CERTIK

Code Security Assessment

Defrost Finance III

Jan 5th, 2022

Table of Contents

Summary

This report has been prepared for Defrost Finance III to discover issues and vulnerabilities in the source code of the Defrost Finance III 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;
- 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.

Overview

Project Summary

Project Name	Defrost Finance III
Platform	Avalanche C-Chain
Language	Solidity
Codebase	https://github.com/DefrostFinance/defrost-finance-farm/tree/master/contracts/defrostBoostFarm
Commit	22c697ad31e2c4841530f82e2a875a4c355872f1 a58f1c5981874f46024bfa62b33953fd630e5b8a

Audit Summary

Delivery Date	Jan 05, 2022
Audit Methodology	Static Analysis, Manual Review
Key Components	

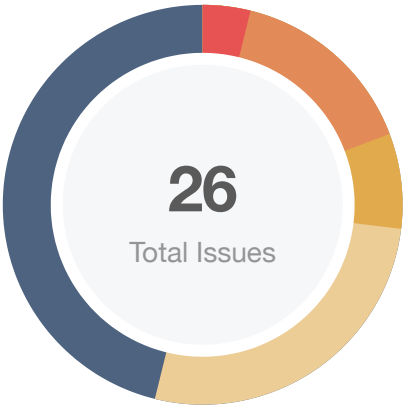
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	1	0	0	0	0	1
Major	4	0	0	1	3	0
Medium	2	0	0	1	1	0
Minor	7	0	0	1	3	3
Informational	12	0	0	5	3	4
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
TFD	boostTokenFarm.sol	ceb8bd0614c28beed9b256b2f5669f532880830f420a76ca43e6afb7890f3f2d
TFF	boostTokenFarmData.sol	6d52dffc2e3afdea00768f79dc86a0f52697dd7764a16b320c69d1a773eec35f
BFD	defrostBoostFarm.sol	bed4437318815a47a77b0a9b284c26ef9b35d970a0941f15a813b825f1720cf2
BFS	defrostBoostFarmStorage.sol	2447098df6e05aa49292357791642de73cb7ec0ed1c0c152a3ccc3c8952556db

Findings



Critical	**1** (3.85%)
Major	**4** (15.38%)
Medium	**2** (7.69%)
Minor	**7** (26.92%)
Informational	**12** (46.15%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
BFD-01	Lack of sanity checks on `totalSupply`	Logical Issue	Minor	Resolved
BFD-02	Lack of prevent the same `LP` token added in the `add` function	Logical Issue	Minor	Resolved
BFD-03	Lack of withdrawing `lpToken` tokens when update the `pool.extFarmInfo.extFarmAddr`	Logical Issue	Critical	Resolved
BFD-04	Fee Collector	**Centralization / Privilege**	**Major**	Acknowledged
BFD-05	Unknown Implementations	Volatile Code	Minor	Partially Resolved
BFD-06	Incompatibility With Deflationary Tokens	Volatile Code	Minor	Partially Resolved
BFD-07	The whitelist	Logical Issue	Informational	Resolved
BFD-08	Centralization Risk	**Centralization / Privilege**	**Major**	Partially Resolved
BFD-09	Missing Input Validation	Logical Issue	Informational	Partially Resolved
BFD-11	`extRewardPerBlock`	Coding Style	Informational	Resolved
BFD-12	Check whether an `uint256` type variable is less than zero	Logical Issue	Informational	Partially Resolved
BFD-13	Lack of updating the value of `extEnableClaim`	Logical Issue	Informational	Acknowledged
BFD-14	Potential loss in the `JoeToken` token	Logical Issue	Medium	Acknowledged

