



Security Assessment

Mrmint - Audit

CertiK Verified on Apr 25th, 2023





CertiK Verified on Apr 25th, 2023

Mrmint - Audit

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

Executive Summary

TYPES

DeFi

ECOSYSTEM

Binance Smart Chain (BSC)

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 04/25/2023

KEY COMPONENTS

N/A

CODEBASE

- <https://bscscan.com/address/0x09ad1287d3bcf930baff65edf4e0460edd512b9d>
- <https://bscscan.com/address/0x3e81Aa8d6813Ec9D7E6ddB4>

[...View All](#)

COMMITTS

- TeamVestingWallet: 0x09AD1287D3BcF930bAff65Edf4E0460EdD512b9d
- MrMint: 0x3e81Aa8d6813Ec9D7E6ddB4e523fb1601a0e86F3

[...View All](#)

Vulnerability Summary



16

Total Findings

15

Resolved

0

Mitigated

0

Partially Resolved

1

Acknowledged

0

Declined

0

Unresolved

0

Critical

Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.

1

Major

1 Resolved



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.

3

Minor

3 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.

12

Informational

11 Resolved, 1 Acknowledged



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 | MRMINT - AUDIT

I **Summary**

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

I **Review Notes**

[External Dependencies](#)

[Addresses](#)

I **Decentralization Efforts**

[Privileged Roles](#)

[Owner in MrMint](#)

[Owner in MrMintTokenVesting](#)

[Recommendations](#)

[Short Term:](#)

[Long Term:](#)

[Permanent:](#)

[Alleviation](#)

I **Mainnet**

I **Findings**

[0X7-01 : Potential Denial of Service on Revoking Vesting By Multisig](#)

[0X0-02 : Missing check on `start` and `duration`](#)

[0X0-03 : Potential Locked Ether](#)

[0X8-01 : Usage of `transfer` for sending Ether](#)

[0X0-01 : Redundant Code Component](#)

[0X6-01 : Failed transfers will emit event](#)

[0X7-02 : Discussion on `computeReleasableAmount`](#)

[MMC-01 : Missing Zero Address Validation](#)

[MMC-03 : Unlocked Compiler Version](#)

[MMT-05 : Discussion on `token`](#)

[MMT-06 : Discussion on preconditions needed to create vestings](#)

[MMT-08 : Missing Error Messages](#)

MMT-11 : Function will revert if holder address has no previous vesting

MMT-14 : Unnecessary `payable` Address Type

TES-01 : Missing Emit Events

TES-02 : Unused Events

| Optimizations

0X0-04 : Unnecessary Use of SafeMath

0X8-02 : State Variable Should Be Declared Constant

| Appendix

| Disclaimer

CODEBASE | MRMINT - AUDIT

Repository



- <https://bscscan.com/address/0x09ad1287d3bcf930baff65edf4e0460edd512b9d>
- <https://bscscan.com/address/0x3e81Aa8d6813Ec9D7E6ddB4e523fb1601a0e86F3>

Commit

- TeamVestingWallet: 0x09AD1287D3BcF930bAff65Edf4E0460EdD512b9d
- MrMint: 0x3e81Aa8d6813Ec9D7E6ddB4e523fb1601a0e86F3

AUDIT SCOPE | MRMINT - AUDIT

2 files audited ● 2 files without findings

ID	Repo	File	SHA256 Checksum
● TVW	mainnet	 TeamVestingWallet.sol	7472b88085f396ca2ca293b1c6bb30b9878ce 496067d87f8eeb260a6f972b96a
● MMS	mainnet	 MrMint.sol	f5bdb7181ba8c871aa408bba58d6de11dd2ff7 b84d8a069269de2f6c7b32dea8

APPROACH & METHODS | MRMINT - AUDIT

This report has been prepared for Mrmint to discover issues and vulnerabilities in the source code of the Mrmint - Audit 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.

