CERTIK

Venus Labs - Leverage Strategies Manager

Security Assessment

CertiK Assessed on Dec 10th, 2025

CertiK Assessed on Dec 10th, 2025

Venus Labs - Leverage Strategies Manager

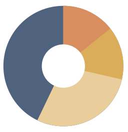
The security assessment was prepared by CertiK.

Executive Summary

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

LANGUAGE	TIMELINE
Solidity	Preliminary comments published on 12/03/2025 Final report published on 12/10/2025

Vulnerability Summary



Total Findings	Resolved	Timelock	Partially Resolved	Acknowledged	Declined
14	11	1	0	2	0

🟧 2	Centralization	1 Timelock, 1 Acknowledged	Centralization findings highlight privileged roles & functions and their capabilities, or instances where the project takes custody of users' assets.
🟥 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.
🟧 0	Major		Major risks may include logical errors that, under specific circumstances, could result in fund losses or loss of project control.
🟨 2	Medium	2 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨 4	Minor	3 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.
🟦 6	Informational	6 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 LABS - LEVERAGE STRATEGIES MANAGER

Summary

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

Overview

PR-12

Dependencies

Third Party Dependencies

Recommendations

Findings

VLL-02 : Centralized Control Of Contract Upgrade

VLL-07 : Centralization Related Risks

VLL-03 : Account Saftey Check May Not Account For Unaccrued Interest

VLL-08 : Exiting Leverage May Not Account For Potential Treasury Percentage

VLL-04 : Maximum Flashloan Amounts Do Not Handle Interest Accrual

VLL-09 : Logic Cannot Handle Fee On Transfer Tokens

VLL-10 : Contract Is Not Compatible With VBNB

VLL-13 : Excess Tokens Protecting Against Price Volatility Will Be Lost

VLL-01 : Discussion On Intended Protocol Support

VLL-05 : Revert Does Not Include Error Code

VLL-06 : Discussion On Leverage Limit

VLL-11 : Potential Reentrancy Risks

VLL-12 : Typos And Inconsistencies

VLL-14 : Unused Import

Appendix

Disclaimer

CODEBASE | VENUS LABS - LEVERAGE STRATEGIES MANAGER

Repository

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

Commit

Base: 6a4ba3e01b38ddc96673490c6b407b8d28769bb9
Update1: d6e1e95f247199489f35cc2621521874754eb683

AUDIT SCOPE | VENUS LABS - LEVERAGE STRATEGIES MANAGER

VenusProtocol/venus-periphery
LeverageStrategiesManager.sol
ILeverageStrategiesManager.sol

APPROACH & METHODS | VENUS LABS - LEVERAGE STRATEGIES MANAGER

This audit was conducted for Venus Labs to evaluate the security and correctness of the smart contracts associated with the Venus Labs - Leverage Strategies Manager project. The assessment included a comprehensive review of the in-scope smart contracts. The audit was performed using a combination of Manual Review and Static Analysis.

The review process emphasized the following areas:

- Architecture review and threat modeling to understand systemic risks and identify design-level flaws.
- Identification of vulnerabilities through both common and edge-case attack vectors.
- Manual verification of contract logic to ensure alignment with intended design and business requirements.
- Dynamic testing to validate runtime behavior and assess execution risks.
- Assessment of code quality and maintainability, including adherence to current best practices and industry standards.

The audit resulted in findings categorized across multiple severity levels, from informational to critical. To enhance the project's security and long-term robustness, we recommend addressing the identified issues and considering the following general improvements:

- Improve code readability and maintainability by adopting a clean architectural pattern and modular design.
- Strengthen testing coverage, including unit and integration tests for key functionalities and edge cases.
- Maintain meaningful inline comments and documentations.
- Implement clear and transparent documentation for privileged roles and sensitive protocol operations.
- Regularly review and simulate contract behavior against newly emerging attack vectors.

OVERVIEW | VENUS LABS - LEVERAGE STRATEGIES MANAGER

This audit concerns the changes made in the in scope files outlined in the following PR:

- PR-12

Note that any centralization risks present in the existing codebase before these PRs were not considered in this audit and only those added in these PRs are addressed in the audit. We recommend all users carefully review the centralization risks, much of which can be found in our previous audits, which can be found here: https://skynet.certik.com/projects/venus.

