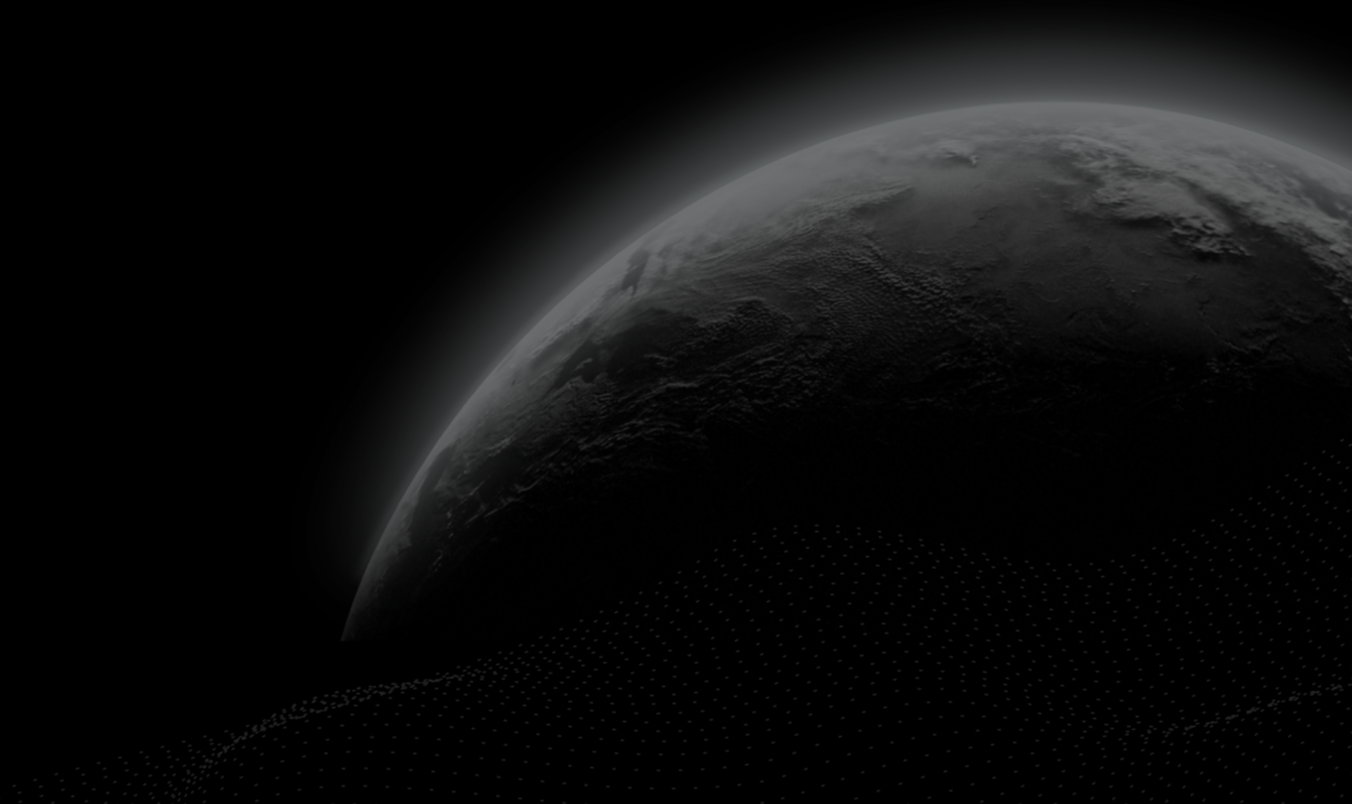
CERTIK

Security Assessment

Venus - Swap Router

CertiK Assessed on May 22nd, 2023



CertiK Assessed on May 22nd, 2023

Venus - Swap Router

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum (ETH)	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 05/22/2023	N/A

CODEBASE

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

...View All

COMMITS

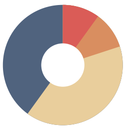
base: 2168c01c210ef9131369bf21e60d335cf3020725

update1: 346d32e59b64e7224302c6104f7c338fc7e38e60

update2: 46cea0c672626a5e53fdf0ebd9d534407f622a85

...View All

Vulnerability Summary



10	9	1	0	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

🟥 1	Critical	1 Resolved	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.
🟧 1	Major	1 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.
🟨 4	Minor	4 Resolved	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.
🟦 4	Informational	4 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 - SWAP ROUTER