REVIEW NOTES | MRMINT - AUDIT

External Dependencies

Addresses

The following are external addresses used within the contracts:

- `_token` - token address that implements IERC20 interface and calls `safeTransfer()` and `balanceOf()` functions)

It is assumed that these contracts and addresses are valid and are implemented properly within the current project.

Notice: In the auditing version, all hardcoded addresses are reflected on the BSC testnet. During the Mainnet deployment, the team should verify the appropriate addresses to hardcode.

DECENTRALIZATION EFFORTS | MRMINT - AUDIT

In the `MrMint` project, multiple privileged roles are adopted to ensure the dynamic runtime updates and management of the project.

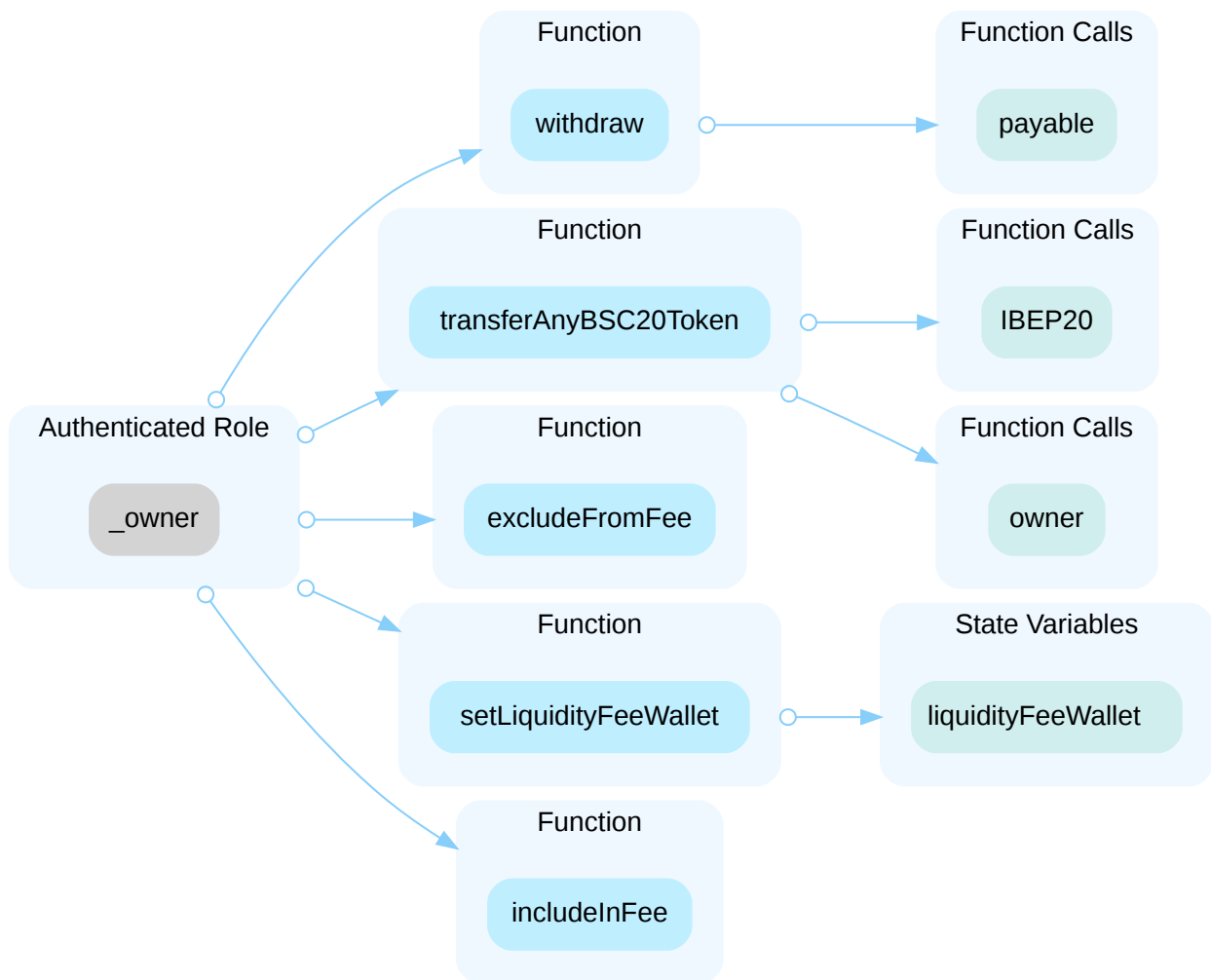
Privileged Roles

Owner in MrMint

In the contract `MrMint` the role `_owner` has authority over the functions shown in the diagram and list below.

- `withdraw()`
- `transferAnyBSC20Token()`
- `excludeFromFee()`
- `setLiquidityFeeWallet()`
- `includeInFee()`
- `renounceOwnership()` - inherited from Ownable contract
- `transferOwnership()` - inherited from Ownable contract

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and steal any BNB or BEP20 tokens, include or excluded addresses from paying fees during transfers, change the `liquidityFeeWallet`.