ID	Title	Category	Severity	Status
BFD-15	Missing Emit Events	Coding Style	Informational	Acknowledged
BFD-16	Potential Logic Flaw in `quitExtFarm`	Logical Issue	Informational	Acknowledged
BFD-17	Potential loss in `rewardToken`	Logical Issue	Informational	Acknowledged
BFS-01	Redundant Code Components	Volatile Code	Informational	Resolved
DFC-01	Unlocked compiler version	Language Specific	Informational	Resolved
DFC-02	Unknown Imported Source File	Logical Issue	Informational	Partially Resolved
TFD-01	Centralization Risk	**Centralization / Privilege**	**Major**	Partially Resolved
TFD-02	Lack of the check of the reward balance	Logical Issue	Medium	Partially Resolved
TFD-03	Lack of Input Validation	Volatile Code	Minor	Acknowledged
TFD-05	Potential Gas Waste in `rewardPerToken()`	Gas Optimization	Minor	Partially Resolved
TFD-06	Redundant Calculation for `reward`	Gas Optimization	Minor	Resolved
TFD-07	`duration` Update Issue	Logical Issue	Informational	Acknowledged
TFD-08	Centralization Risk	**Centralization / Privilege**	**Major**	Partially Resolved

BFD-01 | Lack of sanity checks on `totalSupply`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 297~299	Resolved

Description

Return zero if the value of `totalSupply` is zero in the `getExtFarmRewardRate` function.

Alleviation

The development team heeded our advice and resolved this issue in commit a58f1c5981874f46024bfa62b33953fd630e5b8a.

BFD-02 | Lack of prevent the same `LP` token added in the `add` function

Category	Severity	Location	Status
Logical Issue	Minor	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 297, 134	Resolved

Description

The same `lpToken` token should be prevented to be added into the pool.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The development team heeded our advice and resolved this issue in commit a58f1c5981874f46024bfa62b33953fd630e5b8a.

BFD-03 | Lack of withdrawing `lpToken` tokens when update the `pool.extFarmInfo.extFarmAddr`

Category	Severity	Location	Status
Logical Issue	Critical	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 370	Resolved

Description

Lack of withdrawing `lpToken` tokens when update the `pool.extFarmInfo.extFarmAddr` in the `setDoubleFarming` function if the pool has a non-zero `extFarmAddr` address. It is also needed to deposit the withdrawn `lpToken` tokens to the new pool.

Alleviation

The development team heeded our advice and resolved this issue in commit a58f1c5981874f46024bfa62b33953fd630e5b8a.

BFD-04 | Fee Collector

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 755	Acknowledged

Description

There is an amount of token to be transferred to the `teamRewardSc` account in the `mintUserRewardAndTeamReward` function.

Recommendation

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

[Defrost Finance Team]: `teamRewardSc` is a contract that has the logic to distribute tokens to different users.

BFD-05 | Unknown Implementations

Category	Severity	Location	Status
Volatile Code	Minor	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 8~43	Partially Resolved

Description

There are several unknown implementations in the contract `defrostBoostFarm.sol`.:

- `ITeamRewardSC`
- `IReleaseSC`
- `ITokenFarmSC`
- `IChef`

The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts.

Recommendation

We understand that the business logic of this protocol requires interaction with these functions. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Defrost Finance Team]:

- implementations for ITeamRewardSC is:https://github.com/DefrostFinance/defrost-finance-farm/blob/master/contracts/defrostTeamDistribute/defrostTeamDistribute.sol
- implementations for IReleaseSC is: https://github.com/DefrostFinance/defrost-finance-farm/blob/master/contracts/farmRelease/tokenRelease.sol
- implementations for ITokenFarmSC: https://github.com/DefrostFinance/defrost-finance-farm/blob/master/contracts/defrostBoostFarm/boostTokenFarm.sol
- implementations for IChef: This is third party contract, https://snowtrace.io/address/0x188bed1968b795d5c9022f6a0bb5931ac4c18f00#code

BFD-06 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Volatile Code	Minor	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 558, 514	Partially Resolved

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user stakes 100 deflationary tokens (with a 10% transaction fee) in a `DefrostFarm`, only 90 tokens actually arrived in the contract. However, the user can still withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction.