CODEBASE | VENUS - SWAP ROUTER

Repository

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

Commit

base: 2168c01c210ef9131369bf21e60d335cf3020725

update1: 346d32e59b64e7224302c6104f7c338fc7e38e60

update2: 46cea0c672626a5e53fdf0ebd9d534407f622a85

AUDIT SCOPE | VENUS - SWAP ROUTER

13 files audited ● 1 file with Mitigated findings ● 3 files with Resolved findings ● 9 files without findings

ID	Repo	Commit	File	SHA256 Checksum
● SRS	VenusProtocol/venus-protocol	2168c01	SwapRouter.sol	f1704093204379bb2509afa17e30fa9f040bc9a46cc19569ef05a5aafacf9ebc
● RHS	VenusProtocol/venus-protocol	2168c01	RouterHelper.sol	2cbd0ecae1750c23650e515bfa4fe1f7e6e858dfee8d099854de5235dee23769
● PLS	VenusProtocol/venus-protocol	2168c01	lib/PancakeLibrary.sol	564b1a4966d00e41a76a251668c2fdec975c91305323c60cce896b0931803bc2
● THS	VenusProtocol/venus-protocol	2168c01	lib/TransferHelper.sol	794551ed6786dad623fed295a69648743dc4104737201456a241871f51fc637d
● IRH	VenusProtocol/venus-protocol	2168c01	IRouterHelper.sol	11ad497ca74697921f4c51b466558445dd855be84ec2b7a9c31493094977d020
● CES	VenusProtocol/venus-protocol	2168c01	interfaces/CustomErrors.sol	c1d8c80c5c624e7f33f18aa947c26db4259041709b9c0fbeecab35f081bfcc5b
● IPP	VenusProtocol/venus-protocol	2168c01	interfaces/IPancakePair.sol	78004f98a1651d708e0b9f047a7fe1f11c475f950928c102b9c698996dcae592
● IPS	VenusProtocol/venus-protocol	2168c01	interfaces/IPancakeSwapV2Factory.sol	5afb6644a6a4d3454a455aca3f2e3b8f9d41140ac4ab3bfbee0dc9e04fd73376
● IPV	VenusProtocol/venus-protocol	2168c01	interfaces/IPancakeSwapV2Router.sol	0ea252ccaa40b6301579beea9af948f061ac3b9e2c9a7ff256d15ab49cad96e5
● IVB	VenusProtocol/venus-protocol	2168c01	interfaces/IVBNB.sol	10f901fcf3e67e4812bbfe62c7ce0fea759ca560f36849e42fe9e874934b723c
● IVS	VenusProtocol/venus-protocol	2168c01	interfaces/IVtoken.sol	c1dcd57717c4273fa6d8c9fc6525e381e5374432e1b529af9e45aa8b5217add0
● IWB	VenusProtocol/venus-protocol	2168c01	interfaces/IWBNB.sol	ee40fd2540f8c351f58251b85cf4784187bb5a729ed6a88b1983ddbd07cd8549

ID	Repo	Commit	File	SHA256 Checksum
ICS	VenusProtocol/venus-protocol	2168c01	interfaces/InterfaceComptroller.sol	a6b9b0f1791fcd1c1f2b8d57afe7a9bedbacdcaba5624b05474877a1ee5ea98d

APPROACH & METHODS | VENUS - SWAP ROUTER

This report has been prepared for Venus to discover issues and vulnerabilities in the source code of the Venus - Swap Router 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 - SWAP ROUTER

RouterHelper

This contract is designed to contain all the helper functions needed to perform swaps via PancakeSwap. Its logic handles swaps that also have a respective swap supporting fee on transfers using the same function. This is done by taking an `enum` as input to indicate if the swap does or does not support fees on transfer. Importantly, if the swap is supporting fees, the output amount is not checked to be greater than the `amountOutMin` , thus any contract using these functions must check the output amount is at least the minimum amount. This is done for all such scenarios in `SwapRouter` .

SwapRouter

This contract is designed to interact with PancakeSwap to allow users to swap, swap then supply to the Venus protocol, or to swap and repay to the Venus Protocol. This helps users do this in a single transaction, as opposed to first using PancakeSwap to swap tokens and then having to use the Venus Protocol to supply or repay.

