CERTIK

Security Assessment

CyberTime

Apr 19th, 2021

Summary

This report has been prepared for CyberTime smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, 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 37 findings that ranged from major 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	CyberTime
Platform	BSC
Language	Solidity
Codebase	https://github.com/cybertime-eth/cybertime-smart-contracts/
Commits	1. 054f4c066a07dc38d16940f8cb8402c41003c8aa 2. abf8d521bd733db59a675a5b2aeff23eb86a757e 3. 0977f93fcc19a1ad8b232f5444c5deefe24b7138

Audit Summary

Delivery Date	Apr 19, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	42
Critical	0
Major	1
Minor	6
Informational	35
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
ACK	auction/Auction.sol	75cc882f998ba03c051f379266ee02c601dfa63629271c165394b94333459214
CTF	farming/CTFFarming.sol	56bbd07d0565532d66929933baa1459d6dac98694c6bcd226944d36817543db5
NFT	farming/NFTLFarming.sol	f47c5db29ac459dcbdd8cf304b96df22809f2d1df5479bce0ffc05795b2e1208
CTT	tokens/ERC20/CTFToken.sol	8d85538b10d4d75303d303f8e1330b5c0f5313c28d2f83f6b5dd07e55bb1a891
NFL	tokens/ERC20/NFTLToken.sol	b3c4557cf0f4e59836b45d0af08e2c2a1d50ecd56d96450517d261e9dac51ab1

Centralization

To initial setup project correctly, improve overall project quality, preserve the upgradability, the following functions are adopted in the codebase:

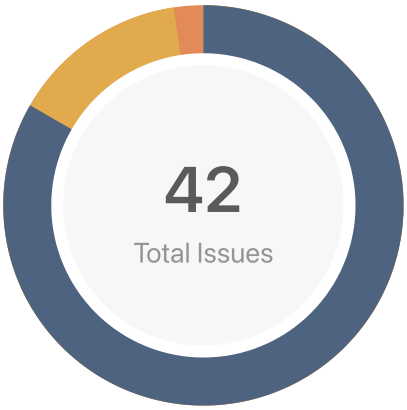
- `add()` in `CTFFarming`, `NFTLFarming` contract
- `set()` in `CTFFarming`, `NFTLFarming` contract
- `changeCTFPerBlock()` in `CTFFarming`, `NFTLFarming` contract
- `dev()` in `CTFFarming`, `NFTLFarming` contract
- `add()` in `Auction` contract
- `changeIncrementRate()` in `Auction` contract
- `changeSalesDistribution()` in `Auction` contract
- `changeBurnRate()` in `Auction` contract
- `distributeSales()` in `Auction` contract

The `onlyDev` can change the value of `auction.incrementRate` , `distribution` and `burnRate` through functions `changeIncrementRate()` , `changeSalesDistribution()` and `changeBurnRate()`. These codes obviously are dangerous and our analysis vectors regarded these as Critical severity considering if the private key of the `onlyDev` is lost.

The advantage of the above functions in the codebase is that the client reserves the ability to adjust the project according to the runtime require to best serve the community. It is also worthy of note the potential drawbacks of these functions, which should be clearly stated through client's action/plan on how to prevent abuse of the these functionalities

To improve the trustworthiness of the project, any dynamic runtime updates in the project should be notified to the community. Any plan to implement aforementioned functions must be also considered to adopt Timelock with reasonable delay to allow the user to withdraw their funds, Multisig with community-selected 3-party independent co-signers, and/or DAO with transparent governance with the project's community in the project to manage sensitive role accesses.