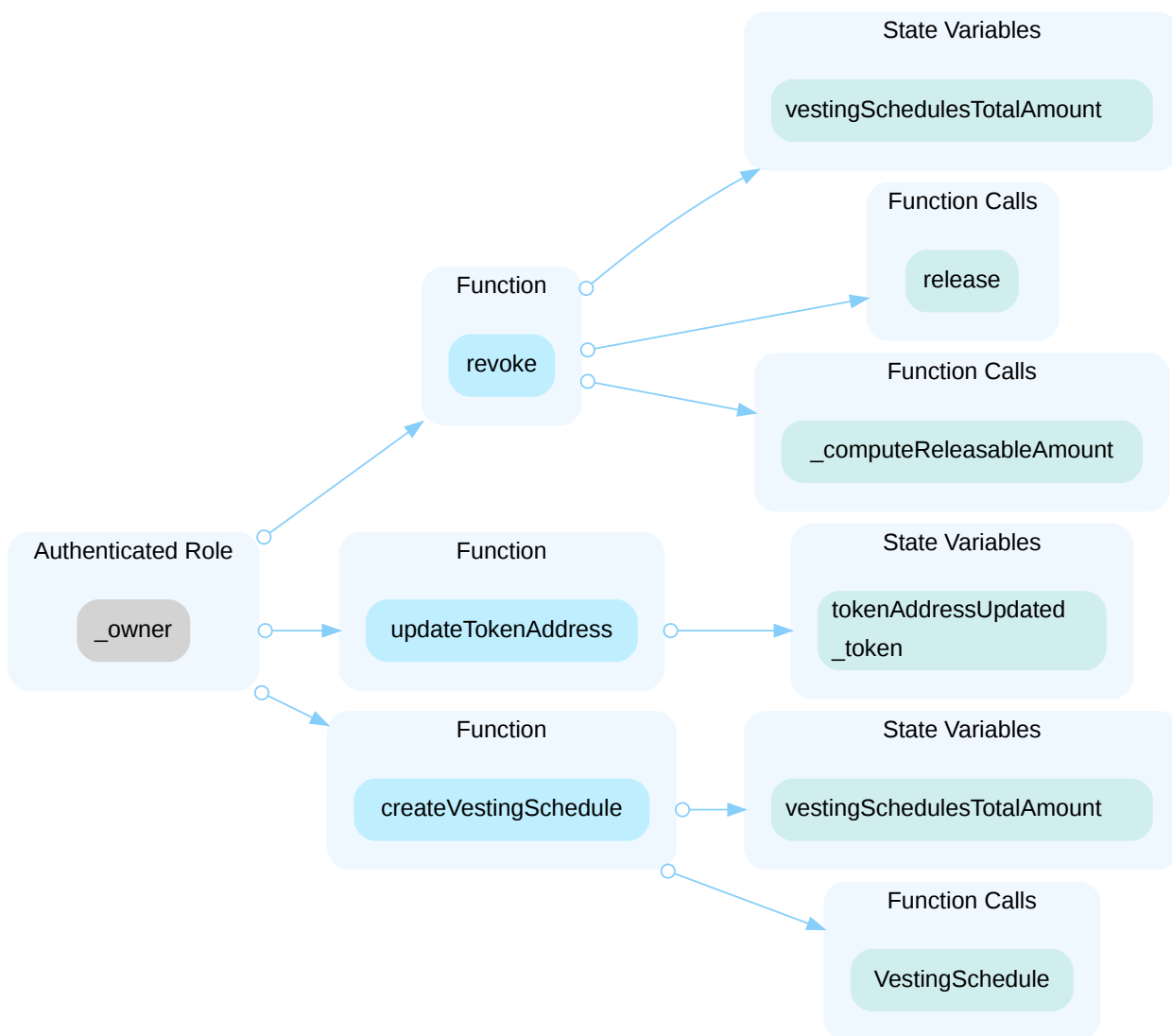


Owner in MrMintTokenVesting

In the contract `MrMintTokenVesting` the role `_owner` has authority over the functions shown in the diagram and list below.

- `revoke()`
- `release()`
- `updateTokenAddress()`
- `createVestingSchedule()`
- `renounceOwnership()` - inherited from `Ownable` contract
- `transferOwnership()` - inherited from `Ownable` contract

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and create, revoke or release vestings and update the `_token` state variable.



Recommendations

The advantage of privileged roles in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worth noting the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private keys of the privileged accounts are compromised, it could lead to devastating consequences for the project.

The risk describes the current project design and potentially makes iterations to improve the security operations and level of decentralization, which in most cases cannot be resolved entirely at the present stage. The client is advised to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, it is strongly recommended that centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature 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:

The Timelock and Multi sign (2/3, 3/5) combination *mitigates* this risk 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 being compromised;
AND
- A medium/blog link for sharing the timelock contract and multi-signature addresses with the public audience.

Long Term:

The combination of Timelock and a DAO *mitigates* this risk 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-signature addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renouncing the ownership and never claiming back the privileged roles.
OR
- Remove the risky functionality.

Note: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team provided the following multi-sig address on BSC mainnet [0x70488948648d8d1f8CD3b91d76897DdD27384a82](#) on BSC mainnet, which includes three owners with 3/3 confirmation rule(out-of-scope):

- [0xcF931E5656D3d40eb64Be94d182E63612B3ef10c](#)
- [0xe53a8Cc515205fBF03D657Ed582e9C8ddD8b6d3e](#)
- [0xe7f344E26CFaa51737D3A9540aC508f903a75031](#)

MAINNET | MRMINT - AUDIT

MrMint project was deployed on BSC mainnet on March 30th 2023:

- MrMint : 0x3e81Aa8d6813Ec9D7E6ddB4e523fb1601a0e86F3
- TeamVestingwallet(MrMintTokenVesting) : 0x09AD1287D3BcF930bAff65Edf4E0460EdD512b9d
- Multi-sig (out-of-scope): 0x70488948648d8d1f8CD3b91d76897DdD27384a82