The `DefrostFarm` takes the pool token balance(the `currentSupply`) into account when calculating the users' reward. An attacker can repeat the process of deposit and withdraw to lower the token balance(`currentSupply`) in a deflationary token pool and cause the contract to increase the reward amount.

Reference: https://thoreum-finance.medium.com/what-exploit-happened-today-for-gocerberus-and-garuda-also-for-lokum-ybear-piggy-caramelswap-3943ee23a39f

Recommendation

We advise the client to regulate the set of pool tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[Defrost Finance Team]: We do not use deflationary token as lp.

BFD-07 | The whitelist

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 727	Resolved

Description

The user in the whitelist may be able to boost their mining APR potentially by up to `fixedWhitelistRatio`. It is not mentioned in the document, https://docs.defrost.finance/tokenomics/mining-boosting.

Alleviation

[Defrost Finance Team]: We added this to the doc.

BFD-08 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 1	Partially Resolved

Description

In the contract `DefrostFarm`, the role `owner` has the authority over the following function:

- `add(address _lpToken, uint256 _bonusStartTime, uint256 _bonusEndBlock, uint256 _rewardPerBlock, uint256 _totalMineReward, uint256 _duration, uint256 _secPerBlk )`
- `updatePoolInfo(uint256 _pid,uint256 _bonusEndBlock,uint256 _rewardPerBlock,uint256 _totalMineReward,uint256 _duration)`
- `distributeFinalExtReward(uint256 _pid, uint256 _amount)`
- `enableDoubleFarming(uint256 _pid, bool enable)`
- `setDoubleFarming(uint256 _pid,address extFarmAddr,uint256 _extPid)`
- `disableExtEnableClaim(uint256 _pid)`
- `emergencyWithdrawExtLp(uint256 _pid)`
- `quitDefrostFarm(address _to)`
- `quitExtFarm(address extFarmAddr, address _to)`
- `getBackLeftRewardToken(address _to)`
- `setDefrostAddress( address _rewardToken,address _h2o,address _teamRewardSc,address _releaseSc,address _tokenFarm,address _smelt)`
- `setFixedTeamRatio(uint256 _ratio)`
- `setFixedWhitelistPara(uint256 _incRatio,uint256 _whiteListfloorLimit)`
- `setWhiteList(address[] memory _user)`
- `setWhiteListMemberStatus(address _user,bool _status)`
- `setBoostFarmFactorPara(uint256 _BaseBoostTokenAmount,uint256 _BaseIncreaseRatio,uint256 _BoostTokenStepAmount,uint256 _RatioIncreaseStep,uint256 _MaxFactor)`

Any compromise to the `owner` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be

improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Defrost Finance Team]: The modifier is `onlyOrigin`, which is controlled by a multi-signature contract. It is not `onlyOwner`, the owner has no permission to call these function add,updatePoolInfo,distributeFinalExtReward...

For a multi-signature contract, please refer to: https://github.com/DefrostFinance/defrost-finance-farm/blob/master/contracts/modules/multiSignature.sol

BFD-09 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 812, 798~802	Partially Resolved

Description

Maybe the given input `_amount` is missing the check. A malicious user that deposits zero tokens or withdraws zero tokens can also get rewards by the statement `withdraw(_pid,0)`. Is that designed as expected?

Alleviation

[Defrost Finance Team]: Yes, it is designed as expected, allow user to deposit 0 to update farm status and allow user to get rewards.

BFD-11 | `extRewardPerBlock`

Category	Severity	Location	Status
Coding Style	Informational	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 290	Resolved

Description

```
uint256 extRewardPerBlock = chef.joePerSec();
```

According the above statement, the variable `extRewardPerBlock` should be renamed as `extRewardPerSec` to avoid misunderstanding.

Alleviation

The development team heeded our advice and resolved this issue in commit a58f1c5981874f46024bfa62b33953fd630e5b8a.