Findings



Critical	0 (0.00%)
Major	1 (2.38%)
Minor	6 (14.29%)
Informational	35 (83.33%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
ACK-01	Variable Declare as Immutable	Gas Optimization	Informational	Resolved
ACK-02	Lack of Input Validation	Volatile Code	Informational	Resolved
ACK-03	Missing Error Message	Logical Issue	Informational	Pending
ACK-04	Missing Emits Events	Coding Style	Informational	Resolved
ACK-05	Missing Emits Events	Coding Style	Informational	Resolved
ACK-06	Code Simplify	Gas Optimization	Informational	Resolved
ACK-07	Centralized Risk	Control Flow	Major	Pending
CTF-01	Missing Input Validation	Logical Issue	Informational	Resolved
CTF-02	Input Validation Check	Gas Optimization	Informational	Resolved
CTF-03	Comment Typo	Coding Style	Informational	Resolved
CTF-04	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	Pending
CTF-05	Comment Typo	Coding Style	Informational	Resolved
CTF-06	Pending Amount Validation	Gas Optimization	Informational	Resolved
CTF-07	Validation Check	Gas Optimization	Informational	Pending
CTF-08	Unhandled Return Value	Logical Issue	Minor	Pending
CTF-09	Missing Emit Events	Gas Optimization	Informational	Resolved

ID	Title	Category	Severity	Status
CTF-10	Input Validation Check	Volatile Code	Informational	Pending
CTF-11	Input Validation Check	Volatile Code	Informational	Pending
CTF-12	Incompatibility With Deflationary Tokens	Logical Issue	Minor	Pending
CTF-13	Input Validation Check	Volatile Code	Informational	Resolved
CTF-14	Input Validation Check	Volatile Code	Informational	Pending
CTF-15	Missing Emit Events	Gas Optimization	Informational	Resolved
CTT-01	Missing Emit Events	Gas Optimization	Informational	Resolved
CTT-02	Unhandled Return Value	Logical Issue	Minor	Pending
CTT-03	Missing Emit Events	Gas Optimization	Informational	Resolved
NFL-01	Missing Emit Events	Gas Optimization	Informational	Resolved
NFL-02	Unhandled Return Value	Logical Issue	Minor	Pending
NFL-03	Missing Emit Events	Gas Optimization	Informational	Resolved
NFT-01	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	Pending
NFT-02	Missing Input Validation	Logical Issue	Informational	Resolved
NFT-03	Input Validation Check	Gas Optimization	Informational	Resolved
NFT-04	Comment Typo	Coding Style	Informational	Resolved
NFT-05	Incompatibility With Deflationary Tokens	Logical Issue	Minor	Pending
NFT-06	Pending Amount Validation	Gas Optimization	Informational	Resolved
NFT-07	Input Validation Check	Volatile Code	Informational	Resolved
NFT-08	Input Validation Check	Volatile Code	Informational	Resolved
NFT-09	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	Pending
NFT-10	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	Pending
NFT-11	Missing Emit Events	Gas Optimization	Informational	Pending
NFT-12	Unhandled Return Value	Logical Issue	Minor	Pending

ID	Title	Category	Severity	Status
NFT-13	Input Validation Check	Volatile Code	Informational	Pending
NFT-14	Missing Emit Events	Gas Optimization	Informational	Resolved

ACK-01 | Variable Declare as Immutable

Category	Severity	Location	Status
Gas Optimization	Informational	auction/Auction.sol: 16	Resolved

Description

Variable that only be assigned in constructor can be declare as `immutable`. Immutable state variables can be assigned during contract creation, but will remain constant throughout the life-time of a deployed contract. The big advantage of immutable is that reading them is significantly cheaper than reading from regular state variables, since immutables will not be stored in storage, but their values will be directly inserted into the runtime code.

Recommendation

We recommend using immutable state variable for `NFTL`

```
16  ...
17 IERC20 immutable  public NFTL; // add ERC20 interface to NFTL address
18
19  ...
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

ACK-02 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	auction/Auction.sol: 167	Resolved

Description

The given input `_expiry` and `_quantity` is missing a valid parameter check.

Recommendation

We recommend adding validation for `_expiry` as following:

```
require(_quantity > 0, "auction: _quantity should be greater than zero");
require(_expiry > block.timestamp, "auction: _expiry should be a future block");
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

ACK-03 | Missing Error Message

Category	Severity	Location	Status
Logical Issue	Informational	auction/Auction.sol: 170	Pending

Description

The require statement is missing a clear error message

Recommendation

We recommend adding error message for better error tracking, and actionable information by the users

ACK-04 | Missing Emits Events

Category	Severity	Location	Status
Coding Style	Informational	auction/Auction.sol: 192	Resolved

Description

Function changeSalesDistribution which is an important function. Missing event makes it difficult to track off-chain parameter changes. An event should be emitted for significant transactions like this.