DEPENDENCIES | VENUS - SWAP ROUTER

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:

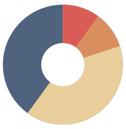
- PancakeSwap
- ERC20 Tokens

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. 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.

FINDINGS | VENUS - SWAP ROUTER



10	1	1	0	4	4
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Venus - Swap Router. Through this audit, we have uncovered 10 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
PLS-01	Incorrect Fee Amount Will Cause All Swaps To Fail	Logical Issue	Critical	● Resolved
SRS-01	**Centralization Risks In SwapRouter.Sol**	**Centralization / Privilege**	**Major**	● **Mitigated**
PLS-02	Incomplete Check	Logical Issue	Minor	● Resolved
SRS-02	Missing Checks	Logical Issue	Minor	● Resolved
SRS-03	Missing Zero Address Validation	Logical Issue	Minor	● Resolved
SWA-01	Potential Reentrancy (Out-Of-Order Events)	Volatile Code	Minor	● Resolved
RHS-01	Can Use `safeTransfer()`	Inconsistency	Informational	● Resolved
SRS-04	Comments For Functions Supporting Fee	Inconsistency	Informational	● Resolved
SRS-05	`Natspec` Comments Missing Parameters	Inconsistency	Informational	● Resolved
SRS-06	Incorrect Comment	Inconsistency	Informational	● Resolved

PLS-01 | INCORRECT FEE AMOUNT WILL CAUSE ALL SWAPS TO FAIL

Category	Severity	Location	Status
Logical Issue	● Critical	lib/PancakeLibrary.sol (base): 29, 71, 89	● Resolved

Description

The `PancakeLibrary` is used to interact with pairs created through the `PancakeRouter v2` (See 0x10ed43c718714eb63d5aa57b78b54704e256024e for deployed code). In `v1` the fee was 0.2%, however, the fees were changed in `v2` to be 0.25%. The logic here assumes that the fee is only 0.2%, when in fact a 0.25% fee will be taken. This means that the calculated `amountIn` will be less than the amount needed and the `amountOut` will be greater than the amount that can be received, causing the constant product to not be preserved and reverting the transaction.

Recommendation

We recommend changing the logic to account for the 0.25% fee.

Alleviation

`[CertiK]` : The client made the recommended changes in commit: e8c36766f0b6065a751c0e62383487d9ba49874f.

SRS-01 | CENTRALIZATION RISKS IN SWAPROUTER.SOL

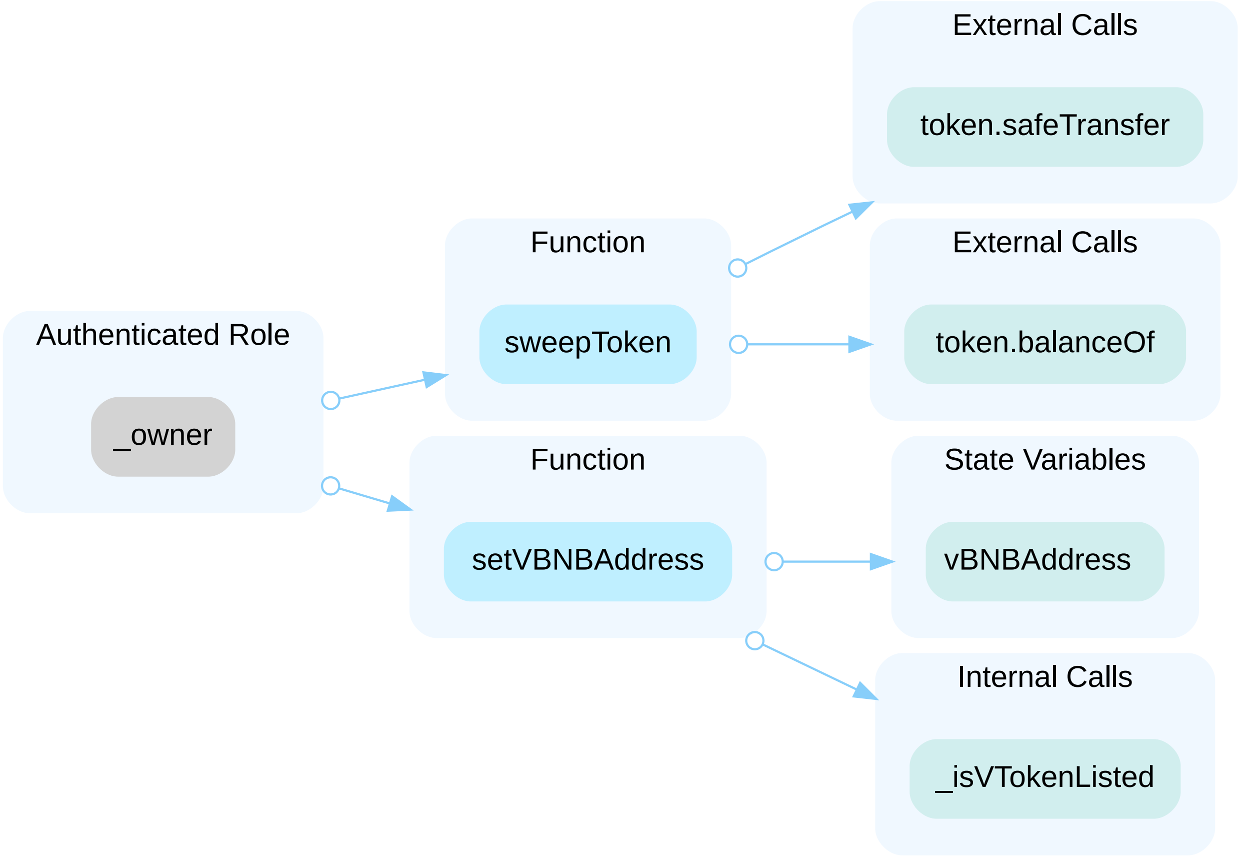
Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**SwapRouter.sol (base): 62, 661**	● **Mitigated**