PR-12

PR-12 introduces flash-loan–based shortcut functionalities within the core pool enabling users to reach their target leveraged position in a single transaction, eliminating the need for multiple swap-and-borrow iterations and thereby reducing gas overhead.

DEPENDENCIES | VENUS LABS - LEVERAGE STRATEGIES MANAGER

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:

- Oracles
- ERC20 Tokens
- Swap protocols

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 the 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 LABS - LEVERAGE STRATEGIES MANAGER

14	0	2	0	2	4	6
Total Findings	Critical	Centralization	Major	Medium	Minor	Informational

This report has been prepared for Venus Labs to identify potential vulnerabilities and security issues within the reviewed codebase. During the course of the audit, a total of 14 issues were identified. Leveraging a combination of Manual Review & Static Analysis the following findings were uncovered:

ID	Title	Category	Severity	Status
VLL-02	**Centralized Control Of Contract Upgrade**	**Centralization**	**Centralization**	● **1h Timelock**
VLL-07	**Centralization Related Risks**	**Centralization**	**Centralization**	● **Acknowledged**
VLL-03	Account Saftey Check May Not Account For Unaccrued Interest	Logical Issue	Medium	● Resolved
VLL-08	Exiting Leverage May Not Account For Potential Treasury Percentage	Logical Issue	Medium	● Resolved
VLL-04	Maximum Flashloan Amounts Do Not Handle Interest Accrual	Logical Issue	Minor	● Resolved
VLL-09	Logic Cannot Handle Fee On Transfer Tokens	Inconsistency	Minor	● Acknowledged
VLL-10	Contract Is Not Compatible With VBNB	Logical Issue	Minor	● Resolved
VLL-13	Excess Tokens Protecting Against Price Volatility Will Be Lost	Logical Issue	Minor	● Resolved
VLL-01	Discussion On Intended Protocol Support	Coding Style	Informational	● Resolved
VLL-05	Revert Does Not Include Error Code	Coding Style	Informational	● Resolved
VLL-06	Discussion On Leverage Limit	Logical Issue	Informational	● Resolved

ID	Title	Category	Severity	Status
VLL-11	Potential Reentrancy Risks	Logical Issue	Informational	● Resolved
VLL-12	Typos And Inconsistencies	Coding Issue, Inconsistency	Informational	● Resolved
VLL-14	Unused Import	Code Optimization	Informational	● Resolved

VLL-02 | Centralized Control Of Contract Upgrade

Category	Severity	Location	Status
Centralization	● Centralization	LeverageStrategiesManager.sol (Base): 18	● 1h Timelock

Description

The proxy admin has the authority to update the implementation contract LeverageStrategiesManager.

Any compromise to the proxy admin account may allow a hacker to take advantage of this authority and change the implementation contract which is pointed by proxy and therefore execute potential malicious functionality in the implementation contract.

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 Labs, 12/08/2025]: Issue acknowledged. I won't make any changes for the current version.

After deployment LeverageStrategiesManager owner will be set to the Venus NORMAL_TIMELOCK and it will take ownership of it in the VIP process.

VLL-07 | Centralization Related Risks

Category	Severity	Location	Status
Centralization	● **Centralization**	**LeverageStrategiesManager.sol (Base): 131, 183, 236**	● **Acknowledged**

Description

The functions `enterLeverage()` , `enterLeverageFromBorrow()` , and `exitLeverage()` include input swap data that is passed to the swap helper to execute a swap. It is assumed that this will call `multicall()` which requires a signature from the backend signer. Any compromise to the backend signer can allow arbitrary external calls to any contract during the leverage process.

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 Labs, 12/08/2025]: Issue acknowledged. We acknowledge risk the backend signer being compromised. Any generic calls made during a compromised multicall would be called from SwapHelper context and must result with sending expected tokens back to the LeverageStrategiesManager otherwise it would be reverted.