Recommendation

We recommend emitting an event to log the update of onlyDev in changeSalesDistribution.

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`

ACK-05 | Missing Emits Events

Category	Severity	Location	Status
Coding Style	Informational	auction/Auction.sol: 197	Resolved

Description

Function `changeBurnRate` which is an important function. Missing event makes it difficult to track off-chain parameter changes. An event should be emitted for significant transactions like this.

Recommendation

We recommend emitting an event to log the update of onlyDev in `changeBurnRate`.

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`

ACK-06 | Code Simplify

Category	Severity	Location	Status
Gas Optimization	Informational	auction/Auction.sol: 82~88	Resolved

Description

The aforementioned lines can improve by using memory variable for gas optimization

Recommendation

We recommend the code can be simplified as following for optimization:

```
82          uint256 newBidderAmount = auction.bids[msg.sender].add(_amt);
83          // update the amount user has staked
84          auction.bids[msg.sender] = newBidderAmount;
85
86          // update the highest bid amount
87          if (auction.highestBidAmt < newBidderAmount) {
88              auction.highestBidAmt = newBidderAmount);
89          }
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

ACK-07 | Centralized Risk

Category	Severity	Location	Status
Control Flow	Major	auction/Auction.sol: 159, 179, 196, 200, 191	Pending

Description

`onlyDev` is an important role in the contract. The owner address can operate on following functions:

- `add()`
- `changeIncrementRate()`
- `changeSalesDistribution()`
- `distributeSales()`

These functions can be invoked to manipulate the variables `_newDistribution`, and `_newBurnRate` , which will allow `onlyDev` to transfer any amount of token to any address with the code snippet.

Recommendation

We advise the client to carefully manage the project's private key and avoid any potential risks of being hacked. We also advise the client to adopt Timelock with reason delay to allow the user to withdraw their funds, Multisig with community-selected 3-party independent co-signers, and/or DAO with transparent governance with the project's community in the project to manage sensitive role accesses.

CTF-01 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	farming/CTFFarming.sol: 150	Resolved

Description

Missing validation for inputs `_from` && `_to`.

Recommendation

We advice adding validation for `_from` && `_to` to prevent return misleading result

```
150 function getMultiplier(uint256 _from, uint256 _to) public  view   returns (uint256)
151 {
152     require(_from <= _to, "_from must be less than or equal to _to");
    }
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

CTF-02 | Input Validation Check

Category	Severity	Location	Status
Gas Optimization	Informational	farming/CTFFarming.sol: 138~141	Resolved

Description

The code will not changes any parameter when _allocPoint is the same as the current poolInfo[_pid].allocPoint.

Recommendation

We recommend adding a validation check when the poolInfo[_pid].allocPoint != _allocPoint

```
138 ...
139 if (poolInfo[_pid].allocPoint  != _allocPoint) {
140 totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(
141             _allocPoint
142         );
143         poolInfo[_pid].allocPoint = _allocPoint;
144 }
145 ...
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

CTF-03 | Comment Typo

Category	Severity	Location	Status
Coding Style	Informational	farming/CTFFarming.sol: 54	Resolved

Description

The linked comment statement contains a typo in its body, poitns -> points.

Recommendation

We advise that the comment text is corrected.

Alleviation

The comment typo was properly fixed.

CTF-04 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	farming/CTFFarming.sol: 130, 270, 282	Pending

Description

There's no sanity check to validate if a pool is existing. The current implementation simply relies on the implicit, compiler-generated bound-checks of arrays to ensure the pool index stays within the array range [0, poolInfo.length-1]. However, considering the importance of validating given pools and their numerous occasions, a better alternative is to make explicit the sanity checks by introducing a new modifier.

Recommendation

Apply necessary sanity checks to ensure the given `_pid` is legitimate by adding a new modifier validatePool to functions `set()`, `claim()`, `emergencyWithdraw()`.

