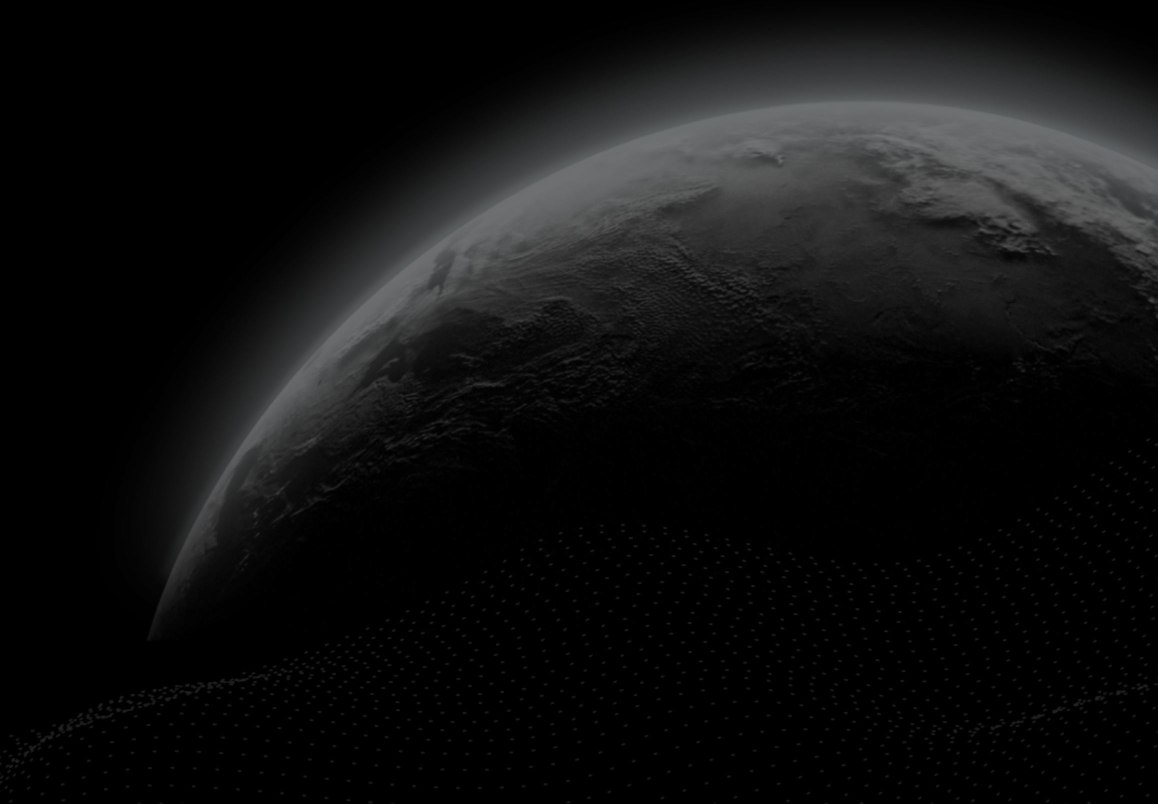
CERTIK

Security Assessment

mashida Token

CertiK Verified on Jan 12th, 2023



CertiK Verified on Jan 12th, 2023

mashida Token

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	BSC	Manual Review, Static Analysis

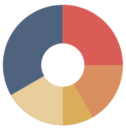
LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 01/12/2023	N/A

CODEBASE

https://github.com/mashidatoken/Token-Contract/blob/973ecdddbc5736f8e225269cf50877de3131a119/MashidaNoTax\ https://github.com/mashidatoken/Token-

...View All

Vulnerability Summary



12	9	0	0	3	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥 3	Critical	3 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.
🟧 2	Major	2 Acknowledged	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.
🟨 1	Medium	1 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨 2	Minor	1 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.
🟦 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 | MASHIDA TOKEN

Summary

- Executive Summary
- Vulnerability Summary
- Codebase
- Audit Scope
- Approach & Methods

Findings

- MNE-01 : "_allowances" calculation formula error
- MNR-01 : User balance can be increased after calling of `_transferToken` function
- MNT-01 : User balance can be increased after use of `transfer` function
- MNE-02 : Centralization Risks in MashidaNoTax_REV02
- MNT-02 : Centralization Risks in MashidaNoTax
- MNT-03 : Confusing Logic
- MNE-03 : Out of scope Dependency
- MNT-04 : Third Party Dependency
- MNT-05 : Redundant Code Components
- MNT-06 : Missing Emit Events
- MNT-07 : Usage of Magic Number
- MNT-08 : Redundant code in function `takeRes`