VLL-03 | Account Saftey Check May Not Account For Unaccrued Interest

Category	Severity	Location	Status
Logical Issue	● Medium	LeverageStrategiesManager.sol (Base): 710~714	● Resolved

Description

The function `_checkAccountSafe()` verifies that a user’s account is not in shortfall, performing this check before and after leveraged position changes. However, when called before leveraging, any unaccrued interest may not yet be accounted for, potentially allowing an unsafe account to pass the safety check. This discrepancy arises because interest accrual occurs during the leveraging process, not prior to the initial safety check. Consequently, the account's health status may be inaccurately assessed if unaccrued interest is significant.

Recommendation

We recommend ensuring interest is accrued prior to calling `_checkAccountSafe()` .

Alleviation

[CertiK, 12/08/2025]: The client made the recommended changes resolving this finding in commit d6e1e95f247199489f35cc2621521874754eb683.

VLL-08 | Exiting Leverage May Not Account For Potential Treasury Percentage

Category	Severity	Location	Status
Logical Issue	● Medium	LeverageStrategiesManager.sol (Base): 523, 567	● Resolved

Description

The function `_handleExitSingleAsset()` calculates the `flashLoanRepayAmount` and expects that the call `market.redeemUnderlyingBehalf(onBehalf, flashLoanRepayAmount)` will transfer at least the `flashLoanRepayAmount` to the contract.

However, if the `treasuryPercent()` is nonzero, then a portion of the redeemed amount is transferred to the treasury (**Link**), and only the remaining amount is then transferred to this contract. If the `treasuryPercent()` is nonzero, then this function will always revert as it will not redeem enough collateral so that the contract receives the `flashLoanRepayAmount`.

Note that similarly in the function `_handleExitCollateral()`, the `collateralAmountToRedeem` must take into account the treasury fee if the `treasuryPercent()` is nonzero. However, this amount is input by the user, so that users or the UI should properly take this into account when determining the input amount.

Recommendation

We recommend accounting for the `treasuryPercent()` when calculating the amount of collateral that must be redeemed to repay the flashloan.

Alleviation

[CertiK, 12/08/2025]: The client made the recommended changes resolving this finding in commit 87b6deccf746b9229f773518ee136e65b486f395.

VLL-04 | Maximum Flashloan Amounts Do Not Handle Interest Accrual

Category	Severity	Location	Status
Logical Issue	● Minor	LeverageStrategiesManager.sol (Base): 83, 129, 181, 234, 282	● Resolved

Description

When entering a leveraged position, a user must input an amount to flashloan to leverage the position with. In the extreme case, they can flashloan the maximum amount to have the maximum leverage for their position reaching their collateral factor limit. We assume this will be fetched via a UI, however, the fetched amount must depend on the account state at a specific block. If the account has existing borrows, so that it will accrue some borrow interest for each block, then this amount may not be correct as the user will not know the precise block that their transaction will be included in or if the interest rates may change. Users cannot predict the exact block their transaction will be included in, nor foresee interest rate changes, which can result in stale or insufficient input values when performing these operations through a UI.

Similarly, when exiting a leveraged position, the user must input an amount to flashloan in order to repay the debt for their position. In the extreme case a user may wish to flahloan their entire borrowed amount to completely deleverage, however, they will accrue some borrow interest for each block. In this case the operation may fail or leave a small residual debt because the interest accrued in the interim was not considered. Users cannot predict the exact block their transaction will be included in, nor foresee interest rate changes, which can result in stale or insufficient input values when performing these operations through a UI.

Recommendation

If maximum leverage is intended to be supported, then we recommend refactoring the code to allow for the maximum leverage while accounting for potential interest accrual. In addition, we recommend refactoring the code to ensure that users can completely deleverage their position while accounting for potential interest accrual.

Alleviation

[CertiK, 12/09/2025]: The client made the recommended changes in commit for exiting leverage positions cb521c8ff797f8c27821be2e150573a0b177cb6e. For entering leverage positions they stated that it is a "Known limitation by design."