```
function set(
        uint256 _pid,
        uint256 _allocPoint,
        bool _withUpdate
    ) validatePoolByPid(_pid) public onlyDev {
....
}
```

CTF-05 | Comment Typo

Category	Severity	Location	Status
Coding Style	Informational	farming/CTFFarming.sol: 186	Resolved

Description

The linked comment statement contains a typo in its body, vairable -> variable.

Recommendation

We advise that the comment text is corrected.

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

CTF-06 | Pending Amount Validation

Category	Severity	Location	Status
Gas Optimization	Informational	farming/CTFFarming.sol: 231	Resolved

Description

When the pending amount is 0, there is no need to process `safeCTFTransfer()`

Recommendation

We recommend adding a validation check before process the aforementioned line:

```
231 if (user.amount > 0) {
232             ...
233             if (pending > 0 ) {
234 safeCTFTransfer(msg.sender, pending);
235 }
236         }
```

Alleviation

[CyberTime]: The team addressed the issue and reflected in the commit `0977f93fcc19a1ad8b232f5444c5deefe24b7138`

CTF-07 | Validation Check

Category	Severity	Location	Status
Gas Optimization	Informational	farming/CTFFarming.sol: 231	Pending

Description

Adding Validation check for the fee amount, when fee is zero, the lp token has nothing to transfer.

Recommendation

We recommend adding validation check for

```
uint256 fees;
if (_amount > 0) {
 fees = _amount.mul(2).div(100);

        user.amount = user.amount.add(_amount.sub(fees));
        user.rewardDebt = user.amount.mul(pool.accCTFPerShare).div(1e12);

        pool.lpToken.safeTransferFrom(
            address(msg.sender),
            address(this),
            _amount
        );

if (fees > 0) {
// send fees in the form of LP tokens to feeReceiver addr
        pool.lpToken.transfer(lpFeeReceiver, fees);
}
}
```

CTF-08 | Unhandled Return Value

Category	Severity	Location	Status
Logical Issue	Minor	farming/CTFFarming.sol: 296, 298	Pending

Description

token's transfer is not void-returning functions. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation

We recommend checking the output of the aforementioned functions before continuing processing.

CTF-09 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	farming/CTFFarming.sol: 303, 279	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `dev()`
- `ctfPerBlock()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function.

```
event SetDev(address indexed user, address indexed _devaddr);

function dev(address _devaddr) public onlyDev {
    devaddr = _devaddr;
    emit SetDev(msg.sender, _devaddr);
}
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

CTF-10 | Input Validation Check

Category	Severity	Location	Status
Volatile Code	Informational	farming/CTFFarming.sol: 252	Pending

Description

When pending amount is 0, `safeCTFTransfer` will not have any direct impact.

Recommendation

We advice adding check for only execute the safeCTFTransfer when pending amount is greater than 0.

```
1 if (pending > 0) {
2 safeCTFTransfer(msg.sender, pending);
3 }
```

CTF-11 | Input Validation Check

Category	Severity	Location	Status
Volatile Code	Informational	farming/CTFFarming.sol: 260, 265	Pending

Description

When the given input `_amount` is 0, aforementioned lines will not have any direct impact.

Recommendation

We advice adding check for only execute the safeCTFTransfer when pending amount is greater than 0.

```
1 if (_amount  > 0) {
2 user.amount = user.amount.sub(_amount);
3 pool.lpToken.safeTransfer(address(msg.sender), _amount);
4 }
```

CTF-12 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	farming/CTFFarming.sol: 222, 252	Pending

Description

The users deposit LP tokens into the CTFFarming pool and in return get a proportionate share of the pool's rewards. Later on, the users can withdraw their assets from the pool. In this procedure, `deposit()` and `withdraw()` are involved in transferring users' assets into (or out of) the CTFFarming pool. When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistencies.

Recommendation

Regulate the set of LP tokens supported in CyberTime and, if there is a need to support deflationary tokens, add necessary mitigation mechanisms to keep track of accurate balances.