Optimizations

- MNT-09 : Variables That Could Be Declared as Immutable
- MNT-10 : Unnecessary Use of SafeMath
- MNT-11 : State Variable Should Be Declared Constant
- MNT-12 : Unused State Variable

Appendix

Disclaimer

CODEBASE | MASHIDA TOKEN

Repository

https://github.com/mashidatoken/Token-Contract/blob/973ecdddbc5736f8e225269cf50877de3131a119/MashidaNoTax\

https://github.com/mashidatoken/Token-Contract/blob/3a371f428810101fb6f796866b5ee4a3e5d7eb9a/MashidaNoTax_REV01\

https://github.com/mashidatoken/Token-Contract/commit/94bd7cc891ac7e5ca9f87beca0e8b6250bc8a20e\

https://github.com/mashidatoken/Token-Contract/blob/b7b94334346fb0007a177e5f185b94e94f6d790e/MashidaNoTax

AUDIT SCOPE | MASHIDA TOKEN

3 files audited ● 2 files with Acknowledged findings ● 1 file with Resolved findings

ID	File	SHA256 Checksum
● MNT	MashidaNoTax	422e03d462f4260961d560151f4c480352bd30678fec5234a83381e6cda911ca
● MNE	MashidaNoTax_REV02	ad3533fe6a416dbf6c13134bd44d148d1fe7aa64db9a015a0a8d99953d031103
● MNR	MashidaNoTax_REV01	2c94e2ce1a4f3ef9642d51e74c2914ac1a0461ac315ecb0efb03f2e3df2651e0

APPROACH & METHODS | MASHIDA TOKEN

This report has been prepared for mashida Token to discover issues and vulnerabilities in the source code of the mashida Token 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.

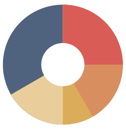
The auditing process pays special attention to the following considerations:

- 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.

FINDINGS | MASHIDA TOKEN



Total Findings	Critical	Major	Medium	Minor	Informational
12	3	2	1	2	4

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

ID	Title	Category	Severity	Status
MNE-01	"_allowances" Calculation Formula Error	Logical Issue	Critical	● Resolved
MNR-01	User Balance Can Be Increased After Calling Of `_transferToken` Function	Logical Issue	Critical	● Resolved
MNT-01	User Balance Can Be Increased After Use Of `transfer` Function	Logical Issue	Critical	● Resolved
MNE-02	**Centralization Risks In MashidaNoTax_REV02**	**Centralization / Privilege**	**Major**	● **Acknowledged**
MNT-02	**Centralization Risks In MashidaNoTax**	**Centralization / Privilege**	**Major**	● **Acknowledged**
MNT-03	Confusing Logic	Logical Issue	Medium	● Resolved
MNE-03	Out Of Scope Dependency	Volatile Code	Minor	● Acknowledged
MNT-04	Third Party Dependency	Volatile Code	Minor	● Resolved
MNT-05	Redundant Code Components	Volatile Code	Informational	● Resolved
MNT-06	Missing Emit Events	Coding Style	Informational	● Resolved

ID	Title	Category	Severity	Status
MNT-07	Usage Of Magic Number	Magic Numbers	Informational	● Resolved
MNT-08	Redundant Code In Function `takeRes`	Mathematical Operations	Informational	● Resolved

MNE-01 | "_ALLOWANCES" CALCULATION FORMULA ERROR

Category	Severity	Location	Status
Logical Issue	● Critical	MashidaNoTax_REV02 (v3): 151	● Resolved

Description

According to the function's meaning of `transferFrom` , `_allowances [sender] [_msgSender()]` should be reduced rather than increased.

Recommendation

The allowance should be decreased instead of increased.

MNR-01 | USER BALANCE CAN BE INCREASED AFTER CALLING OF `_transferToken` FUNCTION

Category	Severity	Location	Status
Logical Issue	● Critical	MashidaNoTax_REV01 (v2)	● Resolved

Description

In the code of the fixed version(https://github.com/mashidatoken/Token-Contract/commit/3a371f428810101fb6f796866b5ee4a3e5d7eb9a), the "_transferToken" function has 2 logic issues.

```
190  function _transferToken(address sender, address recipient, uint256 amount)
internal returns (bool) {
191
192
193
194      _transfer(sender, recipient, amount);
195
196
197
198      uint256 amountReceived = amount;
199
200          _transfer(sender, recipient, amount);
201
202
203      _balances[recipient] = _balances[recipient].add(amountReceived);
204      emit Transfer(sender, recipient, amount);
205      return true;
206    }
```

1. The "_transferToken" function calls the "_transfer" function twice, which will cause the `sender` to transfer the token to the `recipient` twice.
2. According to 203 lines of code, the token of the `amountReceived` amount will be added to the `recipient`, so the recipient can obtain more tokens of the `amountReceived` amount after calling this function.