FINDINGS | MRMINT - AUDIT



16

Total Findings

0

Critical

1

Major

0

Medium

3

Minor

12

Informational

This report has been prepared to discover issues and vulnerabilities for Mrmint - Audit. Through this audit, we have uncovered 16 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
0X7-01	Potential Denial Of Service On Revoking Vesting By Multisig	Logical Issue	Major	● Resolved
0X0-02	Missing Check On <code>_start</code> And <code>_duration</code>	Logical Issue	Minor	● Resolved
0X0-03	Potential Locked Ether	Language Specific	Minor	● Resolved
0X8-01	Usage Of <code>transfer</code> For Sending Ether	Volatile Code	Minor	● Resolved
0X0-01	Redundant Code Component	Volatile Code	Informational	● Resolved
0X6-01	Failed Transfers Will Emit Event	Logical Issue	Informational	● Resolved
0X7-02	Discussion On <code>_computeReleasableAmount</code>	Logical Issue	Informational	● Acknowledged
MMC-01	Missing Zero Address Validation	Volatile Code	Informational	● Resolved
MMC-03	Unlocked Compiler Version	Language Specific	Informational	● Resolved
MMT-05	Discussion On <code>_token</code>	Logical Issue	Informational	● Resolved
MMT-06	Discussion On Preconditions Needed To Create Vestings	Logical Issue	Informational	● Resolved

ID	Title	Category	Severity	Status
MMT-08	Missing Error Messages	Coding Style	Informational	● Resolved
MMT-11	Function Will Revert If Holder Address Has No Previous Vesting	Logical Issue	Informational	● Resolved
MMT-14	Unnecessary payable Address Type	Language Specific	Informational	● Resolved
TES-01	Missing Emit Events	Coding Style	Informational	● Resolved
TES-02	Unused Events	Coding Style	Informational	● Resolved

0X7-01 | POTENTIAL DENIAL OF SERVICE ON REVOKING VESTING BY MULTISIG

Category	Severity	Location	Status
Logical Issue	● Major	0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00 (MrMintTokenVesting v3): 859	● Resolved

Description

When the vesting schedule is revocable, the role multisig wallet may stop the vesting plan and send claimable tokens to users via `release`.

```
858     if(vestedAmount > 0){
859         release(vestingScheduleId, vestedAmount);
860     }
861
```

The concern is, the `release` function can only be invoked by the `beneficiary`, which means as long as `beneficiary` is not the multisig wallet and the claimable amount is larger than 0, the revoke function will not work.

```
884     require(
885         isBeneficiary,
886         "TokenVesting: only beneficiary can release vested tokens"
887     );
```

Scenario

We can assume the scenario that:

1. The multisig wallet creates a vesting plan for the user `address(1001)`, with 100s cliff, 10000s duration, also revocable.
2. After 101s, which means the cliff passed, the vested amount will be larger than 0. The multisig wallet wants to revoke the vesting plan, and it will be failed.

Proof of Concept

Here is PoC with Foundry framework and inside the test we create a mock ERC20 token for the test token:


```
1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ^0.8.13;
3
4 import "forge-std/Test.sol";
5 import "../src/mrmint.sol";
6
7
8 contract mrmintTest is Test {
9     MockERC20 erc20;
10    MrMintTokenVesting mmv;
11
12    function setUp() public {
13        erc20 = new MockERC20();
14        mmv = new MrMintTokenVesting(address(this));
15
16        erc20.mint(address(mmv), 10000000 * 10**18);
17        mmv.updateTokenAddress(address(erc20));
18
19    }
20
21    function testRevoke() public {
22
23        address beneficiary = address(1001);
24
25        mmv.createVestingSchedule(
26            beneficiary,
27            block.timestamp + 1,
28            100, // cliff
29            10000, // duration
30            1,
31            true,
32            10000 * 10 ** 18
33        );
34
35        vm.warp(block.timestamp + 1 + 100 + 1);
36
37        uint256 index = 0;
38        bytes32 id = keccak256(abi.encodePacked(beneficiary, index));
39
40        vm.expectRevert("TokenVesting: only beneficiary can release vested
tokens");
41        mmv.revoke(id);
42
43    }
44
45 }
46
```

Recommendation

Recommend reconsidering the accessibility of the `release()` function to avoid unexpected DoS.

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x6e1573Fd434Db2Ff49bef573836B854D7c64C88a](#) by updating the access control.

