CERTIK

Security Assessment

RetroWorld

Jul 13th, 2021

Table of Contents

Summary

Overview

Project Summary

Audit Summary

Vulnerability Summary

Audit Scope

Findings

MCC-01 : Potential Undesired Code Behaviour

MCC-02 : Potential Loss of Pool Rewards

MCC-03 : Unclear Error Message

MCC-04 : Incompatibility With Deflationary Tokens

MCC-05 : Reentrancy Attack Risks

MCC-06 : Over Minted Token

MCC-07 : Centralization Risks

MCC-08 : Unhandled Return Values

MCC-09 : Missing Events Emission for Significant Transactions

MCC-10 : Different Solidity Versions

PPC-01 : Division by Zero

PPC-02 : Different Solidity Versions

Appendix

Disclaimer

About

Summary

This report has been prepared for Retroworld to discover issues and vulnerabilities in the source code of the RetroWorld project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review 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:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	RetroWorld
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0x5bc62934450ed677Ba5F7A64fAc73D447E4005a4 https://bscscan.com/address/0x449e6633A0271616f2D9712D0220d309dd61b8FE https://bscscan.com/address/0x9385eaaccb0789829354ed67d5b73d3f0816198d
Commit	

Audit Summary

Delivery Date	Jul 13, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
● Critical	0	0	0	0	0	0
● Major	0	0	0	0	0	0
● Medium	0	0	0	0	0	0
● Minor	6	0	0	0	6	0
● Informational	6	0	0	0	6	0
● Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum

There are a few depending injection contracts or addresses in the current project:

`devAddress`, `gldReferral` and `gld` for the contract `MasterChef`.

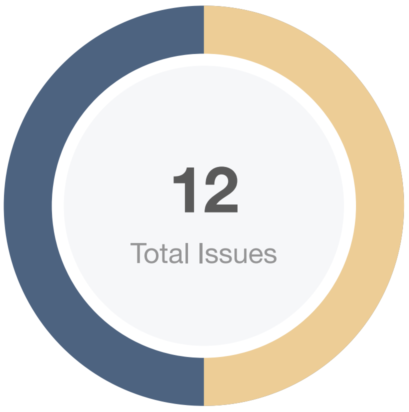
We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

To set up the project correctly, improve overall project quality and preserve upgradability, the following roles, are adopted in the codebase:

The `owner` role is adopted to add a new liquidity pool, set the existing liquidity pool's configuration, and update `MasterChef` configurations in the contract `MasterChef`.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**0** (0.00%)
🟨	**Medium**	**0** (0.00%)
🟨	**Minor**	**6** (50.00%)
🟦	**Informational**	**6** (50.00%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
MCC-01	Potential Undesired Code Behaviour	Logical Issue	● Minor	ⓘ Acknowledged
MCC-02	Potential Loss of Pool Rewards	Logical Issue	● Minor	ⓘ Acknowledged
MCC-03	Unclear Error Message	Coding Style	● Informational	ⓘ Acknowledged
MCC-04	Incompatibility With Deflationary Tokens	Logical Issue	● Minor	ⓘ Acknowledged
MCC-05	Reentrancy Attack Risks	Logical Issue	● Minor	ⓘ Acknowledged
MCC-06	Over Minted Token	Logical Issue	● Minor	ⓘ Acknowledged
MCC-07	Centralization Risks	**Centralization / Privilege**	● **Minor**	ⓘ **Acknowledged**
MCC-08	Unhandled Return Values	Coding Style	● Informational	ⓘ Acknowledged
MCC-09	Missing Events Emission for Significant Transactions	Coding Style	● Informational	ⓘ Acknowledged
MCC-10	Different Solidity Versions	Language Specific	● Informational	ⓘ Acknowledged
PPC-01	Division by Zero	Logical Issue	● Informational	ⓘ Acknowledged
PPC-02	Different Solidity Versions	Language Specific	● Informational	ⓘ Acknowledged

MCC-01 | Potential Undesired Code Behaviour

Category	Severity	Location	Status
Logical Issue	● Minor	projects/retroworld/contracts/MasterChef.sol: 1174~1181	ⓘ Acknowledged

Description

If a duplicate pool is detected in the for loop, the function will be terminated by the `return` statement on L#1178. The `require` statement on L#1181 is meaningless because if the variable `isDuplicateFound` evaluates to true, the function execution will never reach L#1178.

