CERTIK

# Preliminary Comments

# FarmHero

Jun 7th, 2021

# Table of Contents

# Summary

This report has been prepared for FarmHero 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 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 | FarmHero |
| Description | MasterChef+Strategy |
| Platform | BSC |
| Language | Solidity |
| Codebase | |
| Commits | |

## Audit Summary

| | |
|---|---|
| Delivery Date | Jun 07, 2021 |
| Audit Methodology | Static Analysis, Manual Review |
| Key Components | |

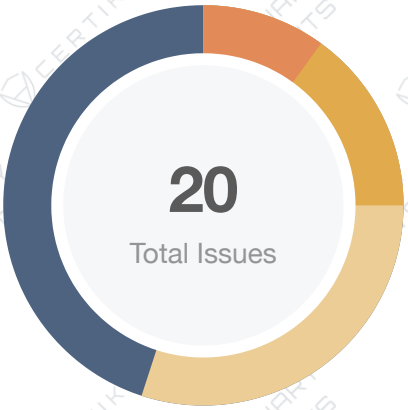
## Vulnerability Summary

| | |
|---|---|
| Total Issues | 20 |
| Critical | 0 |
| Major | 2 |
| Medium | 3 |
| Minor | 6 |
| Informational | 9 |
| Discussion | 0 |

## Audit Scope

| ID | file | SHA256 Checksum |
|---|---|---|
| HCK | Hero.sol | 266042f471bba8ec187fc832623a50a627586a679a3ac16a61f1537e9b67df77 |
| HFC | HeroFarm.sol | 689eaa2a6f6916919728bde4df20fd896a7d262989035cf9d4b5d548d52ddbdd |
| SXH | StratX2_HERO.sol | d7eaa976e1033a05de11f20b3916155f2b91c2593afeadd90492be92eca5cb48 |

# Findings




| | |
|---|---|
| Critical | 0 (0.00%) |
| Major | 2 (10.00%) |
| Medium | 3 (15.00%) |
| Minor | 6 (30.00%) |
| Informational | 9 (45.00%) |
| Discussion | 0 (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| **HCK-01** | Privileged ownership of HERO mint() | **Centralization / Privilege** | **Medium** | **Pending** |
| HCK-02 | Zero Address Validation | Volatile Code | Minor | Pending |
| HFC-01 | Unused local variables | Gas Optimization, Volatile Code | Informational | Pending |
| HFC-02 | Missing Checks for Reentrancy | Logical Issue | Medium | Pending |
| HFC-03 | Timestamp Dependence | Logical Issue | Minor | Pending |
| HFC-04 | Calls inside a loop | Logical Issue | Minor | Pending |
| HFC-05 | Zero Address Validation | Volatile Code | Minor | Pending |
| **HFC-06** | Privileged ownership of HEROMaxSupply | **Centralization / Privilege** | **Medium** | **Pending** |
| HFC-07 | Missing Return Value Handling | Logical Issue | Minor | Pending |
| HFC-08 | `isContract` is not safe | Logical Issue | Major | Pending |
| HFC-09 | Unused code in `withdrawNFT` | Gas Optimization | Informational | Pending |
| HFC-10 | Unused function parameters | Gas Optimization, Volatile Code | Informational | Pending |
| HFC-11 | Redundant Modifier `whenNotPaused` | Gas Optimization | Informational | Pending |
| HFC-12 | Unused State Variable | Gas Optimization | Informational | Pending |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| HFC-13 | add() Function Not Restricted | Volatile Code | Major | Pending |
| HFC-14 | Missing Emit Events | Volatile Code | Informational | Pending |
| HFC-15 | Functions Should Be Declared External | Gas Optimization | Informational | Pending |
| SXH-01 | Zero Address Validation | Volatile Code | Minor | Pending |
| SXH-02 | Unused function parameters | Gas Optimization, Volatile Code | Informational | Pending |
| SXH-03 | Unused State Variable | Gas Optimization | Informational | Pending |

## HCK-01 | Privileged ownership of HERO mint()

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization / Privilege | Medium | Hero.sol: 42 | Pending |

### Description

The Minters of HERO have permission to mint() any number of HERO token to any address without obtaining the consensus of the community.

### Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

# HCK-02 | Zero Address Validation

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Minor | Hero.sol: 34 | Pending |

## Description

Functions like below are missing zero address validation when critical addresses are initialized or set.

```
StratX2.setGov(address)
StratX2.setUniRouterAddress(address)
StratX2.setFomoAddress(address,address)
StratX2.setFomoAddress(address,address)
StratX2.setDevAddress(address)
HeroFarm.initialize(address,uint256,address[])
HeroFarm.setWithdrawFee(address,bool)
```

## Recommendation

Recommend applying `require` statements to all important functions make sure critical state variables are not set to `address(0)`.

## HFC-01 | Unused local variables

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization, Volatile Code | Informational | HeroFarm.sol: 2066, 2300, 2303, 2306, 2309, 2496 | Pending |

### Description

There are many unused local variables, like nftFarmingReward, teamReward, communityReward, ecosystemReward and wantLockedTotal.

### Recommendation

We recommend to remove all unused variables.

## HFC-02 | Missing Checks for Reentrancy

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Medium | HeroFarm.sol: 2066, 2093, 2268 | Pending |

### Description

Function add, set and updatePool have state updates or event emits after external calls and thus are vulnerable to reentrancy attack.

### Recommendation

We recommend applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for the aforementioned functions to prevent reentrancy attack.

# HFC-03 | Timestamp Dependence

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Minor | HeroFarm.sol: 2078, 2120, 2199, 2228, 2271 | Pending |

## Description

Dangerous usage of block.timestamp, block.timestamp can be manipulated by miners. Please understand the security risk level and trade-off of using `block.timestamp` or alias `now` as one of core factors in the contract.

## Recommendation

Correct use of 15-second rule to minimize the impact caused by timestamp variance

## HFC-04 | Calls inside a loop

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Minor | HeroFarm.sol: 2399, 2377, 2485, 2577 | Pending |

### Description

In function depositNFT(),withdrawNFT() and emergencyWithdrawNFT() have external calls inside a loop, calls inside a loop might lead to a denial-of-service attack.

### Recommendation

Favor pull over push strategy for external calls.

# HFC-05 | Zero Address Validation

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Minor | HeroFarm.sol: 2014, 2115 | Pending |

## Description

Functions like below are missing zero address validation when critical addresses are initialized or set.

```
StratX2.setGov(address)
StratX2.setUniRouterAddress(address)
StratX2.setFomoAddress(address,address)
StratX2.setFomoAddress(address,address)
StratX2.setDevAddress(address)
HeroFarm.initialize(address,uint256,address[])
HeroFarm.setWithdrawFee(address,bool)
```

## Recommendation

Recommend applying `require` statements to all important functions make sure critical state variables are not set to `address(0)`.

# HFC-06 | Privileged ownership of HEROMaxSupply

| Category | Severity | Location | Status |
|---|---|---|---|
| Centralization / Privilege | Medium | HeroFarm.sol: 2634~2636 | Pending |

## Description

Owner can change HEROMaxSupply to any number at any time.

```
2634    function setHEROMaxSupply(uint256 _supply) external onlyOwner {
2635        HEROMaxSupply = _supply;
2636    }
```

## Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations

# HFC-07 | Missing Return Value Handling

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Minor | HeroFarm.sol: 2611~2614, 2558 | Pending |

## Description

IStrategy.withdraw and IERC20.transfer are not void-returning functions. Ignoring the return value might cause some unexpected exception.

