its__msn itsmsn

ITISH

AUDIT COMPANY

Audit Details

Contract Name
MissInternetToken

Deployer address

0x6234bdb78e84752eeb4a10b3794f4d32977eb925

Client contacts:

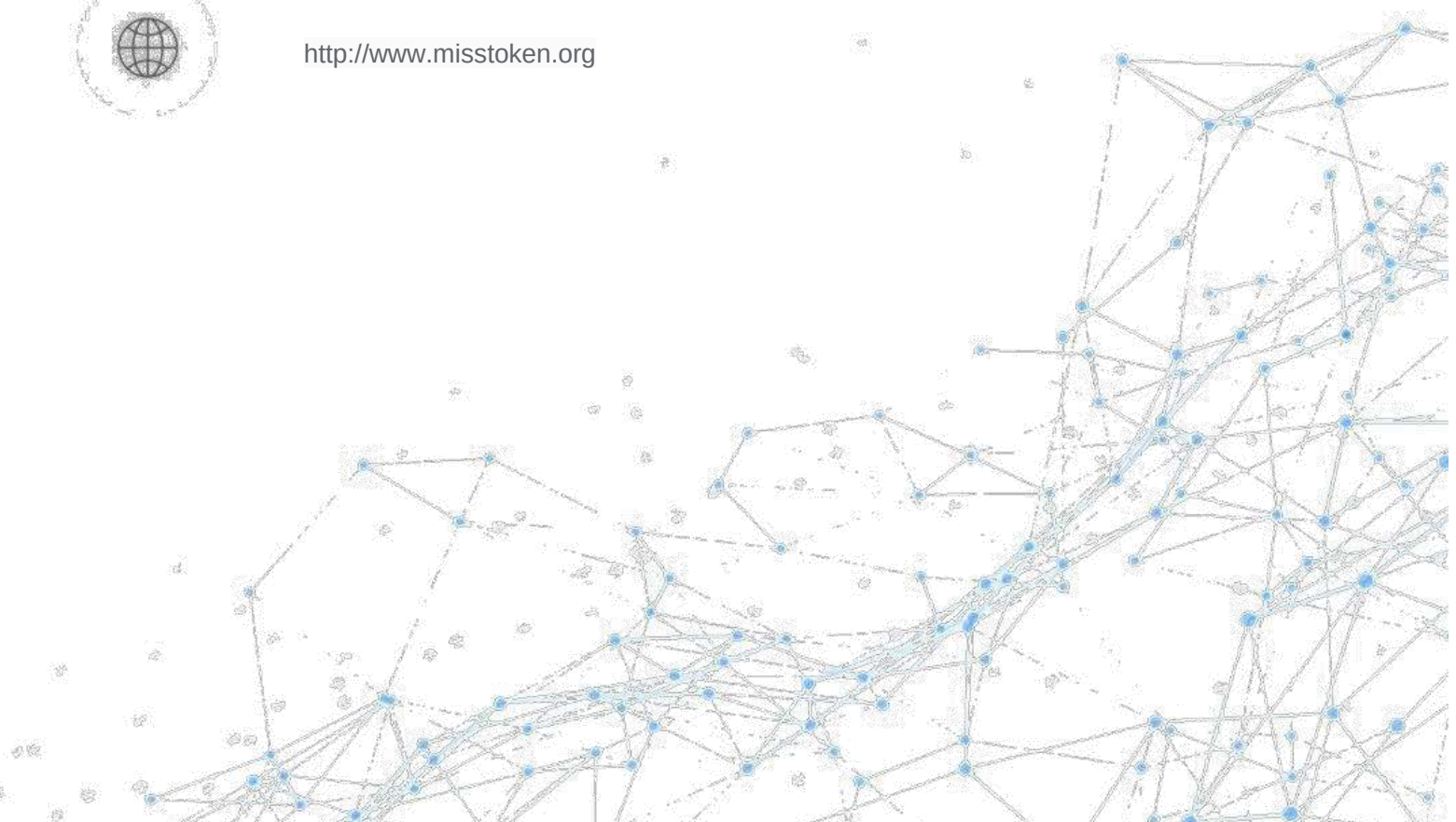
MissInternetToken team

Blockchain

Binance

Project website:

http://www.misstoken.org



Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Itish and its affiliates (includingholding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Itish) owe no duty of care towards you or any other person, nor does Itish make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Itish hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Itish hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Itish, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

Itish was commissioned Miss internet token to perform an audit of smart contracts:

https://bscscan.com/token/0x6234bdb78e84752eeb4a10b3794f4d32977eb925#code

The purpose of the audit was to achieve the following:

