention used throughout the codebase.

Similarly, in the function `setPrime()` , the event `PrimeUpdated` uses `prime` when `_prime` can be used in its place to save gas. Alternatively, the event can be emitted before setting `prime` to `_prime` and the temporary variable `oldPrime` removed. The latter is slightly more gas efficient, but may not be consistent with the typical convention used throughout the codebase.

In the function `addOrUpdateDistributionConfigs()` , the event `DistributionConfigUpdated` can use `_config.destination` and `_config.schema` as that are checked to be the same as `config.destination` and `config.schema` .

In the contract `VBNBAdmin` :

In the function `setProtocolShareReserve()` , the event `ProtocolShareReserveUpdated` uses `protocolShareReserve` when `protocolShareReserve_` can be used in its place to save gas. Alternatively, the event can be emitted before setting `protocolShareReserve` to `protocolShareReserve_` and the temporary variable `oldProtocolShareReserve` can be removed. The latter is slightly more gas efficient, but may not be consistent with the typical convention used throughout the codebase.

In the contract `VToken` in `isolated-pools` :

In the function `setReduceReservesBlockDelta()` , the event `NewReduceReservesBlockDelta` can be emitted before setting `reduceReservesBlockDelta` to `_newReduceReservesBlockDelta` and the temporary variable `oldReduceReservesBlockDelta_` can be removed. This is slightly more gas efficient, but may not be consistent with the typical convention used throughout the codebase.

Recommendation

We recommend adding the optimizations above.

Alleviation

`[CertiK, 08/28/2023]` : The client made the recommended changes in commits:

- 9b894de0031557d9241a412efd18a0220598c562;
- 1992f805e6abe10fc957f357ab5d0192460e86d4;
- ac25981a089080391bb9cbea75585ede388bad7f.

VTV-02 | CHECK CAN OPTIMIZE `_reduceReservesFresh()`

Category	Severity	Location	Status
Code Optimization	● Optimization	VToken.sol (PR262-VToken): 1518~1558	● Resolved

Description

The function `_reduceReservesFresh()` is now called psuedo-automatically after the `reduceReservesBlockDelta` of blocks has passed. This may be called frequently enough so that the input `reduceAmount` is 0. Thus a check if the input `reduceAmount` is 0 that will simply return can be added to avoid executing unnecessary logic.

Recommendation

We recommend adding a check if the `reduceAmount` is zero to avoid unnecessary logic in this case.

Alleviation

`[CertiK, 08/25/2023]` : The client made the recommended changes in commit: 0695114b8d10fcd46a3abaa76ee397b29bc63e4e.

APPENDIX | VENUS - ALLOCATION OF INCOME

Finding Categories

Categories	Description
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.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

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 | CERTIK

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.

CertiK | Securing the Web3 World

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.