BFD-12 | Check whether an `uint256` type variable is less than zero

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 402 , 405	Partially Resolved

Description

```
if(pool.currentSupply <= 0) return 0;
```

There is no need to check whether an `uint256` type variable is less than zero.

Recommendation

Consider refactoring the codes as shown below:

```
if(pool.currentSupply == 0) return 0;
```

Alleviation

The development team heeded our advice and resolved this issue in commit a58f1c5981874f46024bfa62b33953fd630e5b8a.

BFD-13 | Lack of updating the value of `extEnableClaim`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 349	Acknowledged

Description

Lack of updating the value of `extEnableClaim` as false in the `else` branch of the `enableDoubleFarming` function. Is that designed as expected?

Alleviation

[Defrost Finance Team]: This function can update extEnableClaim to false.

```
function disableExtEnableClaim(uint256 _pid) public onlyOrigin
```

BFD-14 | Potential loss in the `JoeToken` token

Category	Severity	Location	Status
Logical Issue	Medium	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 443~450, 480~487	Acknowledged

Description

Potential loss in the `JoeToken` token if the value of `pool.extFarmInfo.extEnableClaim` is false.

Alleviation

[Defrost Finance Team]: `pool.extFarmInfo.extEnableClaim` will not be false if we open the double farming feature. Only it can be false if we do not open double farm feature or close double farm feature with careful consideration.

BFD-15 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 785~786, 712, 703, 695, 649~654, 689~693	Acknowledged

Description

The functions that affect the status of sensitive variables should be able to emit events as notifications to users.

- `setDefrostAddress( address _rewardToken,address _h2o,address _teamRewardSc,address _releaseSc,address _tokenFarm,address _smelt)`
- `setFixedTeamRatio(uint256 _ratio)`
- `setFixedWhitelistPara(uint256 _incRatio,uint256 _whiteListfloorLimit)`
- `setWhiteList(address[] memory _user)`
- `setWhiteListMemberStatus(address _user,bool _status)`
- `setBoostFarmFactorPara(uint256 _BaseBoostTokenAmount,uint256 _BaseIncreaseRatio,uint256 _BoostTokenStepAmount,uint256 _RatioIncreaseStep,uint256 _MaxFactor)`

Recommendation

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

Alleviation

[Defrost Finance Team]: Because the contract code is too big to deploy on chain because of gas limit. So we do not add event for them to reduce code size.

BFD-16 | Potential Logic Flaw in `quitExtFarm`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 626~641	Acknowledged

Description

The external farming may be still in active when calling `quitExtFarm`, after transferring tokens to the target account `_to`, `joeToken` may be not enough to withdraw for the user. Is that designed as expected?

Alleviation

[Defrost Finance Team]: `quitExtFarm` is just for an emergency function for exception, normally it is not called, it will not affect normal process.

BFD-17 | Potential loss in `rewardToken`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 604~606	Acknowledged

Description

There is a potential loss in the `rewardToken` if the value of `rewardBal` less than the value of `_amount` in the `safeRewardTransfer` function.

Alleviation

[Defrost Finance Team]: `safeRewardTransfer` is only called in emergency function: `quitDefrostFarm` and `getBackLeftRewardToken` function, loss will not be cared about if there are some emergency condition.

BFS-01 | Redundant Code Components

Category	Severity	Location	Status
Volatile Code	Informational	contracts/DefrostFinance/defrostBoostFarmStorage.sol (247c493): 99~103, 70~78, 9~45	Resolved

Description

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

Recommendation

We advise to remove the redundant statements for production environments.

Alleviation

The development team heeded our advice and resolved this issue in commit a58f1c5981874f46024bfa62b33953fd630e5b8a.

DFC-01 | Unlocked compiler version

Category	Severity	Location	Status
Language Specific	Informational	contracts/DefrostFinance/boostTokenFarmData.sol (247c493): 1 contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 1 contracts/DefrostFinance/boostTokenFarm.sol (247c493): 1 contracts/DefrostFinance/defrostBoostFarmStorage.sol (247c493) : 1	Resolved

Description

The contract 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 `v0.5.16` the contract should contain the following line:

```
pragma solidity 0.5.16;
```

Alleviation

The development team heeded our advice and resolved this issue in commit a58f1c5981874f46024bfa62b33953fd630e5b8a.