Scenario

Scenario-1

1. User1 has `10` Mashida tokens.
2. User1 calls the `transfer` function with his/her own address and `5` tokens.

3. User1's new balance will become `15` tokens.
4. Repeat steps 2 until profit. Scenario-2
5. User1 has `10` Mashida tokens.
6. User1 calls the `transfer` function with User2's address and `5` tokens.
7. User1's new balance will become `0` tokens and User2's balance will increase "15" tokens.

Recommendation

According to the code logic, the "transfer" and "transferFrom" functions can directly call the "_transfer" function instead of calling the "_transferToken" function. After replacement, "_transferToken" can be deleted.

Alleviation

Fixed in commit b7b94334346fb0007a177e5f185b94e94f6d790e

MNT-01 | USER BALANCE CAN BE INCREASED AFTER USE OF `transfer` FUNCTION

Category	Severity	Location	Status
Logical Issue	● Critical	MashidaNoTax (v1): 322, 374~378	● Resolved

Description

The `_transfer` function is responsible for transferring tokens between addresses. The contract takes some residue in every transfer depending on the transfer amount. Some of the addresses are excluded from this, such as `owner, contract address, and DEAD` in the constructor as below:

```
    _excludedResidue[owner()] = true;
    _excludedResidue[address(this)] = true;
    _excludedResidue[DEAD] = true;
```

The `_transfer` function checks whether the user is in an excluded address as below:

```
if (!_excludedResidue[sender]){
         unchecked {
       _balances[sender] = senderBalance - amount + takeRes(amount);
     }
        }
else {
        unchecked {
       _balances[sender] = senderBalance - amount;
        }
      }
```

Because none of the users are in the `_excludedResidue,` the code flow will follow the if condition. In this situation contract will call the `takeRes` function, which is like the below:

```

 function takeRes (uint256 amount_) private returns(uint256){
    uint256 res = _residue.mul(amount_).div(_leaves);
    _balances[address(this)] = _balances[address(this)].add(res);
    return amount_.sub(amount_.sub(res));
  }
```

If the amount is higher than the `_leaves` values, this function will increase the contract token balance and then return a value bigger than 1.

The below line will increase user balance according to the return value from the `takeRes` function.

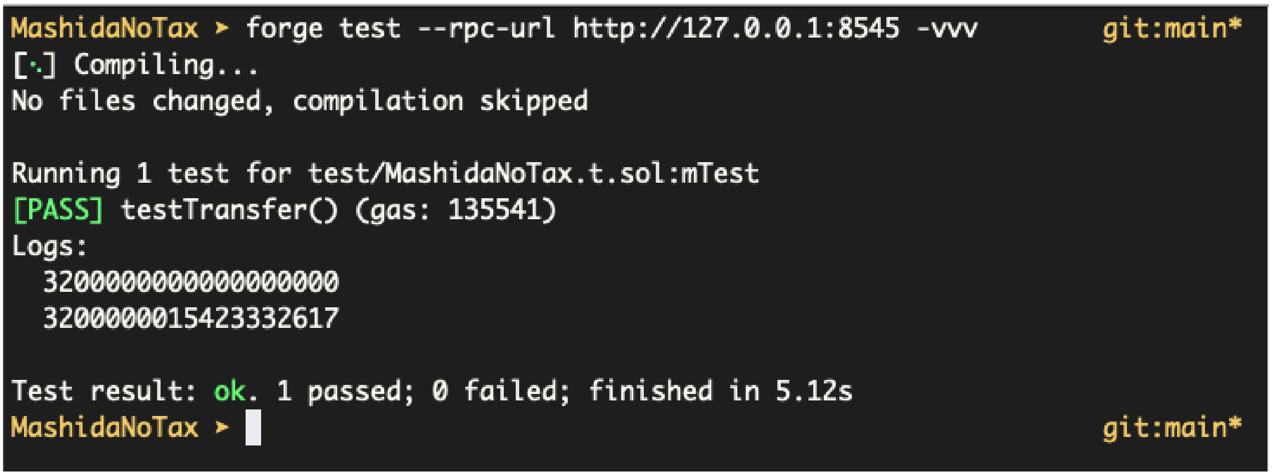
```
_balances[sender] = senderBalance - amount + takeRes(amount);
```

