Bug ID #1

Unchecked Transfer

Vulnerability Type
Improper Validation

Severity
Medium

Description
The contracts PancakeRouter.sol and PancakeRouter01.sol have a vulnerability related to unchecked transfer operations. Specifically, the return values of an external transfer call to IPancakePair(pair).transferFrom(msg.sender, pair, liquidity) are not validated. It's important to note that these tokens do not revert in case of a transfer failure and simply return false. If one of these tokens is employed, a deposit may not revert in the event of transfer failure, potentially allowing attackers to deposit tokens without consequences.

Affected Code
- https://github.com/9mmPro/v2-contracts/blob/main/PancakeRouter01.sol#L110
- https://github.com/9mmPro/v2-contracts/blob/main/PancakeRouter.sol#L137

Impacts
If a transfer fails (for example, due to insufficient balance or other reasons), not checking the return value means the contract will continue execution as if the transfer was successful. This could lead to a loss of funds as the contract might proceed with operations assuming the transfer was successful when it was not.

Remediation
It is recommended to use SafeERC20, or ensure that the transfer return value is checked.

Retest
-

Bug ID #2

Missing Zero Address Validations

Vulnerability Type
Missing Input Validation

Severity
Low

Description:
The contracts were found to be setting new addresses without proper validations for zero addresses.
Address type parameters should include a zero-address check otherwise contract functionality may become inaccessible or tokens burned forever.
Depending on the logic of the contract, this could prove fatal and the users or the contracts could lose their funds, or the ownership of the contract could be lost forever.

Affected Code

- https://github.com/9mmPro/v2-contracts/blob/main/PancakeRouter.sol#L24-L27
- https://github.com/9mmPro/v2-contracts/blob/main/PancakeRouter01.sol#L21-L24
- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeFactory.sol#L18-L20
- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeFactory.sol#L45
- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeFactory.sol#L50

Impacts
If address type parameters do not include a zero-address check, contract functionality may become unavailable or tokens may be burned permanently.

Remediation
Add a zero address validation to all the functions where addresses are being set.

Retest
-

Bug ID #3

Manipulation Of initial Token Address

Vulnerability Type
Access Control

Severity
Low

Description
During a manual review, it was observed that the contract PancakePair.sol allows a factory user to initialize token0 and token1 at any time by making an external call. Malicious activity can be done to manipulate the functions within the contract i.e. sync, skim, swap, burn, and mint and the intended operation can be bypassed because all these functions have a dependency on token0 and token1.

Affected Code

- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakePair.sol#L191-L196

Impacts
Allowing external parties to initialize token0 and token1 at any time opens the door to potential security vulnerabilities. Malicious actors can exploit this capability to manipulate the contract's functions, potentially causing financial loss or other harm.

Remediation
It is recommended to move the initialization of token0 and token1 to constructor() so that it can be called once at the time of deployment by the contract owner. In case to continue with the initialize function, it is recommended to declare the initialize function as internal and the function call should be done within constructor().

Retest
-

Bug ID #4

MultiCall May lead to DOS

Vulnerability Type
Dos

Severity
Low

Description
Within the contract PancakePair.sol, multiple calls are being executed within the same transaction. This call is performed immediately following another call within the same transaction. There exists a possibility that this call may never be executed if a prior call fails permanently, potentially leading to a Denial-of-Service (DOS) vulnerability. Such a situation might be intentionally triggered by a malicious call.

Affected Code

- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakePair.sol#L67-L71

Impacts
A successful DOS attack on the contract can disrupt the normal operation of the platform, rendering it temporarily or even permanently non-functional. This can lead to financial losses and damage the reputation of the project.

Remediation
The report identifies a potential DOS vulnerability in the contract PancakePair.sol. This can be exploited maliciously or even occur unintentionally. The user's call to skim() function will fail.

Retest
-

Bug ID #5

Weak Pseudo-Random Number Generation

Vulnerability Type
Weak PRNG

Severity
Low

Description
During a manual review, we noticed the use of block.timestamp in PancakePair.sol contract function _update() uses a weak pseudo-random number generation i.e. uint32(block.timestamp % 2 ** 32). The contract developers should be aware that this does not mean the current time. Miners can influence the value of block.timestamp to perform Maximal Extractable Value (MEV) attacks. The use of block.timestamp creates a risk that time manipulation can be performed to manipulate price oracles. Miners can modify the timestamp by up to 900 seconds.

Affected Code

- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakePair.sol#L74-L87

Impacts
The vulnerability may allow malicious miners to manipulate the contract's price oracles by altering the timestamp, potentially leading to misleading or erroneous price information.

Remediation
To mitigate the vulnerability and lower the risk of MEV attacks, it is strongly advised not to rely on "now" or "block.timestamp" as sources of randomness. Instead, consider utilizing "block.number" in place of "block.timestamp" or "now" in your code. This substitution minimizes the susceptibility to MEV attacks, enhancing the overall security and integrity of the contracts and libraries.