Description

In the contract `SwapRouter` the role `WBNB` has authority to send `BNB` to the contract. If it is set to a malicious contract, then it can allow an attacker to send `BNB` to the contract. This is an immutable variable and `WBNB` is not an upgradeable contract, so this only needs to be checked once after deployment to be the correct address.

In the contract `SwapRouter` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and do the following:

- transfer any tokens held by the contract to any other address;
- set the `vBNBAddress` to any other listed `vToken`, which would revert or if the comptroller is compromised could be set to a malicious contract.



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

`[Venua]` : WBNB will be set to 0xbb4cdb9cbd36b01bd1cbaebf2de08d9173bc095c during the deployment, and we'll manually check it after that.

Regarding the `_owner`, it will be transferred to the address 0x939bD8d64c0A9583A7Dcea9933f7b21697ab6396 during the deployment, accepting the ownership in a VIP (Venus Improvement Proposal), voted by the community. This Timelock contract has a delay of 3 days (24 hours voting + 48 hours delay) before executing any change.

PLS-02 | INCOMPLETE CHECK

Category	Severity	Location	Status
Logical Issue	● Minor	lib/PancakeLibrary.sol (base): 68~70, 85~86	● Resolved

Description

The following check is performed in the function `getAmountIn()` and `getAmountOut()` :

```
if (reserveIn == 0 && reserveOut == 0) {
            revert InsufficientLiquidity();
        }
```

However, this check does not cover the case when either `reserveIn` or `reserveOut` is 0, while the other is nonzero.

Recommendation

We recommend checking if `reserveIn` is zero *or* `reserveOut` is zero.

Alleviation

`[CertiK]` : The client made the recommended changes in commit: 774eed1266561fb63951617e750c2d3cad370524.

SRS-02 | MISSING CHECKS

Category	Severity	Location	Status
Logical Issue	● Minor	SwapRouter.sol (base): 88, 109, 130, 151, 173, 193, 214, 235, 255, 275, 297, 318, 338, 357, 370, 393, 417, 439, 675~677, 690~692	● Resolved

Description

It is checked that the input `vTokenAddress` is listed, however, there is no check that the final element of the input `path` is the underlying of the `vToken`. As approval is given for the last element of the input `path` to either repay or supply, it should be checked that it corresponds to the underlying of the `vToken`.

It is not checked that the input `vBNBAddress` is listed. It should be checked that this is a listed market, or alternatively, the input can be removed and the address stored by the contract itself.