CTF-13 | Input Validation Check

Category	Severity	Location	Status
Volatile Code	Informational	farming/CTFFarming.sol: 263, 268, 278	Resolved

Description

When the given input `pending` is 0, aforementioned lines will not have any direct impact.

Recommendation

We advice adding check for only execute the safeCTFTransfer when pending amount is greater than 0.

```
1 if (pending  > 0) {
2 safeCTFTransfer(msg.sender, pending);
3 }
```

Alleviation

[CyberTime]: The team addressed the issue and reflected in the commit `0977f93fcc19a1ad8b232f5444c5deefe24b7138`

CTF-14 | Input Validation Check

Category	Severity	Location	Status
Volatile Code	Informational	farming/CTFFarming.sol: 237, 240~244	Pending

Description

When the given input `_amount` is 0, aforementioned lines will not have any direct impact.

Recommendation

We advice adding check for only execute the safeCTFTransfer when pending amount is greater than 0.

```
1 if (_amount  > 0) {
2 user.amount = user.amount.sub(_amount);
3 pool.lpToken.safeTransfer(address(msg.sender), _amount);
4 }
```

CTF-15 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	farming/CTFFarming.sol: 308~309	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `dev()`
- `ctfPerBlock()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function.

```
event SetDev(address indexed user, address indexed _devaddr);

function dev(address _devaddr) public onlyDev {
    devaddr = _devaddr;
    emit SetDev(msg.sender, _devaddr);
}
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

CTT-01 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	tokens/ERC20/CTFToken.sol: 27, 46	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `mint()`
- `migrate()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function.

```
event SetAddFarmingContract(address indexed farmingContractAddr, address indexed
_admin);

function addFarmingContract(address _farmingContractAddr) public{
    require(msg.sender == owner, "CTFToken: You're not owner");
        require(
            farmingContract == address(0),
            "Farming Contract Already Added"
        );
        farmingContract = _farmingContractAddr;
    emit SetAddFarmingContract(_farmingContractAddr, msg.sender);
}
```

Alleviation

[CyberTime]: The team addressed the issue and reflected in the commit `b6445c9fc067d8beda0b9c45d14574b987781027`

CTT-02 | Unhandled Return Value

Category	Severity	Location	Status
Logical Issue	Minor	tokens/ERC20/CTFToken.sol: 52	Pending

Description

token's transfer is not void-returning functions. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation

We recommend checking the output of the aforementioned functions before continuing processing.

CTT-03 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	tokens/ERC20/CTFToken.sol: 36	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `mint()`
- `addFarmingContract()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function.

```
event SetAddFarmingContract(address indexed farmingContractAddr, address indexed
_admin);

function addFarmingContract(address _farmingContractAddr) public{
    require(msg.sender == owner, "CTFToken: You're not owner");
        require(
            farmingContract == address(0),
            "Farming Contract Already Added"
        );
        farmingContract = _farmingContractAddr;
    emit SetAddFarmingContract(_farmingContractAddr, msg.sender);
}
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`

NFL-01 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	tokens/ERC20/NFTLToken.sol: 28, 47	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `mint()`
- `addFarmingContract()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function.

```
event SetAddFarmingContract(address indexed farmingContractAddr, address indexed
_admin);

function addFarmingContract(address _farmingContractAddr) public{
    require(msg.sender == owner, "CTFToken: You're not owner");
        require(
            farmingContract == address(0),
            "Farming Contract Already Added"
        );
        farmingContract = _farmingContractAddr;
    emit SetAddFarmingContract(_farmingContractAddr, msg.sender);
}
```

Alleviation

[CyberTime]: The team addressed the issue and reflected in the commit `b6445c9fc067d8beda0b9c45d14574b987781027`

NFL-02 | Unhandled Return Value

Category	Severity	Location	Status
Logical Issue	Minor	tokens/ERC20/NFTLToken.sol: 53	Pending

Description

token's transfer is not void-returning functions. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation

We recommend checking the output of the aforementioned functions before continuing processing.