VLL-09 | Logic Cannot Handle Fee On Transfer Tokens

Category	Severity	Location	Status
Inconsistency	● Minor	LeverageStrategiesManager.sol (Base): 684~687	● Acknowledged

Description

The function `_borrowAndRepayFlashLoanFee()` uses the `borrowMarket` balance of borrowed assets to determine the amount obtained during `borrowBehalf` to repay the flash loan. However, this can cause issues if tokens that have a fee on transfer are used as the contract will not receive this full amount.

The function `_handleEnterSingleAsset()` assumes it receives the `collateralAmount` , however, this is the input amount and it does not check the actual amount received by the contract.

The function `_handleEnterCollateral()` assumes it receives the `collateralAmount` , however, this is the input amount and it does not check the actual amount received by the contract.

The function `_handleEnterBorrow()` assumes it receives the `borrowedAmountSeed` , however, this is the input amount and it does not check the actual amount received by the contract.

Furthermore, the flashloan amounts sent in the callback correspond to the input flashloaned amount and does not check the actual amount received by the contract during the flashloan.

Recommendation

We recommend ensuring no fee on transfer tokens are supported by the protocol. In addition, for extra security we recommend instead using the balance of `address(this)` to determine the actual amount of borrowed asset received by the contract in the function `_borrowAndRepayFlashLoanFee()` .

Alleviation

[Venus Labs, 12/08/2025]: Issue acknowledged. We do not currently support fee-on-transfer tokens for Venus markets and LeverageStrategiesManager.

VLL-10 | Contract Is Not Compatible With VBNB

Category	Severity	Location	Status
Logical Issue	● Minor	LeverageStrategiesManager.sol (Base): 593, 602, 624, 635, 652	● Resolved

Description

The contract is designed to interact with markets whose underlying token is an ERC20 token. As such, it is not designed to be compatible with `vBNB` .

Recommendation

We recommend adding a check that the input collateral and borrow markets are not `vBNB` .

Alleviation

[CertiK, 12/08/2025]: The client made the recommended changes resolving this finding in commit 46069b931342eedd48baf3fb627699096473a431.

VLL-13 | Excess Tokens Protecting Against Price Volatility Will Be Lost

Category	Severity	Location	Status
Logical Issue	● Minor	LeverageStrategiesManager.sol (Base): 266	● Resolved

Description

In the function `exitLeverage()` , after the swap and repayment processes are completed, any remaining amount of the borrowed token is transferred to the treasury through the call to `_transferDustToTreasury(_borrowedMarket);` . This leftover can occur if users swap more than is required to cover the flashloan and associated fees, often as a precaution against volatile price movements that could otherwise cause their transaction to fail. As a result, users attempting to ensure the success of their transaction by sending excess tokens will not be reimbursed for the surplus; these funds are retained by the protocol treasury rather than returned to the user.

Recommendation

We recommend that users provided with a mechanism to receive any excess tokens remaining after the swap and repayment processes in exitLeverage(). Modify the logic so that, after all debts and fees are settled, any surplus funds are returned directly to the user rather than sent to the treasury. This will ensure users are not penalized for safeguarding their transaction against price volatility.

Alleviation

[CertiK, 12/08/2025]: The client updated the code so that excess amounts of the borrowed token are returned to the user instead of the treasury, resolving this finding in commit d5dc0ac92d347eaaf9fd9d6f492aa697e8698f33.

VLL-01 | Discussion On Intended Protocol Support

Category	Severity	Location	Status
Coding Style	● Informational	LeverageStrategiesManager.sol (Base): 724	● Resolved

Description

The `enterMarketBehalf()` function is currently implemented only for the core pool, as introduced in VIP 573. As a result, leverage strategies managed by this contract are exclusively compatible with the core pool and not with isolated pools.

Please confirm whether this contract is intended for use solely with the core pool or if future support for isolated pools is planned. If use is limited to the core pool, consider organizing the codebase to clearly separate protocol-specific logic, reducing the risk of misconfiguration or developer confusion.

Recommendation

We recommend answering the questions above.

Alleviation

[Venus Labs, 12/08/2025]: Issue acknowledged. The LeverageStrategiesManager contract is intended for use only with the Core Pool on BNB Mainnet.