DFC-02 | Unknown Imported Source File

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DefrostFinance/defrostBoostFarm.sol (247c493): 3~6 contracts/DefrostFinance/boostTokenFarm.sol (247c493): 4~13	Partially Resolved

Description

The aforementioned imported source files are unknown.

Alleviation

[Defrost Finance Team]: The source file is in this directory: https://github.com/DefrostFinance/defrost-finance-farm/tree/master/contracts/modules.

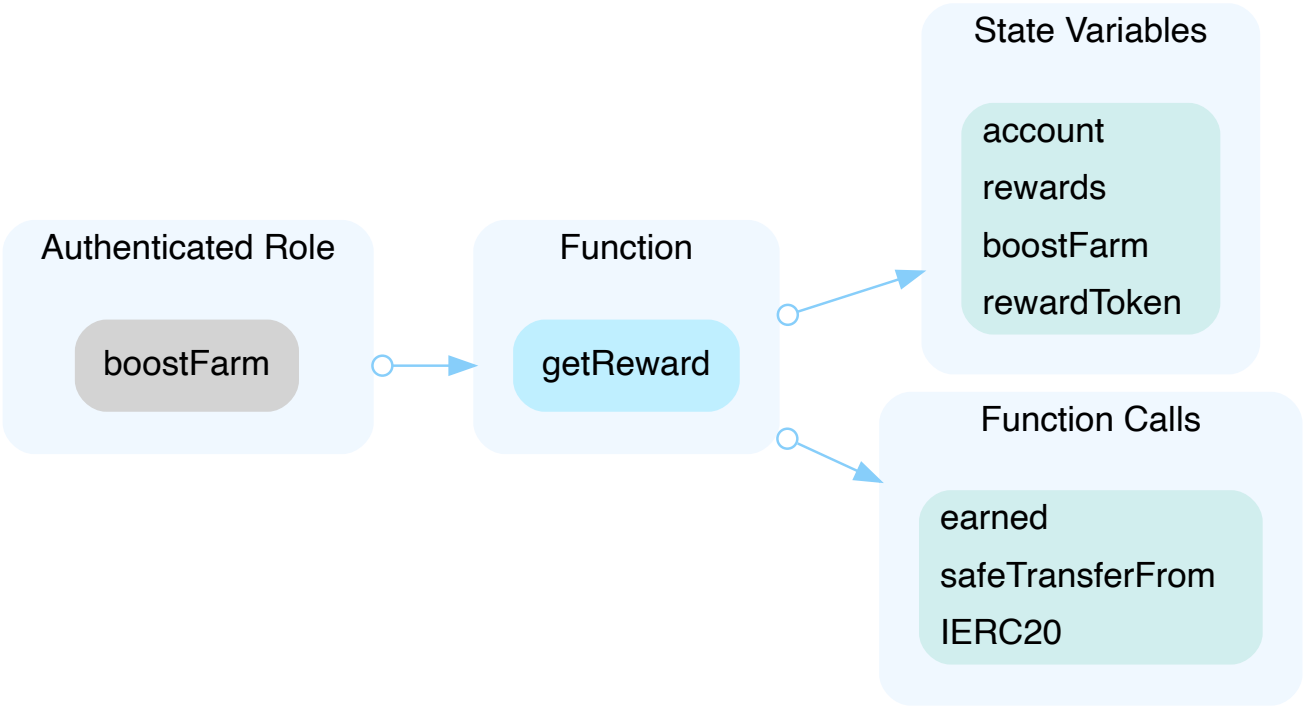
TFD-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/DefrostFinance/boostTokenFarm.sol (247c493): 108~115	Partially Resolved

Description

In the contract, `BoostTokenFarm`, the role, `boostFarm`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `boostFarm` may allow the hacker to take advantage of this and transfer reward tokens to the users.