NFL-03 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	tokens/ERC20/NFTLToken.sol: 37	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `mint()`
- `addFarmingContract()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function.

```
event SetAddFarmingContract(address indexed farmingContractAddr, address indexed
_admin);

function addFarmingContract(address _farmingContractAddr) public{
    require(msg.sender == owner, "CTFToken: You're not owner");
        require(
            farmingContract == address(0),
            "Farming Contract Already Added"
        );
        farmingContract = _farmingContractAddr;
    emit SetAddFarmingContract(_farmingContractAddr, msg.sender);
}
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`

NFT-01 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	farming/NFTLFarming.sol: 136	Pending

Description

There's no sanity check to validate if a pool is existing. The current implementation simply relies on the implicit, compiler-generated bound-checks of arrays to ensure the pool index stays within the array range [0, poolInfo.length-1]. However, considering the importance of validating given pools and their numerous occasions, a better alternative is to make explicit the sanity checks by introducing a new modifier.

Recommendation

Apply necessary sanity checks to ensure the given `_pid` is legitimate by adding a new modifier validatePool to functions `set()`, `claim()`, `emergencyWithdraw()`.

```
function set(
        uint256 _pid,
        uint256 _allocPoint,
        bool _withUpdate
    ) validatePoolByPid(_pid) public onlyDev {
....
}
```

NFT-02 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	farming/NFTLFarming.sol: 156	Resolved

Description

Missing validation for inputs `_from` && `_to`.

Recommendation

We advice adding validation for `_from` && `_to` to prevent return misleading result

```
150 function getMultiplier(uint256 _from, uint256 _to) public  view   returns (uint256)
151 {
152     require(_from <= _to, "_from must be less than or equal to _to");
    }
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

NFT-03 | Input Validation Check

Category	Severity	Location	Status
Gas Optimization	Informational	farming/NFTLFarming.sol: 144~147	Resolved

Description

The code will not changes any parameter when _allocPoint is the same as the current poolInfo[_pid].allocPoint.

Recommendation

We recommend adding a validation check when the poolInfo[_pid].allocPoint != _allocPoint

```
138 ...
139 if (poolInfo[_pid].allocPoint  != _allocPoint) {
140 totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(
141             _allocPoint
142         );
143         poolInfo[_pid].allocPoint = _allocPoint;
144 }
145 ...
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

NFT-04 | Comment Typo

Category	Severity	Location	Status
Coding Style	Informational	farming/NFTLFarming.sol: 192	Resolved

Description

The linked comment statement contains a typo in its body, vairable -> variable.

Recommendation

We advise that the comment text is corrected.

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

NFT-05 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	farming/NFTLFarming.sol: 231	Pending

Description

The users deposit LP tokens into the CTFFarming pool and in return get a proportionate share of the pool's rewards. Later on, the users can withdraw their assets from the pool. In this procedure, `deposit()` and `withdraw()` are involved in transferring users' assets into (or out of) the CTFFarming pool. When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistencies.

Recommendation

Regulate the set of LP tokens supported in CyberTime and, if there is a need to support deflationary tokens, add necessary mitigation mechanisms to keep track of accurate balances.

NFT-06 | Pending Amount Validation

Category	Severity	Location	Status
Gas Optimization	Informational	farming/NFTLFarming.sol: 240	Resolved

Description

When the pending amount is 0, there is no need to process `safeCTFTransfer()`

Recommendation

We recommend adding a validation check before process the aforementioned line:

```
231 if (user.amount > 0) {
232             ...
233             if (pending > 0 ) {
234 safeCTFTransfer(msg.sender, pending);
235 }
236         }
```

Alleviation

[CyberTime]: The team addressed the issue and reflected in commit `0977f93fcc19a1ad8b232f5444c5deefe24b7138`

NFT-07 | Input Validation Check

Category	Severity	Location	Status
Volatile Code	Informational	farming/NFTLFarming.sol: 268, 285	Resolved

Description

When pending amount is 0, `safeCTFTransfer` will not have any direct impact.

Recommendation

We advice adding check for only execute the safeNFTLTransfer when pending amount is greater than 0.