VLL-05 | Revert Does Not Include Error Code

Category	Severity	Location	Status
Coding Style	● Informational	LeverageStrategiesManager.sol (Base): 712	● Resolved

Description

The function `_checkAccountSafe()` reverts with the `OperationCausesLiquidation` error when an account has a liquidity shortfall or when the comptroller returns an error. Currently, if the comptroller triggers a revert, the specific error code is not propagated within the error. This omission can make it challenging for developers and users to identify the root cause of the failure, complicating troubleshooting and resolution.

Recommendation

We recommend including the error code that may be emitted in `OperationCausesLiquidation`.

Alleviation

[CertiK, 12/08/2025]: The client made the recommended changes resolving this finding in commit 32512b2e3c3490e54f5c81ba9b4ff0c4c556c021.

VLL-06 | Discussion On Leverage Limit

Category	Severity	Location	Status
Logical Issue	● Informational	LeverageStrategiesManager.sol (Base): 80	● Resolved

Description

A users leverage is limited by the collateral factor (that is the percentage that can be borrowed), in general the leverage limit is given by the series `(collateral factor)^n` , which converges to a value of `1/(1 - collateral factor)` . We would like to ensure this is the intended leverage limit for the strategy.

Recommendation

We recommend answering the questions above.

Alleviation

[Venus Labs, 12/08/2025]: Issue acknowledged.

The leverage limit now corresponds to the maximum amount a user can borrow, based on the collateral factor of the asset they will use as collateral.

We confirm that (borrowing power) / (1 - collateral factor) is the intended limit for the strategy.

VLL-11 | Potential Reentrancy Risks

Category	Severity	Location	Status
Logical Issue	● Informational	LeverageStrategiesManager.sol (Base): 33~49, 597~601, 624, 635, 652	● Resolved

Description

The protocol makes external calls to third party ERC20 tokens and swapping protocols. If these third parties implement hooks or make further external calls, they may be used to reenter the contract or the Venus protocol.

For example, if a token that implements hooks is supported, when `_transferDustToInitiator()` is called it may be used to reenter the contract. In particular, this can cause events to be emitted out of order and can cause `_checkAccountSafe()` after the reentrant call is finished.

Recommendation

We recommend ensuring no tokens with hooks are supported by the protocol. In addition, for additional security the check `_checkAccountSafe()` and the emitting of the event can be performed prior to transferring dust. Furthermore, we recommend ensuring any swapping protocols that are supported do not make external calls that can be used to reenter the Venus protocol or that any such protocol level reentrant calls are carefully vetted to ensure their security.

Alleviation

[CertiK, 12/09/2025]: The client made the recommended changes in commit e7b0d0f1394502a90584cd24934d5b98a752d768.

VLL-12 | Typos And Inconsistencies

Category	Severity	Location	Status
Coding Issue, Inconsistency	● Informational	LeverageStrategiesManager.sol (Base): 703~704	● Resolved

Description

Here are some natspec typos in the project:

`_checkAccountSafe()` is not only used to check whether a position remains safe after leverage operations, but also *before* in the case of entering positions.

Recommendation

We recommend that the client write a natspec consistent with the use of the functions.

Alleviation

[CeriK, 12/09/2025]: The client updated the comment in commit 0b618eecdec463644dbb921489737b256adf73a8.

VLL-14 | Unused Import

Category	Severity	Location	Status
Code Optimization	● Informational	LeverageStrategiesManager.sol (Update1): 4, 18	● Resolved

Description

`Ownable2StepUpgradeable` is imported, however, its functionality is not used.

Recommendation

We recommend either removing the unused imports or including functionality that utilizes them.

Alleviation

[Venus Labs, 12/10/2025]: Issue acknowledged. I won't make any changes for the current version.

Retained for forward compatibility. Future versions may introduce owner-restricted functions (configurable fees, emergency pause, swap helper management). Removing now would require storage-layout-breaking upgrade to reintroduce later.

APPENDIX | VENUS LABS - LEVERAGE STRATEGIES MANAGER

Finding Categories

Categories	Description
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
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.
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.

DISCLAIMER | CERTIK

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.

Elevating Your Web3 Journey

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is the largest 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.