Recommendation

We recommend checking that the `underlying` of the input `vTokenAddress` is the same as the last element of the input `path`. In addition, we recommend either checking that the input `vBNBAddress` is listed or removing the input and storing the address in the contract to reference.

Alleviation

`[CertiK]`: The client made the recommended changes in commits:

- 99fc4b2014e4a44cd4d484454f46860ca39121f4;
- 346d32e59b64e7224302c6104f7c338fc7e38e60;
- 46cea0c672626a5e53fdf0ebd9d534407f622a85.

SRS-03 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Logical Issue	● Minor	SwapRouter.sol (base): 59	● Resolved

Description

In the `constructor()` , the input `_comptrollerAddress` is not checked if it is the zero address.

Recommendation

We recommend adding a check for the zero address.

Alleviation

`[CertiK]` : The client made the recommended changes in commit: 48eb87ca15b5dd373a6db112f9f777561f38cf54.

SWA-01 | POTENTIAL REENTRANCY (OUT-OF-ORDER EVENTS)

Category	Severity	Location	Status
Volatile Code	● Minor	RouterHelper.sol (base): 68, 96, 113, 114, 115, 117, 118, 119, 133, 134, 140, 141, 143, 144, 164~169, 170, 174~179, 180, 184, 186, 189, 191, 205~210, 211, 212, 227, 228, 229, 231, 232, 248~253, 254, 255, 257, 259; SwapRouter.sol (base): 315, 316, 354, 355, 445, 446, 664, 666; lib/TransferHelper.sol (base): 27, 35	● Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

- Reentrancy can occur during the swaps as they will make external calls to the receiver of the tokens, this will cause the corresponding swap events to be emitted out of order.
- If a token implements hooks that make external calls, then reentrancy can also occur during token transfers, which may cause events to be emitted out of order.
- Token addresses are also provided as inputs, so that it is possible that a token is provided that can cause reentrancy with any external call made to it (for example if `balanceOf`), which may cause events to be emitted out of order.

This finding is considered minor because the reentrancy only causes out-of-order events.

Recommendation

We recommend adding a lock to the swapping functions in the `SwapRouter` to prevent reentrancy and prevent any possible issues in future iterations due to reentrancy. As the contract is close to the size limit this would require refactoring of the code, however, much of the code repeats the same logic which can be placed in an internal function to reduce the contracts size.

Alleviation

`[CertiK]`: The client made the recommended changes in commits:

- 844f78d9c21416671ea5c79cae47181098428d16;
- b9c85efd4764ed1cb5099d929272b2a848a85b7c.

RHS-01 | CAN USE `safeTransfer()`

Category	Severity	Location	Status
Inconsistency	● Informational	RouterHelper.sol (base): 134, 228	● Resolved

Description

When transferring `WBNB` the call is checked to be successful via an assert statement. However, the `TransferHelper` library's functions can be used in place of this.

Recommendation

We recommend using `safeTransfer()` instead of checking the return value via an assert statement for consistency.

Alleviation

`[CertiK]` : The client made the recommended changes in commit: 9e7103e6e1c4b58e0a8d2eb0d598368912a279f7.

SRS-04 | COMMENTS FOR FUNCTIONS SUPPORTING FEE

Category	Severity	Location	Status
Inconsistency	● Informational	SwapRouter.sol (base): 92, 134, 218, 259, 384	● Resolved

Description

The comments above some of the functions that support fee on transfer tokens in the contract `SwapRouter` do not mention how it is intended to be used for fee on transfer tokens.

Recommendation

We recommend adding comments to these functions explaining their design to be used with fee on transfer tokens.

Alleviation

`[CertiK]` : The client made the recommended changes in commits:

- 4dae8d41b71aba8fe5243d551771c9d9ea2163d4;
- 346d32e59b64e7224302c6104f7c338fc7e38e60.

SRS-05 | `Natspec` COMMENTS MISSING PARAMETERS

Category	Severity	Location	Status
Inconsistency	● Informational	SwapRouter.sol (base): 83, 104, 125, 146, 168, 188, 209, 230, 250, 270, 292, 312, 333, 351, 374, 397, 421, 442, 661	● Resolved

Description

The `natspec` comments of many functions cited above are missing the parameter `deadline` .

The `natspec` comment for the function `sweepToken()` , is missing the parameter `to` .

Recommendation

We recommend adding a `natspec` comment for the missing parameters.