- **Ensure that the smart contract functions as intended.**
- **Identify potential security issues with the smart contract.**

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contract Details

Token contract details for 15.04.2023

contract name	**MissInternetToken**
Contract creator	0x6234bdb78e84752eeb4a10b3794f4d32977eb925
Transaction’s count	**48**

Contract TopTransactions

Txn Hash	Method ⓘ	Age	From		To	Quantity
0xe4b9eca632d4e23f80...	0x1e0f8b22	2 days 18 hrs ago	0x53b319057e75bd1de2...	→	0xf9069ba7d4a6c2aefd4...	129,819,404.385648301180464257
0xe4b9eca632d4e23f80...	0x1e0f8b22	2 days 18 hrs ago	0x6234bdb78e84752eeb...	→	Null: 0x000...000	529,875.119941421637471282
0xe4b9eca632d4e23f80...	0x1e0f8b22	2 days 18 hrs ago	0x53b319057e75bd1de2...	→	0x6234bdb78e84752eeb...	1,068,750.239882843274942565
0x370d0fe72171bb6259...	0x1e0f8b22	2 days 18 hrs ago	0x53b319057e75bd1de2...	→	0xf9069ba7d4a6c2aefd4...	10,816,283,698.804025098372021326
0x370d0fe72171bb6259...	0x1e0f8b22	2 days 18 hrs ago	0x6234bdb78e84752eeb...	→	Null: 0x000...000	44,166,259.995118469788273556
0x370d0fe72171bb6259...	0x1e0f8b22	2 days 18 hrs ago	0x53b319057e75bd1de2...	→	0x6234bdb78e84752eeb...	88,312,519.990236939578547112
0x7e544de0c620f6c0f5e...	Swap Exact Token...	3 days 2 hrs ago	0x7894d4d7c6bfdd5082...	→	0xf9069ba7d4a6c2aefd4...	36,798,840,457.859937990057261742
0x7e544de0c620f6c0f5e...	Swap Exact Token...	3 days 2 hrs ago	0x6234bdb78e84752eeb...	→	Null: 0x000...000	150,199,348.807591583632886782
0x7e544de0c620f6c0f5e...	Swap Exact Token...	3 days 2 hrs ago	0x7894d4d7c6bfdd5082...	→	0x6234bdb78e84752eeb...	300,398,697.615183167265773565
0x65c3eba48b2ec5da54...	0x5b03891d	3 days 3 hrs ago	0x171d953331ac5d1942...	→	0xf9069ba7d4a6c2aefd4...	8,354,186,332.914905417724269767
0x65c3eba48b2ec5da54...	0x5b03891d	3 days 3 hrs ago	0x6234bdb78e84752eeb...	→	Null: 0x000...000	34,068,719.726183287419282733
0x65c3eba48b2ec5da54...	0x5b03891d	3 days 3 hrs ago	0x171d953331ac5d1942...	→	0x6234bdb78e84752eeb...	68,197,439.452366574838565467
0x3a76b3278e5f86b570...	0xa14cf5df	3 days 5 hrs ago	0xfeaacb5385a0e29eb0...	→	0xf9069ba7d4a6c2aefd4...	11,474,865,218.844496591605718565
0x3a76b3278e5f86b570...	0xa14cf5df	3 days 5 hrs ago	0x6234bdb78e84752eeb...	→	Null: 0x000...000	46,836,184.56671223098614579