OX0-02 | MISSING CHECK ON `_start` AND `_duration`

Category	Severity	Location	Status
Logical Issue	● Minor	0x00476072b53902fb201905b49ae296f94b8fabf6 (MrMintTokenVesting): 868 , 870	● Resolved

Description

When creating a new vesting schedule in `createVestingSchedule()` it must not be possible to create a vesting that started in the past, thus the following check is necessary:

```
require(_start > getCurrentTime(), "TokenVesting: start date must be greater than the current time");
```

Furthermore it must not be possible to create a vesting if the input `_duration` is lower than or equal to the cliff duration. Thus the following check is necessary:

```
require(_duration > _cliff, "TokenVesting: vesting duration must be greater than the cliff duration");
```

Recommendation

We recommend adding the aforementioned checks in `createVestingSchedule()`

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00](#).

0X0-03 | POTENTIAL LOCKED ETHER

Category	Severity	Location	Status
Language Specific	● Minor	0x00476072b53902fb201905b49ae296f94b8fabf6 (MrMintTokenVesting): 788	● Resolved

Description

The contract `MrMintTokenVesting` includes a payable receive function, which permits the deposit of native tokens. However, the contract lacks a function to retrieve these funds, resulting in their permanent lock within the contract.

Recommendation

We recommend adding a function to rescue native token accidentally sent to the contract.

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00](#).

0X8-01 | USAGE OF `transfer` FOR SENDING ETHER

Category	Severity	Location	Status
Volatile Code	● Minor	0x8e846a318510138bdca8c524c4604e321372ca00 (MrMint): 747	● Resolved

Description

It is not recommended to use Solidity's `transfer()` function for transferring Ether, since some contracts may not be able to receive the funds. Those functions forward only a fixed amount of gas (2300 specifically) and the receiving contracts may run out of gas before finishing the transfer. Also, EVM instructions' gas costs may increase in the future. Thus, some contracts that can receive now may stop working in the future due to the gas limitation.

- `MrMint.withdraw()` uses `transfer()`.

```
747 payable(msg.sender).transfer(amount);
```

Recommendation

We recommend using the `Address.sendValue()` function from OpenZeppelin.

Since `Address.sendValue()` may allow reentrancy, we also recommend guarding against reentrancy attacks by utilizing the [Checks-Effects-Interactions Pattern](#) or applying OpenZeppelin [ReentrancyGuard](#).

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0xb1fd350644dbd1a7ec779f1f816f8ec4f4761f3](#).

0X0-01 | REDUNDANT CODE COMPONENT

Category	Severity	Location	Status
Volatile Code	● Informational	0x00476072b53902fb201905b49ae296f94b8fabf6 (MrMintToken Vesting): 768	● Resolved

Description

In `MrMintTokenVesting.sol` the modifier `onlyIfVestingScheduleExists` 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

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00](#).

0X6-01 | FAILED TRANSFERS WILL EMIT EVENT

Category	Severity	Location	Status
Logical Issue	● Informational	0x603b6586084e6ccdab7d270885b93a9bd0caa2cf (MrMint v2): 771~774	● Resolved

Description

The function `transferAnyBSC20Token()` makes no check on the value of `success` variable thus the event `TransferAnyBSC20Token` will be emitted also for failed transfers.

Recommendation

We recommend checking the value of `success` before emitting the event named `TransferAnyBSC20Token` as follows:

```
771     function transferAnyBSC20Token(address tokenAddress, address wallet, uint
tokens) public onlyMultiSigWallet returns (bool success) {
772         success = IBEP20(tokenAddress).transfer(wallet, tokens);
773         require(success, "BEP20 transfer failed");
774         emit TransferAnyBSC20Token(address(this), wallet, tokens);
775     }
```

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x95C08955d9B2293470b1b11d0A6AB86C21374591](#).