```
1 if (pending > 0) {
2 safeNFTLTransfer(msg.sender, pending);
3 }
```

Alleviation

[CyberTime]: The team addressed the issue and reflected in commit `0977f93fcc19a1ad8b232f5444c5deefe24b7138`

NFT-08 | Input Validation Check

Category	Severity	Location	Status
Volatile Code	Informational	farming/NFTLFarming.sol: 270	Resolved

Description

When the given input `_amount` is 0, aforementioned lines will not have any direct impact.

Recommendation

We advice adding check for only execute the safeCTFTransfer when pending amount is greater than 0.

```
1 if (_amount  > 0) {
2 user.amount = user.amount.sub(_amount);
3 pool.lpToken.safeTransfer(address(msg.sender), _amount);
4 }
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

NFT-09 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	farming/NFTLFarming.sol: 275	Pending

Description

There's no sanity check to validate if a pool is existing. The current implementation simply relies on the implicit, compiler-generated bound-checks of arrays to ensure the pool index stays within the array range [0, poolInfo.length-1]. However, considering the importance of validating given pools and their numerous occasions, a better alternative is to make explicit the sanity checks by introducing a new modifier.

Recommendation

Apply necessary sanity checks to ensure the given `_pid` is legitimate by adding a new modifier validatePool to functions `set()`, `claim()`, `emergencyWithdraw()`.

```
function claim(
        uint256 _pid

    ) validatePoolByPid(_pid) public {
....
}
```

Alleviation

We have discussed this issue with the team. For the same reason as outlined in Section 3.3, because the MasterChef contract is already live (with a huge amount of assets), any change needs to be deemed necessary. In this particular case, the team prefers not modifying the code as the compiler-generated bounds-checking is already in place.

NFT-10 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	farming/NFTLFarming.sol: 289	Pending

Description

There's no sanity check to validate if a pool is existing. The current implementation simply relies on the implicit, compiler-generated bound-checks of arrays to ensure the pool index stays within the array range [0, poolInfo.length-1]. However, considering the importance of validating given pools and their numerous occasions, a better alternative is to make explicit the sanity checks by introducing a new modifier.

Recommendation

Apply necessary sanity checks to ensure the given `_pid` is legitimate by adding a new modifier validatePool to functions `set()`, `claim()`, `emergencyWithdraw()`.

```
function emergencyWithdraw(
        uint256 _pid
    ) validatePoolByPid(_pid) public {
....
}
```

NFT-11 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	farming/NFTLFarming.sol: 311, 321	Pending

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `dev()`
- `changeNFTLPerBlock()`
- `updateTeamShare()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function.

```
event SetDev(address indexed user, address indexed _devaddr);

function dev(address _devaddr) public onlyDev {
    devaddr = _devaddr;
    emit SetDev(msg.sender, _devaddr);
}
```

NFT-12 | Unhandled Return Value

Category	Severity	Location	Status
Logical Issue	Minor	farming/NFTLFarming.sol: 303, 305	Pending

Description

token's transfer is not void-returning functions. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation

We recommend checking the output of the aforementioned functions before continuing processing.

NFT-13 | Input Validation Check

Category	Severity	Location	Status
Volatile Code	Informational	farming/NFTLFarming.sol: 242, 245~249	Pending

Description

When the given input `_amount` is 0, aforementioned lines will not have any direct impact.

Recommendation

We advice adding check for only execute the safeCTFTransfer when pending amount is greater than 0.

```
if (_amount  > 0) {
user.amount = user.amount.sub(_amount);
pool.lpToken.safeTransfer(address(msg.sender), _amount);
}
```

NFT-14 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	farming/NFTLFarming.sol: 316	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `dev()`
- `changeNFTLPerBlock()`
- `updateTeamShare()`

Recommendation

Consider adding events for sensitive actions, and emit it in the function.

```
event SetDev(address indexed user, address indexed _devaddr);

function dev(address _devaddr) public onlyDev {
    devaddr = _devaddr;
    emit SetDev(msg.sender, _devaddr);
}
```

Alleviation

[CyberTime]: The team is addressed the issue and reflected the commit `abf8d521bd733db59a675a5b2aeff23eb86a757e`.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that 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 exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

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 and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

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

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.