Scenario

1. User has `14523450000` Mashida tokens.
2. User calls the `transfer` function with his/her own address and `14523450000` tokens.
3. Because the user is not in the excluded residue, the contract will call the `takeRes` function.
4. In the `takeRes` function, the `amount_` is 10 times bigger than the `_leaves` value which will lead to increasing the contract's token balance by 10 and returns 10 as the `res` value.
5. User's new balance will become `14523450010`
6. Repeat steps 2-4 until profit.

Proof of Concept

```
pragma solidity ^0.8.6;

import "forge-std/Test.sol";
import "../src/MashidaNoTax.sol";


contract mTest is Test {
    Mashida public m;
    //Private key of address 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
    //Update LIQUIDITY_PROVISION address with this
    address A =
vm.addr(7781451732547020591114094119440192857955706201476183193064539304138081900940
8);

    function setUp() public {
        m = new Mashida();
    }

    function testTransfer() public{
        console.log(m.balanceOf(A));

        vm.startPrank(A);
        m.transfer(A, 3200000000000000000);
        m.transfer(A, 3200000000000000000);
        m.transfer(A, 3200000000000000000);
        m.transfer(A, 3200000000000000000);
        m.transfer(A, 3200000000000000000);
        m.transfer(A, 3200000000000000000);
        m.transfer(A, 3200000000000000000);
        vm.stopPrank();
        console.log(m.balanceOf(A));

    }
}
```



```
MashidaNoTax ➤ forge test --rpc-url http://127.0.0.1:8545 -vvv          git:main*
[⠊] Compiling...
No files changed, compilation skipped

Running 1 test for test/MashidaNoTax.t.sol:mTest
[PASS] testTransfer() (gas: 135541)
Logs:
  32000000000000000000
  32000000015423332617

Test result: ok. 1 passed; 0 failed; finished in 5.12s
MashidaNoTax ➤                                                          git:main*
```

Recommendation

It's unclear to us what the purpose of the 'takeRes' feature. The auditor recommends either removing it completely or properly modifying the logic so that it cannot be taken advantage of.

Alleviation

The client changed the logic and removed the code.

MNE-02 | CENTRALIZATION RISKS IN MASHIDANOTAX_REV02

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**MashidaNoTax_REV02 (v3): 137**	● **Acknowledged**

Description

In the contract `Mashida` 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

- Set `antiBotEnabled` value

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

The client acknowledge this finding.

MNT-02 | CENTRALIZATION RISKS IN MASHIDANOTAX

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**MashidaNoTax (v1): 47, 51, 211, 243**	● **Acknowledged**

Description

In the contract `Mashida` 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

- Renounce Ownership
- Transfer Ownership

In the contract `Ownable` 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

- Set Residue value
- Change ExcludeResidue value

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

The client acknowledge this finding.

MNT-03 | CONFUSING LOGIC

Category	Severity	Location	Status
Logical Issue	● Medium	MashidaNoTax (v1): 344~345	● Resolved

Description

The linked code transfers the amount of "amountRes" from the "sender" to "msg.sender". It's unclear what's the purpose of such behavior.

```
if(!_excludedResidue[sender])
_transfer(sender, _msgSender(), amountRes);
```

Recommendation

It is recommended that the team checks whether the code matches the intended design. If not, remove the linked code from the contract.

Alleviation

The client changed the logic and removed the code.

MNE-03 | OUT OF SCOPE DEPENDENCY

Category	Severity	Location	Status
Volatile Code	● Minor	MashidaNoTax_REV02 (v3): 7, 89, 117, 118, 183	● Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assume 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.

The security of this thirty-party code "PinkAntiBot" and whether the Mashida's contract correctly utilizes the "PinkAntiBot" are outside the scope of the audit.

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to fully understand the usage and security aspects of any imported third parties before using it. We encourage the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The client acknowledge this finding.

MNT-04 | THIRD PARTY DEPENDENCY

Category	Severity	Location	Status
Volatile Code	● Minor	MashidaNoTax (v1): 237	● Resolved

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assume 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.

```
237    IRouter public _router;
```

- The contract `Mashida` interacts with third party contract with `IRouter` interface via `_router`.