0X7-02 | DISCUSSION ON `_computeReleasableAmount`

Category	Severity	Location	Status
Logical Issue	● Informational	0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00 (MrMintT okenVesting v3): 985	● Acknowledged

Description

The latest version of `_computeReleasableAmount()` changed the following line

```
989     } else if (currentTime >=
vestingSchedule.start.add(vestingSchedule.duration)) {
```

to

```
989     } else if (currentTime >=
vestingSchedule.cliff.add(vestingSchedule.duration)) {
```

Known that `start < cliff < (start + duration) < (cliff + duration)`, the update above indicates the cliff is not part of the duration anymore, which is inconsistent with the original design.

Recommendation

We would like to check with the team about the change above.

Alleviation

[MrMint, 03/29/2023]: The team confirmed the logic change on vesting distribution and confirm it is intended design.

MMC-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Informational	MrMint.sol (MrMint): 444	● Resolved

Description

The following addresses in the input are not zero-checked before being used:

- In `MrMint.sol` the input `_liquidityFeewallet` to the function `setLiquidityFeewallet()`

Recommendation

We recommend adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0xb1ffd350644dbd1a7ec779f1f816f8ec4f4761f3](#).

MMC-03 | UNLOCKED COMPILER VERSION

Category	Severity	Location	Status
Language Specific	● Informational	MrMint.sol (MrMint): <u>Z</u>	● Resolved

Description

The contract `MrMint.sol` has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to different compiler versions. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.2;
```

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0xb1ffd350644dbd1a7ec779f1f816f8ec4f4761f3](https://etherscan.io/address/0xb1ffd350644dbd1a7ec779f1f816f8ec4f4761f3).

MMT-05 | DISCUSSION ON `_token`

Category	Severity	Location	Status
Logical Issue	● Informational	MrMintTokenVesting.sol (MrMintTokenVesting): 754	● Resolved

Description

When the state variable `_token` is set through `updateTokenAddress()`, its value can never be updated.

If the desired `_token` value is not set the following invocations may not work properly or revert unless `_token` is set:

- `getToken()`
- `getWithdrawableAmount()`
- `release()`

Recommendation

Recommend setting the `_token` address in the constructor or implementing validation on the `tokenAddressUpdated` flag. This would ensure the contract is used only when the `_token` address has been properly set.

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0xb1ffd350644dbd1a7ec779f1f816f8ec4f4761f3](#).

MMT-06 | DISCUSSION ON PRECONDITIONS NEEDED TO CREATE VESTINGS

Category	Severity	Location	Status
Logical Issue	● Informational	MrMintTokenVesting.sol (MrMintTokenVesting): 866-902	● Resolved

Description

In the current codebase, the contract owner can create vestings arbitrarily for a specific beneficiary.

Recommendation

We would like to check with the team about the business logic related to creating vesting.

Alleviation

[MrMint]: "In business logic we need to define a locking period for team wallets for 3 years, after completing 3 years contract will issue locked amount in next 3 years as monthly basis."

MMT-08 | MISSING ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	● Informational	MrMintTokenVesting.sol (MrMintTokenVesting): 769 , 777 , 778	● Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00](#).

MMT-11 | FUNCTION WILL REVERT IF HOLDER ADDRESS HAS NO PREVIOUS VESTING

Category	Severity	Location	Status
Logical Issue	● Informational	MrMintTokenVesting.sol (MrMintTokenVesting): 1014	● Resolved

Description

The function `getLastVestingScheduleForHolder()` will revert if `holder` address has no previous vesting since `holdersVestingCount[holder] - 1` will be equal to a negative value. If the holder has no vesting associated the expected outcome should be an error message "No vesting found for the holder address in input"

Recommendation

We recommend adding the following check in `getLastVestingScheduleForHolder()` before calculating the return value to the function