Alleviation

`[CertiK]` : The client made the recommended changes in commit: f04f5cebebe98d54fe77171df03ea7ea48efddb3.

SRS-06 | INCORRECT COMMENT

Category	Severity	Location	Status
Inconsistency	● Informational	SwapRouter.sol (base): 656	● Resolved

Description

The comment above `sweepToken` states that tokens are sent to admin (timelock). However, the tokens are sent to the input `to` address, which may not be the admin.

Recommendation

We recommend changing the comment to reflect that the tokens are sent to the input `to` address.

Alleviation

`[CertiK]` : The client made the recommended changes in the commits:

- ae30cdd868cfe370ca1fd7c2d11288c000892084;
- 346d32e59b64e7224302c6104f7c338fc7e38e60.

OPTIMIZATIONS | VENUS - SWAP ROUTER

ID	Title	Category	Severity	Status
PLS-03	Inefficient Checks	Logical Issue	Optimization	● Resolved
SRS-07	Can Use Single Address Input	Gas Optimization	Optimization	● Resolved
SWA-02	Unchecked Blocks Can Optimize Contract	Gas Optimization	Optimization	● Resolved
SWA-03	Custom Errors Can Be Used	Gas Optimization	Optimization	● Resolved

PLS-03 | INEFFICIENT CHECKS

Category	Severity	Location	Status
Logical Issue	● Optimization	lib/PancakeLibrary.sol (base): 53~56	● Resolved

Description

The following two checks are performed:

```
require(reserveA > 0 && reserveB > 0, "PancakeLibrary: INSUFFICIENT_LIQUIDITY");
```

which will revert if `reserveA` or `reserveB` is zero.

```
if (reserveA == 0 && reserveB == 0) {
            revert InsufficientLiquidity();
        }
```

which will revert if `reserveA` and `reserveB` are zero. Thus this check is not needed as the require statement covers it.

Recommendation

We recommend removing the following check:

```
if (reserveA == 0 && reserveB == 0) {
            revert InsufficientLiquidity();
        }
```

In addition, we recommend refactoring the require check to use custom errors.

Alleviation

`[CertiK]` : The client made the recommended changes in commit: 071474188855749eaa8a852835b19af1347e072d.

SRS-07 | CAN USE SINGLE ADDRESS INPUT

Category	Severity	Location	Status
Gas Optimization	● Optimization	SwapRouter.sol (base): 705~714, 721~724	● Resolved

Description

The functions `_checkForAmountOut()` and `_getSwapAmount()` have an input array of addresses `path` , however, only the last element is used. A single address can instead be taken as input to reduce gas costs. In addition, the last element of the path `path[path.length - 1]` is used multiple times in functions, storing this in a temporary variable and referencing the temporary variable can save gas.

Recommendation

We recommend using a single address as input and using a temporary variable to store the last element to save gas.

Alleviation

`[CertiK]` : The client made the recommended changes in commits:

- 99fc4b2014e4a44cd4d484454f46860ca39121f4;
- 346d32e59b64e7224302c6104f7c338fc7e38e60.

SWA-02 | UNCHECKED BLOCKS CAN OPTIMIZE CONTRACT

Category	Severity	Location	Status
Gas Optimization	● Optimization	RouterHelper.sol (base): 60, 75; lib/PancakeLibrary.sol (base): 104, 121	● Resolved

Description

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

Recommendation

We recommend incrementing the counter in an unchecked block to save gas.

Alleviation

`[CertiK]` : The client made the recommended changes in commit: b3542eb3a772aabeac78537d0e93c02c0594c3aa.

SWA-03 | CUSTOM ERRORS CAN BE USED

Category	Severity	Location	Status
Gas Optimization	● Optimization	SwapRouter.sol (base): 403, 588, 663, 713; lib/PancakeLibrary.sol (base): 56; lib/TransferHelper.sol (base): 10~13, 19~22, 28~31, 36	● Resolved

Description

Custom Errors are used throughout the codebase, however, string errors are still used in the codebase.

Recommendation

We recommend replacing the string errors with custom errors to reduce gas costs.

Alleviation

`[CertiK]` : The client made the recommended changes in commit: ebef2cd84a55222f2503ff427c1547d69f5cd864.

APPENDIX | VENUS - SWAP ROUTER

Finding Categories

Categories	Description
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.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Inconsistency	Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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.