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The client changed the logic and removed the code.

MNT-05 | REDUNDANT CODE COMPONENTS

Category	Severity	Location	Status
Volatile Code	● Informational	MashidaNoTax (v1): 239	● Resolved

Description

The linked statements do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation

We advise to remove the redundant statements for production environments.

Alleviation

The client removed this code.

MNT-06 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	● Informational	MashidaNoTax (v1): 211, 243	● Resolved

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

The client changed the logic and removed the code.

MNT-07 | USAGE OF MAGIC NUMBER

Category	Severity	Location	Status
Magic Numbers	● Informational	MashidaNoTax (v1): 339	● Resolved

Description

The linked magic number is hardcoded in the codebase.

Recommendation

According to the meaning of the code, we recommend using `_leaves` to replace this value.

Alleviation

The client changed the logic and removed the code.

MNT-08 | REDUNDANT CODE IN FUNCTION `takeRes`

Category	Severity	Location	Status
Mathematical Operations	● Informational	MashidaNoTax (v1): 377	● Resolved

Description

The function `takeRes` is responsible for taking residue in each transfer. The return value of this function is hard to understand because it is a simplified version of the expression

```
amount_ - (amount_ - res)
```

Which is equal to `res`. This mathematical expression, while correct, is not immediately apparent in terms of what it represents or what the function is doing.

Recommendation

By changing the return statement as below, the return value can be clearer, making the code more readable and easier to understand.

```
return res;
```

Alleviation

The client changed the logic and removed the code.

OPTIMIZATIONS | MASHIDA TOKEN

ID	Title	Category	Severity	Status
MNT-09	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	● Resolved
MNT-10	Unnecessary Use Of SafeMath	Gas Optimization	Optimization	● Resolved
MNT-11	State Variable Should Be Declared Constant	Gas Optimization	Optimization	● Resolved
MNT-12	Unused State Variable	Gas Optimization	Optimization	● Resolved

MNT-09 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	● Optimization	MashidaNoTax (v1): 30, 236	● Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

MNT-10 | UNNECESSARY USE OF SAFEMATH

Category	Severity	Location	Status
Gas Optimization	● Optimization	MashidaNoTax (v1): 175, 202, 288, 352, 358, 368, 375, 376, 377	● Resolved

Description

The `SafeMath` library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations will automatically revert in case of integer overflow or underflow.

```
175  library SafeMath {
```

- An implementation of `SafeMath` library is found.

```
202    using SafeMath for uint256;
```

- `SafeMath` library is used for `uint256` type in `Mashida` contract.

```
288        _allowances[sender][_msgSender()] = _allowances[sender]
[_msgSender()].sub(amount, "Insufficient allowance.");
```

- `SafeMath.sub` is called in `transferFrom` function of `Mashida` contract.

Note: Only a sample of 2 `SafeMath` library usage in this contract (out of 26) are shown above.

Recommendation

We advise removing the usage of `SafeMath` library and using the built-in arithmetic operations provided by the Solidity programming language.

MNT-11 | STATE VARIABLE SHOULD BE DECLARED CONSTANT

Category	Severity	Location	Status
Gas Optimization	● Optimization	MashidaNoTax (v1): 208, 209	● Resolved

Description

```
208    uint256 public _leaves = 1452345000;
```

- `_leaves` should be declared `constant` .

```
209    address public _ownAddress;
```

- `_ownAddress` should be declared `constant` .

Recommendation

We recommend adding the `constant` attribute to state variables that never change.

Alleviation

The client changed the logic and removed the code.

MNT-12 | UNUSED STATE VARIABLE

Category	Severity	Location	Status
Gas Optimization	● Optimization	MashidaNoTax (v1): 218, 226	● Resolved

Description

One or more state variables are never used in the codebase.

Variable `ZERO` in `Mashida` is never used in `MashidaNoTax` .

```
218    address constant ZERO = address(0);
```

```
195  contract Mashida is
```

Variable `FACTORY` in `Mashida` is never used in `MashidaNoTax` .

```
226    address constant FACTORY = 0xB7926C0430Afb07AA7DEfDE6DA862aE0Bde767bc;
//pancakefactory testnet
```

```
195  contract Mashida is
```

Recommendation

We advise removing the unused variables.

Alleviation

The client removed the code.

APPENDIX | MASHIDA TOKEN

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.
Mathematical Operations	Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.
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.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.
Magic Numbers	Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

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.