Recommendation

We advise the client to revise the implementation. To throw a proper error message by the `require` statement on L#1181, a feasible solution is replacing the `return` statement on L#1178 with a `break` statement.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

MCC-02 | Potential Loss of Pool Rewards

Category	Severity	Location	Status
Logical Issue	● Minor	projects/retroworld/contracts/MasterChef.sol: 1168, 1201	ⓘ Acknowledged

Description

The following code snippet in the functions `MasterChef.add()` and `MasterChef.set()` update the pools and distribute the reward if `_withUpdate` evaluates to `true`:

```
    if (_withUpdate) {
        massUpdatePools();
    }
```

However, in the case that `_withUpdate` evaluates to false, a significant loss of reward might happen.

The reward calculation in the function `MasterChef.updatePool()` is:

```
 2077         uint256 gldReward =
multiplier.mul(gldPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
```

Assuming there is only one pool with `pool.allocPoint == 50` and `totalAllocPoint == 50` at the beginning. Now we want to add another pool with `pool.allocPoint == 50`.

There will be two scenarios on calculating the pool reward,

Case 1: _withUpdate is `true` value.

- Step 1, distribute the reward and update the pool.
- Step 2, add or set the given pool information.

In this case, the reward calculated in step 1 is `gldReward = multiplier.mul(gldPerBlock).mul(50).div(50)`, which is equivalent to `multiplier.mul(gldPerBlock)`.

Case 2: _withUpdate is `false` value.

- Step 1, add or set the given pool information.

In this case, the reward calculated later will be gldReward = multiplier.mul(gldPerBlock).mul(50).div(100)`, which is half as much as the reward calculated in case 1.

Recommendation

We advise the client to remove the `_withUpdate` flag and always update pool rewards before updating pool information.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

MCC-03 | Unclear Error Message

Category	Severity	Location	Status
Coding Style	● Informational	projects/retroworld/contracts/MasterChef.sol: 1203	ⓘ Acknowledged

Description

The following `require` statement's error message does not reflect the error clearly:

```
1203        require(preMatureFee <= 10000, "set: invalid deposit fee basis points");
```

Recommendation

We advise the client to implement a proper error message to describe the error.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

MCC-04 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	● Minor	projects/retroworld/contracts/MasterChef.sol: 1287, 1326, 1342	ⓘ Acknowledged

Description

The contract `MasterChef` updates the user's balance after the user deposits tokens to the contract or withdraws tokens from the contract based on the input of token transfer. However, the input of token transfer does not always match the result of the token transfer. This fact might bring unexpected balance inconsistencies.

For example, a deflationary token AToken charges 10% of the transfer amount as transfer fees. If a user deposits 100 AToken to contract `MasterChef`, the contract will only receive 90 AToken, while `user.amount` is set to 100. When the user tries to withdraw the token from the contract, he will not be able to withdraw 100, which shows as `user.amount`, because, if he is the only person who deposited the contract, the contract only has 90 AToken.

Recommendation

We advise the client to carefully review the implementation of `MasterChef.poolInfo[_pid].lpToken` before setting a pool and ensure a deflationary token will not be used as `MasterChef.poolInfo[_pid].lpToken`.

Alleviation

[Retro Finance]: The team acknowledged the issue and ensure a deflationary token will not be utilized for the LP's

MCC-05 | Reentrancy Attack Risks

Category	Severity	Location	Status
Logical Issue	● Minor	projects/retroworld/contracts/MasterChef.sol: 1239, 1247, 1201, 1168, 1303, 1274, 1394	ⓘ Acknowledged

Description

The aforementioned functions emit events or update state variables after external calls. Thus they might be vulnerable to reentrancy attacks.

Recommendation

We advise the client to apply OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

MCC-06 | Over Minted Token

Category	Severity	Location	Status
Logical Issue	● Minor	projects/retroworld/contracts/MasterChef.sol: 1261, 1373	ⓘ Acknowledged

Description

The function `MasterChef.updatePool()` mints 100% + 10% (dev fee is included as 10% of the 100%) `gld` tokens of total rewards.

The function `MasterChef.safeGldTransfer()` mints the tokens that equal to the input `_amount`. However, according to the comment on line 1369, this function is used for rounding errors, which means it only needs to mint at most 1 `gld` token to cover the rounding error. The following code snippet apparently over-mints `gld` tokens.

```
1373   gld.mint( _amount);
```

Recommendation

For the function `MasterChef.updatePool()`, we advise the client to mint 100% of the block reward instead of 100% + 10% since the dev fee is included as 10% of the 100%.