0x3a76b3278e5f86b570...	0xa14cf5df	3 days 5 hrs ago	0x6234bdb78e84752eeb...	→	Null: 0x000...000	46,836,184.56671223098614579
0x3a76b3278e5f86b570...	0xa14cf5df	3 days 5 hrs ago	0xfeaacb5385a0e29eb0...	→	0x6234bdb78e84752eeb...	93,672,369.13342446197229158
0x80dcfcdaff8ff868fcccd...	0x4ae2c0b9	3 days 7 hrs ago	0x74f1f47297d3ebd79f4...	→	0xf9069ba7d4a6c2aefd4...	16,645,517,505.436689559490773648
0x80dcfcdaff8ff868fcccd...	0x4ae2c0b9	3 days 7 hrs ago	0x6234bdb78e84752eeb...	→	Null: 0x000...000	67,940,887.777292610446901116
0x80dcfcdaff8ff868fcccd...	0x4ae2c0b9	3 days 7 hrs ago	0x74f1f47297d3ebd79f4...	→	0x6234bdb78e84752eeb...	135,881,775.554585220893802233
0x83dc3b3c0ee13e86f6f...	0x8a2abef	3 days 7 hrs ago	0x88683c774d2ba1d241...	→	0xf9069ba7d4a6c2aefd4...	14,686,950,974.6897160302887166
0x83dc3b3c0ee13e86f6f...	0x8a2abef	3 days 7 hrs ago	0x6234bdb78e84752eeb...	→	Null: 0x000...000	59,946,738.672202922572607006
0x83dc3b3c0ee13e86f6f...	0x8a2abef	3 days 7 hrs ago	0x88683c774d2ba1d241...	→	0x6234bdb78e84752eeb...	119,893,477.34440584514521 4012
0xc0e4956997f7f49b507...	0x14cd0615	3 days 7 hrs ago	0x8a36d0d88c03b6f20e...	→	0xf9069ba7d4a6c2aefd4...	16,436,998,305.288020351352183281
0xc0e4956997f7f49b507...	0x14cd0615	3 days 7 hrs ago	0x6234bdb78e84752eeb...	→	Null: 0x000...000	67,089,789.001175593270825237
0xc0e4956997f7f49b507...	0x14cd0615	3 days 7 hrs ago	0x8a36d0d88c03b6f20e...	→	0x6234bdb78e84752eeb...	134,179,578.002351186541650475
0x5b27ca811dfb69a7117...	0xc0cb8445	3 days 7 hrs ago	0x585e206207e8c48293...	→	0xf9069ba7d4a6c2aefd4...	15,525,036,521.667082342829831053

First < Page 1 of 2 > Last

Token Functions Details

```
Owner()
Check_owner()
renounceOwnership()
transferOwnership()
nonReentrantBefore()
nonReentrantafter()
name()
symbol()
decimal()
total supply()
balanceof()
transfer()
allowance()
approve()
transferfrom()
increase_allownace()
decrease_allownace()
burn()
mint()
spendallownace()
aftertokentransfer()
burnfrom()
```

Contract Interface Details

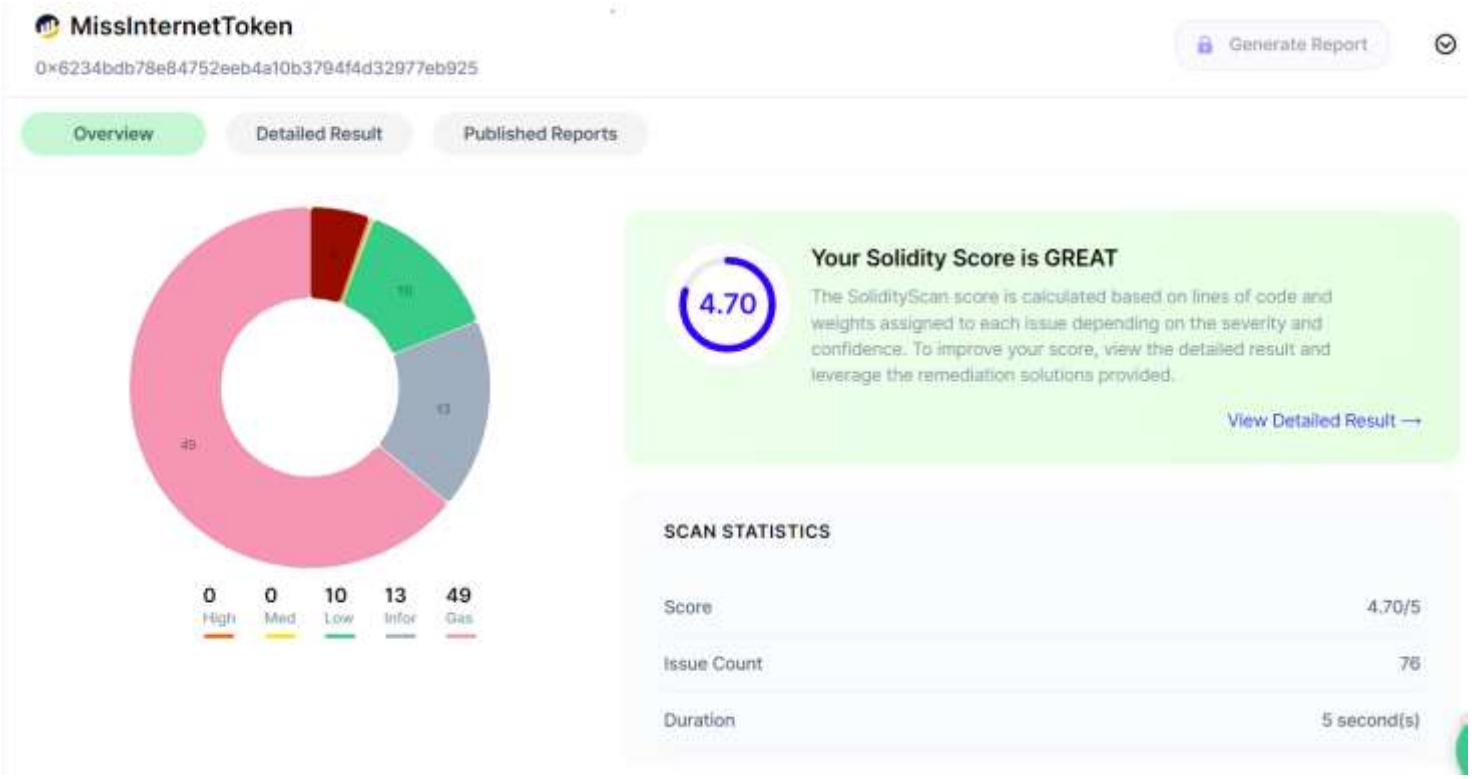
```
interface IERC20
interface IERC20Metadata is IERC20
```