Recommendation

We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Defrost Finance Team]: The privileged account is protected by multisig, the modifier `onlyOrigin` controlled this.

TFD-02 | Lack of the check of the reward balance

Category	Severity	Location	Status
Logical Issue	Medium	contracts/DefrostFinance/boostTokenFarm.sol (247c493): 54, 62	Partially Resolved

Description

Lack of check whether the `boostFarm` contract has enough balance and enough allowance to this contract when updating the amount of the reward and the period of the activity.

Alleviation

[Defrost Finance Team]: The allowance do not need to check because this is done by this in defrostBoostFarm.

```
IERC20(h2o).approve(address(tokenFarm),uint256(-1));
```

Balance do not need to check because updateReward just update status, not send token

TFD-03 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	contracts/DefrostFinance/boostTokenFarm.sol (247c493): 37~41	Acknowledged

Description

The given inputs of the construtor are missing zero address checks.

Recommendation

we advise the client to add proper checks to prevent unexpected errors.

Alleviation

[Defrost Finance Team]: We do not check it by contract itself in order to save code. Normally we will double check the constructor parameters when we deploy contract.

TFD-05 | Potential Gas Waste in `rewardPerToken()`

Category	Severity	Location	Status
Gas Optimization	Minor	contracts/DefrostFinance/boostTokenFarm.sol (247c493): 95~101	Partially Resolved

Description

Since the `rewardPerToken()` is a high frequency used function and performing an external call costs 700 gas, the gas waste should be considered. The `rewardPerToken()` has called `IERC20(boostFarm).totalSupply()` twice, which can be optimized to be called only once.

Recommendation

We advise the client to revisit the function and make an optimization that stores the return value of `IERC20(boostFarm).totalSupply()` in a local variable.

Alleviation

[Defrost Finance Team]: We intended to deploy the contract on avalanche, I think gas problem is not so important.

TFD-06 | Redundant Calculation for `reward`

Category	Severity	Location	Status
Gas Optimization	Minor	contracts/DefrostFinance/boostTokenFarm.sol (247c493): 108~109	Resolved

Description

The modifier `updateReward` has calculated the latest reward for the account, there is no need to call `earned(account)` to calculate the reward again.

Recommendation

We advise the client to revisit the function and simplify this calculation.

Alleviation

The development team heeded our advice and resolved this issue in commit a58f1c5981874f46024bfa62b33953fd630e5b8a.

TFD-07 | `duration` Update Issue

Category	Severity	Location	Status
Logical Issue	Informational	contracts/DefrostFinance/boostTokenFarm.sol (247c493): 58	Acknowledged

Description

Generally, the duration of this farming equals `periodFinish` minus `startTime`. Why does the `duration` is set separately? Is that designed as expected?

Alleviation

[Defrost Finance Team]: Yes, It is designed as expected. Duration is used to calculate reward speed. We normally input reward amount in one day to calculate the reward speed/seconds, so the duration will be one day time(86400 seconds)

TFD-08 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/DefrostFinance/boostTokenFarm.sol (247c493): 76, 62, 54, 49	Partially Resolved

Description

In the contract `BoostTokenFarm`, the role `owner` has the authority over the following function:

- `setPoolToken(address _boostFarm,address _rewardToken)`
- `setMineRate(uint256 _reward,uint256 _duration)`
- `setPeriodFinish(uint256 _startime,uint256 _endtime)`
- `getbackLeftMiningToken(address reciever)`

Any compromise to the `owner` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Defrost Finance Team]: The modifier is `onlyOrigin`, which is controlled by a multisignature contract. It is not `onlyOwner`, the owner has no permission to call these function add,updatePoolInfo,distributeFinalExtReward...

For multi-signature contract, please refer to: https://github.com/DefrostFinance/defrost-finance-farm/blob/master/contracts/modules/multiSignature.sol

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.

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

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.

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.