Retest
-

Bug ID #6

Missing Return Value Handling

Vulnerability Type
Improper Validation

Severity
Low

Description
During a manual review, we noticed return values of external function calls are not handled. This behavior is observed within the contracts PancakeRouter.sol and PancakeRouter01.sol, where certain external methods are invoked, and their return values are disregarded. Specifically, it has been noted that the function _addLiquidity() in both contract PancakeRouter.sol and PancakeRouter02.sol neglects to account for the return value of IPancakeFactory(factory).createPair(tokenA, tokenB).

Affected Code
- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L44
- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L41

Impacts
Failing to check return values can potentially lead to unexpected crashes or errors in the contract's execution, which impacts the overall reliability of the platform.

Remediation
To enhance the robustness of the contract and prevent unexpected crashes, it is strongly recommended to introduce return value checks. By validating the return values of external function calls, the contract can better handle exceptions and errors that may arise during execution. This proactive approach ensures a more reliable and secure operation.

Retest
-

Bug ID #7

Incorrect Usage of blockhash

Vulnerability Type
Business Logic

Severity
Low

Description
The blockhash() is used to return the hash of the given block. This only works for the 256 most recent blocks, excluding the current block.
it always returns 0 for the current block, i.e. blockhash(block.number) always equals 0. This pattern was found being used in the Multicall2 contract.

Affected Code

- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/Multicall2.sol#L71

Impacts
The function will always return bytes32(0) for the return value of blockHash. This could lead to incorrect calculations and assumptions.

Remediation
Check the code as to why it is calculating the blockhash and returning 0 for every call. If the intention is to return 0 always, then handle that appropriately wherever this will be used. Note that the blockhash only returns values for 256 most recent blocks except the current block.

Retest
-

Bug ID #8

Use Ownable2Step

Vulnerability Type

Missing Best Practices

Severity

Low

Description

The "Ownable2Step" pattern is an improvement over the traditional "Ownable" pattern, designed to enhance the security of ownership transfer functionality in a smart contract. Unlike the original "Ownable" pattern, where ownership can be transferred directly to a specified address, the "Ownable2Step" pattern introduces an additional step in the ownership transfer process. Ownership transfer only completes when the proposed new owner explicitly accepts the ownership, mitigating the risk of accidental or unintended ownership transfers to mistyped addresses.

Affected Code

- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeZapV1.sol#L18-L846

Impacts

Without the "Ownable2Step" pattern, the contract owner might inadvertently transfer ownership to an unintended or mistyped address, potentially leading to a loss of control over the contract. By adopting the "Ownable2Step" pattern, the smart contract becomes more resilient against external attacks aimed at seizing ownership or manipulating the contract's behavior.

Remediation

It is recommended to use either Ownable2Step or Ownable2StepUpgradeable depending on the smart contract.

Retest

-

Bug ID #9

Floating and Outdated Pragma

Vulnerability Type

Floating Pragma (SWC-103)

Severity

Low

Description

Locking the pragma helps ensure that the contracts do not accidentally get deployed using an older version of the Solidity compiler affected by vulnerabilities.

The contract allowed floating or unlocked pragma to be used.

This allows the contracts to be compiled with all the solidity compiler versions above the limit specified.

Affected Code

- All the contract files

Impacts

If the smart contract gets compiled and deployed with an older or too recent version of the solidity compiler, there's a chance that it may get compromised due to the bugs present in the older versions or unidentified exploits in the new versions.

Incompatibility issues may also arise if the contract code does not support features in other compiler versions, therefore, breaking the logic. The likelihood of exploitation is low.

Remediation

Keep the compiler versions consistent in all the smart contract files. Do not allow floating pragmas anywhere. It is suggested to use the 0.8.20 pragma version.

Since most of the code is developed for older versions, it is highly recommended to carefully consider this remediation because it may break the code or introduce other inconsistencies.

Reference: https://swcregistry.io/docs/SWC-103

Retest

-

Bug ID #10

Use of Multiple Pragma Versions

Vulnerability Type
Missing Best Practices

Severity
Low

Description
The contracts were found to be using multiple Solidity Compiler versions across different Solidity files. This is not a good coding practice because different versions of the compiler have different caveats, breaking changes and introducing vulnerabilities.

Affected Code
- All the contracts

Impacts
Having different pragma versions across multiple contracts increases the chances of introducing vulnerabilities since each solidity version has its own set of issues and coding practices. Some major version upgrades may also break the contract logic if not handled properly.

Remediation
Instead of using different versions of the Solidity compiler with different bugs and security checks, it is better to use one version across all contracts.

Retest:
-

Bug ID #11

Functions should be declared External

Vulnerability Type
Best Practices

Severity
Informational

Description
Public functions that are never called by a contract should be declared external in order to conserve gas.
The following functions were declared as public but were not called anywhere in the contract, making public visibility useless.

Affected Code

- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/Multicall2.sol#L30-L32
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/Multicall2.sol#L21-L29

Impacts
Smart Contracts are required to have effective Gas usage as they cost real money and each function should be monitored for the amount of gas it costs to make it gas efficient.
"**public**" functions cost more Gas than "**external**" functions.