```
1010     function getLastVestingScheduleForHolder(address holder)
1011         public
1012         view
1013         returns(VestingSchedule memory){
1014             require(holdersVestingCount[holder] > 0, "No vesting found for the
holder address in input");
1015             return
vestingSchedules[computeVestingScheduleIdForAddressAndIndex(holder,
holdersVestingCount[holder] - 1)];
1016         }
```

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00](#).

MMT-14 | UNNECESSARY payable ADDRESS TYPE

Category	Severity	Location	Status
Language Specific	● Informational	MrMintTokenVesting.sol (MrMintTokenVesting): 946	● Resolved

Description

In `release()` function, the address `vestingSchedule.beneficiary` is casted to `address payable` type, but no base currency tokens are transferred in or out from that address. So it is an unnecessary casting operation.

Recommendation

We recommend removing the cast from `address` type to `address payable` type.

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00](#).

TES-01 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	● Informational	MrMintTokenVesting.sol (MrMintTokenVesting): 792 , 866 , 908 , 929 ; MrMint.sol (MrMint): 443 , 447 , 451 , 746 , 756	● Resolved

Description

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

The following centralized functions are not emitting any event:

- in `MrMintTokenVesting.sol` the function `updateTokenAddress()` at line 792
- in `MrMintTokenVesting.sol` the function `createVestingSchedule()` at line 866
- in `MrMintTokenVesting.sol` the function `revoke()` at line 908
- in `MrMintTokenVesting.sol` the function `release()` at line 929
- in `MrMint.sol` the function `setLiquidityFeeWallet()` at line 443
- in `MrMint.sol` the function `excludeFromFee()` at line 447
- in `MrMint.sol` the function `includeInFee()` at line 451
- in `MrMint.sol` the function `withdraw()` at line 746
- in `MrMint.sol` the function `transferAnyBSC20Token()` at line 756

Recommendation

We recommend emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00](#).

TES-02 | UNUSED EVENTS

Category	Severity	Location	Status
Coding Style	● Informational	MrMintTokenVesting.sol (MrMintTokenVesting): 762 , 763 ; MrMint.sol (MrMint): 379	● Resolved

Description

In `MrMintTokenVesting.sol` the following events are declared but never emitted:

- `Released` at line 762
- `Revoked` at line 763

In `MrMint.sol` the following event is declared but never emitted:

- `BuyToken` at line 379

Recommendation

We advise removing the unused events or emitting them in the intended functions.

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0x7a7fd4a0703b0872acb85ecdf0c4c59dd83cdb00](#).

OPTIMIZATIONS | MRMINT - AUDIT

ID	Title	Category	Severity	Status
0X0-04	Unnecessary Use Of SafeMath	Gas Optimization	Optimization	● Acknowledged
0X8-02	State Variable Should Be Declared Constant	Gas Optimization	Optimization	● Resolved

0X0-04 | UNNECESSARY USE OF SAFEMATH

Category	Severity	Location	Status
Gas Optimization	● Optimization	0x00476072b53902fb201905b49ae296f94b8fabf6 (MrMintTokenVesting)	● Acknowledged

Description

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

Recommendation

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

Alleviation

[MrMint, 03/29/2023]: For safemath library we would like to keep it for backward compatibility with old solidity version. We assume there is no harm to use safemath for now.

OX8-02 | STATE VARIABLE SHOULD BE DECLARED CONSTANT

Category	Severity	Location	Status
Gas Optimization	● Optimization	0x8e846a318510138bdca8c524c4604e321372ca00 (MrMint): 392 , 393	● Resolved

Description

State variables that never change should be declared as `constant` to save gas.

```
392     uint256 public liquidityFeePercent = 0.5 * 10**2; // 0.5% Percent of  
liquidity fee;
```

- `liquidityFeePercent` should be declared `constant`.

```
393     uint256 public burnFeePercent = 0.5 * 10**2; // 0.5% Percent of burn fee;
```

- `burnFeePercent` should be declared `constant`.

Recommendation

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

Alleviation

[MrMint, 03/29/2023]: The team resolved this finding in contract [0xb1fd350644dbd1a7ec779f1f816f8ec4f4761f3](#).

APPENDIX | MRMINT - AUDIT

Finding Categories

Categories	Description
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.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
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.

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.