Issues Checking Status

Issue description	checking status
1. Compiler errors.	Passed
2. Compiler Compatibilities	passed
3. Possible delays in data delivery.	Passed
4. Oracle calls.	Moderate
5. Front running.	Passed
6. Timestamp dependence.	Passed
7. Integer Overflow and Underflow.	Passed
8. DoS with Revert.	Severe
9. DoS with block gas limit.	Moderate
10. Methods execution permissions.	Passed
11. Economy model of the contract.	Passed
12. The impact of the exchange rate on the logic.	Severe
13. Private user data leaks.	Passed
14. Malicious Event log.	Passed
15. Scoping and Declarations.	Passed
16. Uninitialized storage pointers.	Passed
17. Arithmetic accuracy.	poor
18. Design Logic.	poor

19.	**Cross-function race conditions.**	**Passed**
20.	Safe Open Zeppelin contracts implementation and usage.	pass
21.	**Fallback function security.**	**Passed**

Security Issues



Low Severity Issues

Issue # 1:

MISSING EVENTS

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract `Miss internet token` was found to be missing these events on the function `_transfer` which would make it difficult or impossible to track these transactions off-chain.

```
 * Requirements:
 *
 * - `to` cannot be the zero address.
 * - the caller must have a balance of at least `amount`.
 */
function transfer(address to, uint256 amount) public virtual override returns (bool) {
    address owner = _msgSender();
    _transfer(owner, to, amount);
    return true;
}

/**
 * @dev See {IERC20-allowance}.
 */
function allowance(address owner, address spender) public view virtual override returns (uint256) {
    return _allowances[owner][spender];
}

/**
 * @dev See {IERC20-approve}.
```

Remediation # 1:

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Issue # 2:

MISSING EVENTS

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract `MISS EVENT TOKEN` was found to be missing these events on the function `_mint` which would make it difficult or impossible to track these transactions off-chain.

```
56      * Emits a {Transfer} event with `from` set to the zero address.
57      *
58      * Requirements:
59      *
60      * - `account` cannot be the zero address.
61      */
62     function _mint(address account, uint256 amount) internal virtual {
63         require(account != address(0), "ERC20: mint to the zero address");
64
65         _beforeTokenTransfer(address(0), account, amount);
66
67         _totalSupply += amount;
68         unchecked {
69             // Overflow not possible: balance + amount is at most totalSupply + amount, which is checked above.
70             _balances[account] += amount;
71         }
72         emit Transfer(address(0), account, amount);
73
74         _afterTokenTransfer(address(0), account, amount);
```

Remediation # 1:

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

```
event TaxDistributed(uint256 amount);

function distributeTax() public nonReentrant {
    ...
    emit TaxDistributed(balance);
}
```

Issue # 3:

MISSING EVENTS

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract `Miss event token` was found to be missing these events on the function `_burn` which would make it difficult or impossible to track these transactions off-chain.

```
     * Requirements:
     *
     * - `account` cannot be the zero address.
     * - `account` must have at least `amount` tokens.
     */
    function _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero address");

        _beforeTokenTransfer(account, address(0), amount);

        uint256 accountBalance = _balances[account];
        require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
        unchecked {
            _balances[account] = accountBalance - amount;
            // Overflow not possible: amount <= accountBalance <= totalSupply.
            _totalSupply -= amount;
        }
```

Remediation # 1:

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

TYPE 2:

Issue # 1:

USE OF FLOATING PRAGMA

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.