Remediation
Use the "**external**" state visibility for functions that are never called from inside the contract.

Retest
-

Bug ID #12

Use of SafeMath

Vulnerability Type
Gas Optimization

Severity
Gas

Description:
SafeMath library is found to be used in the contract. This increases gas consumption more than traditional methods and validations if done manually.
Also, Solidity **0.8.0** and above includes checked arithmetic operations by default, and this renders SafeMath unnecessary.

Affected Code:

- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeERC20.sol#L8-L8
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakePair.sol#L13-L13
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L14-L14

Impacts:
This increases the gas usage of the contract.

Remediation:
We do not recommend using the SafeMath library for all arithmetic operations. It is good practice to use explicit checks where it is really needed and to avoid extra checks where overflow/underflow is impossible.
It is recommended to upgrade to the latest compiler because versions above 0.8.0+ automatically check for overflows and underflows.

Retest:
-

Bug ID #13

Array Length Caching

Vulnerability Type

Gas Optimization

Severity

Gas

Description

During each iteration of the loop, reading the length of the array uses more gas than is necessary. In the most favorable scenario, in which the length is read from a memory variable, storing the array length in the stack can save about 3 gas per iteration. In the least favorable scenario, in which external calls are made during each iteration, the amount of gas wasted can be significant.

Affected Code

- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/Multicall2.sol#L24-L28
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/Multicall2.sol#L59-L67
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L171-L178
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L251-L259
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L373-L391

Impacts

Reading the length of an array multiple times in a loop by calling .length costs more gas.

Remediation

Consider storing the array length of the variable before the loop and use the stored length instead of fetching it in each iteration.

Retest

-

Bug ID #14

Gas Optimization in Require Statements

Vulnerability Type
Gas Optimization

Severity
Gas

Description
The **require()** statement takes an input string to show errors if the validation fails.
The strings inside these functions that are longer than **32 bytes** require at least one additional MSTORE, along with additional overhead for computing memory offset, and other parameters. For this purpose, having strings lesser than 32 bytes saves a significant amount of gas.

Vulnerable Code

- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakePair.sol#L126-L126
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakePair.sol#L147-L147
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakePair.sol#L161-L161
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakePair.sol#L179-L179
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeZapV1.sol#L712-L712
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeZapV1.sol#L713-L713
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L49-L49
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L54-L54
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L114-L114
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L115-L115
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L188-L188

- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L200-L200
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L213-L213
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L226-L226
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L240-L240
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L255-L255
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L52-L52
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L57-L57
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L141-L141
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L142-L142
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L270-L270
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L288-L288
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L306-L306
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L321-L321
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L342-L342
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L362-L362
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L409-L412
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L427-L430
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L449-L449

Impacts

Having longer require strings than 32 bytes costs a significant amount of gas.

Remediation

It is recommended to shorten the strings passed inside **require()** statements to fit under **32 bytes**. This will decrease the gas usage at the time of deployment and at runtime when the validation condition is met.

Retest

-

Bug ID #15

Gas Optimization in Increments

Vulnerability Type
Gas optimization

Severity
Gas

Description
The contract uses **for** loops that use post increments for the variable "**i**". The contract can save some gas by changing this to **++i**.
++i costs less gas compared to **i++** or **i += 1** for unsigned integers. In **i++**, the compiler has to create a temporary variable to store the initial value. This is not the case with **++i** in which the value is directly incremented and returned, thus, making it a cheaper alternative.

Vulnerable Code

- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/Multicall2.sol#L24-L24
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/Multicall2.sol#L59-L59
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter01.sol#L171-L171
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L251-L251
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeRouter.sol#L373-L373

Impacts
Using **i++** instead of **++i** costs the contract deployment around 600 more gas units.

Remediation
It is recommended to switch to **++i** and change the code accordingly so the function logic remains the same and saves some gas.

Retest
-

Bug ID #16

Variables should be Immutable

Vulnerability Type
Gas Optimization

Severity
Gas

Description:
Declaring state variables that are not updated following deployment as immutable can save gas costs in smart contract deployments and function executions. Immutable state variables are those that cannot be changed once they are initialized, and their values are set permanently.

By declaring state variables as immutable, the compiler can optimize their storage in a way that reduces gas costs. Specifically, the compiler can store the value directly in the bytecode of the contract, rather than in storage, which is a more expensive operation.

Affected Code:

- https://github.com/9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeERC20.sol#L17
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakePair.sol#L19-L19
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeZapV1.sol#L40-L40
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeZapV1.sol#L22-L22
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeZapV1.sol#L37-L37
- https://github.com//9mmPro/v2-contracts/blob/9c997a715b3ce0431d0b0653b8ec818d4265e906/PancakeZapV1.sol#L25-L25

Impacts:
Gas usage is increased if the variables that are not updated outside of the constructor are not set as immutable.

Remediation:

An “`immutable`” attribute should be added in the parameters that are never updated outside of the constructor to save the gas.

Retest

-