For the function `MasterChef.safeGldTransfer()`, we advise the client to mint only 1 token to cover the rounding error.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

MCC-07 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	● Minor	projects/retroworld/contracts/MasterChef.sol: 1168, 1201, 1394, 1401, 1406	ⓘ Acknowledged

Description

The role `owner` has the following permissions without obtaining the consensus of the community:

1. Add a new pool by calling the function `MasterChef.add()`;
2. Modify `allocPoint`, `depositFeeBP` and `harvestInterval` of a pool by calling the function `MasterChef.set()`;
3. Modify the reward emission rate by calling the function `MasterChef.updateEmissionRate()`;
4. Set the address of the referral contract by calling the function `MasterChef.setGldReferral()`;
5. Set the referral commission rate by calling the function `MasterChef.setReferralCommissionRate()`.

Recommendation

We advise the client to handle the `owner` account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

1. `Timelock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team plans to implement multi sig and a DAO voting feature to be implemented to ensure full transparency of all actions to be carried out.

MCC-08 | Unhandled Return Values

Category	Severity	Location	Status
Coding Style	● Informational	projects/retroworld/contracts/MasterChef.sol: 1376, 1373	ⓘ Acknowledged

Description

The functions `gld.transfer()` and `gld.mint()` are not void-returning functions. Ignoring their return values, especially when their return values might represent the status if the transaction is executed successfully, might cause unexpected exceptions.

Recommendation

We advise the client to handle the return values of the functions `gld.transfer()` and `gld.mint()`.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

MCC-09 | Missing Events Emission for Significant Transactions

Category	Severity	Location	Status
Coding Style	● Informational	projects/retroworld/contracts/MasterChef.sol: 1381, 1387, 1401, 1406	ⓘ Acknowledged

Description

The aforementioned functions affect the following sensitive variables:

- `MasterChef.setGldReferral()` can update the address for the contract `gldReferral`;
- `MasterChef.setReferralCommissionRate()` can update the referral commission rate.;
- `MasterChef.setDevAddress` can update the `devAddress` by previous dev.
- `MasterChef.setFeeAddress` can update the `feeAddress` by previous feeAddress holder.

These functions should emit an event as the notification to the users for any change related to the aforementioned variables.

Recommendation

We advise the client to emit events for the aforementioned functions.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

MCC-10 | Different Solidity Versions

Category	Severity	Location	Status
Language Specific	● Informational	projects/retroworld/contracts/MasterChef.sol: 7, 33, 101, 197, 358, 656, 684, 706, 895, 996, 1059	ⓘ Acknowledged

Description

While `pragma solidity >=0.6.0 <0.8.0` is specified on L#7, L#33, L#197, L#996the following solidity versions are specified:

- `pragma solidity >=0.6.4` is specified on L#101;
- `pragma solidity >=0.4.0` is specified on L#358;
- `pragma solidity ^0.6.0` is specified on L#656, L#895;
- `pragma solidity 0.6.12` is specified on L#684, L#1059;
- `pragma solidity >=0.6.2 <0.8.0` is specified on L#706.

Recommendation

We recommend specifying the same Solidity version within this file.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

PPC-01 | Division by Zero

Category	Severity	Location	Status
Logical Issue	● Informational	projects/retroworld/contracts/PancakePair.sol: 245, 441~442	ⓘ Acknowledged

Description

The aforementioned code snippets perform divisions without proper division by zero checks.

Recommendation

We advise the client to perform proper division by zero checks before performing division to avoid unexpected exceptions.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

PPC-02 | Different Solidity Versions

Category	Severity	Location	Status
Language Specific	● Informational	projects/retroworld/contracts/PancakePair.sol: 7, 62, 88, 108, 203, 229, 252, 272, 300	ⓘ Acknowledged

Description

While `pragma solidity >=0.5.0` is specified on L#7, L#62, L#252 and L#272, `pragma solidity =0.5.16` is specified on L#88, L#108, L#203, L#229 and L#300.

Recommendation

We recommend specifying the same Solidity version within this file.

Alleviation

[Retro Finance]: The team acknowledged the issue and decided to unchanged the codebase. The team has considered and will update any dynamic actions to the community regarding any changes being made in ample time.

Appendix

Finding Categories

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.

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.

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.

There are a few depending injection contracts or addresses in the current project:

`devAddress`, `gld` and `gldReferral` for the contract `MasterChef`.

We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

To set up the project correctly, improve overall project quality and preserve upgradability, the following role, are adopted in the codebase:

The `owner` role is adopted to add a new liquidity pool, set the existing liquidity pool's configuration, and update `MasterChef` configurations in the contract `MasterChef`.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

Disclaimer

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 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 without CertiK's prior written consent.

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.

About

CERTIK