Remediation # 1:

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.

TYPE 3:
Issue # 1:

Compiler version to recent:

The use of a very recent compiler version may cause compatibility issues with previously written code, resulting in unexpected behaviors or even security vulnerabilities. New compiler versions may have new features or default settings that are different from previous versions, which could lead to unexpected behaviors. Furthermore, newly added security checks in the latest compiler version may not exist in the previous versions, which could create vulnerabilities in the older code.

Remediation # 1:

To avoid this vulnerability, it is recommended to use a stable and well-tested compiler version that is compatible with the code being used. Updating to the latest compiler version should be done gradually and with careful consideration of the code's compatibility with the new version. It is also important to thoroughly test the updated code before deploying it in a live environment.

```
pragma solidity ^0.8.0;
```

Informative

1)

PRESENCE OF OVERPOWERED ROLE

The overpowered owner (i.e., the person who has too much power) is a project design where the contract is tightly coupled to their owner (or owners); only they can manually invoke critical functions.
Due to the fact that this function is only accessible from a single address, the system is heavily dependent on the address of the owner. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g., if the private key of this address is compromised, then an attacker can take control of the contract.

Remediation # 1:

We recommend designing contracts in a trust-less manner. For instance, this functionality can be implemented in the contract's constructor. Another option is to use a MultiSig wallet for this address. For systems that are provisioned for a single user.

BOOLEAN EQUALITY

In Solidity, and many other languages, boolean constants can be used directly in conditionals like `if` and `else` statements.
The contract was found to be equating constants in conditionals which is unnecessary.

Remediation # 2:

It is recommended to directly use boolean constants. It is not required to equate them to true or false.

BLOCK VALUES AS A PROXY FOR TIME

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

Remediation # 3:

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

IN-LINE ASSEMBLY DETECTED
Inline assembly is a way to access the Ethereum Virtual Machine at a low level. This bypasses several important safety features and checks of Solidity. This should only be used for tasks that need it and if there is confidence in using it.

Remediation # 3:

Avoid using inline assembly instructions if possible because it might introduce certain issues in the code if not dealt with properly because it bypasses several safety features that are already implemented in Solidity.

HARD-CODED ADDRESS DETECTED

The contract contains an unknown hard-coded address. This address might be used for some malicious activity. Please check the hard-coded address and its usage.
These hard-coded addresses may be used everywhere throughout the code to define states and interact with the functions and external calls.
Therefore, it is extremely crucial to ensure the correctness of these token contracts as they define various important aspects of the protocol operation.

A misconfigured address mapping could lead to the potential loss of user funds or compromise of the contract owner depending on the function logic.

Remediation # 2:

It is required to check the address. Also, it is required to check the code of the called contract for vulnerabilities. Ensure that the contract validates if there's an address or a code change or test cases to validate if the address is correct.

GAS

1)

USE OF SAFEMATH LIBRARY:

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

Remediation # 1:

We do not recommend using 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.

The compiler should be upgraded to Solidity version 0.8.0+ which automatically checks for overflows and underflows.

CHEAPER INEQUALITIES IN REQUIRE()

The contract was found to be performing comparisons using inequalities inside the `require` statement. When inside the `require` statements, non-strict inequalities `(>=, <=)` are usually costlier than strict equalities `(>, <)`.

Remediation # 1:

It is recommended to go through the code logic, and, if possible, modify the non-strict inequalities with the strict ones to save `~3` gas as long as the logic of the code is not affected.

2)

FUNCTION SHOULD BE EXTERNAL

A function with `public` visibility modifier was detected that is not called internally.
`public` and `external` differs in terms of gas usage. The former use more than the latter when used with large arrays of data. This is due to the fact that Solidity copies arguments to memory on a `public` function while `external` read from calldata which a cheaper than memory allocation.

Remediation # 1:

If you know the function you create only allows for `external` calls, use the `external` visibility modifier instead of `public`. It provides performance benefits and you will save on gas.

3)

LONG REQUIRE/REVERT STRINGS

The `require()` and `revert()` functions take an input string to show errors if the validation fails.
This 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.

Remediation # 1:

It is recommended to short the strings passed inside `require()` and `revert()` 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.

Conclusion

Smart contracts don’t contain High severity issues! Liquidity pair contract’s security is not checked due to out of scope.

Liquidity locking details NOT provided by the team.

Itish note:

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.