## Recommendation

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

## HFC-08 | `isContract` is not safe

| Category | Severity | Location | Status |
|---|---|---|---|
| Logical Issue | Major | HeroFarm.sol: 2638 | Pending |

### Description

If the function `isContract` returns true it means `account` is a initialized contract, but if it returns false, it doesn't mean `account` is not a contract. The true fact is that functions, `depositXX` and `withdrawXX`, can be called in a contract constructor.

```
function isContract(address account) internal view returns (bool) {
  // According to EIP-1052, 0x0 is the value returned for not-yet created accounts
  // and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470 is returned
  // for accounts without code, i.e. `keccak256('')`
  bytes32 codehash;
  bytes32 accountHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
  // solhint-disable-next-line no-inline-assembly
  assembly { codehash := extcodehash(account) }
  return (codehash != accountHash && codehash != 0x0);
}
```

### Recommendation

We advise the client to use EOA check, which is more safe and simple.

```
modifier isEOA() {
  require(tx.origin == msg.sender, "not EOA");
  _;
}
```

## HFC-09 | Unused code in `withdrawNFT`

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Informational | HeroFarm.sol: 2496 | Pending |

### Description

There are some unused code in function `withdrawNFT`.

```
        uint256 wantLockedTotal =
            IStrategy(poolInfo[_pid].strat).wantLockedTotal();
        uint256 sharesTotal = IStrategy(poolInfo[_pid].strat).sharesTotal();
```

### Recommendation

We recommend removing unused code for coding style and gas optimization.

# HFC-10 | Unused function parameters

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization, Volatile Code | Informational | HeroFarm.sol: 2649 | Pending |

## Description

Parameters of functions like onERC721Received(), deposit() and withdraw() are not used.

## Recommendation

We recommend to remove all unused function parameters.

## HFC-11 | Redundant Modifier `whenNotPaused`

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Informational | HeroFarm.sol: 2537 | Pending |

### Description

Function `withdrawAll()` is a wrapper function around `withdraw()`, `withdraw()` already has modifier `whenNotPaused`.

### Recommendation

We recommend to remove all redundant modifiers.

# HFC-12 | Unused State Variable

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Informational | HeroFarm.sol: 1955 | Pending |

## Description

`lastDepositBlock` in UserInfo struct is not used for logical function.

```
struct UserInfo {
    uint256 shares; // How many LP tokens the user has provided.
    uint256 rewardDebt; // Reward debt. See explanation below.
    uint64  gracePeriod; // timestamp of that users can receive the staked LP/Token without deducting the transcation fee
    uint64  lastDepositBlock;
}
```

## Recommendation

We recommend removing unused variable to save gas.

# HFC-13 | add() Function Not Restricted

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Major | HeroFarm.sol: 2066 | Pending |

## Description

The comment in line L2064, mentioned // XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.

The total amount of reward `HEROReward` in function `updatePool()` will be incorrectly calculated if the same `want` token is added into the pool more than once in function `add()`.

However, the code is not reflected in the comment behaviors as there isn't any valid restriction on preventing this issue.

The current implementation is relying on the trust of the owner to avoid repeatedly adding the same `want` token to the pool, as the function will only be called by the owner.

## Recommendation

Detect whether the given pool for addition is a duplicate of an existing pool. The pool addition is only successful when there is no duplicate. Using a mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

```
mapping(address => bool) public poolExistence;

modifier nonDuplicated(address _watnToken) {
        require(poolExistence[_watnToken] == false, "nonDuplicated: duplicated");
        _;
}

```

## HFC-14 | Missing Emit Events

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Informational | HeroFarm.sol: 2114 | Pending |

### Description

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

- `setWithdrawFee()`

### Recommendation

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

```
event SetWithdrawFee(address indexed _feeAddress, bool indexed _enable);

function setWithdrawFee(address _feeAddress, bool _enable) external onlyOwner{
        feeAddress = _feeAddress;
        withdrawFee = _enable;
        emit SetWithdrawFee(feeAddress, withdrawFee);
}
```

# HFC-15 | Functions Should Be Declared External

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Informational | HeroFarm.sol: 2339 | Pending |

## Description

Functions which are never called internally within the contract should have external visibility. For example, `add`, `set`, `deposit`, `depositNFT`, `withdrawNFT`, `emergencyWithdraw`, `emergencyWithdrawNFT` and `inCaseTokensGetStuck`.

## Recommendation

We recommend changing the visibility of the aforementioned functions to external.

## SXH-01 | Zero Address Validation

| Category | Severity | Location | Status |
|---|---|---|---|
| Volatile Code | Minor | StratX2_HERO.sol: 2537, 2551, 2560, 2561, 2570 | Pending |

### Description

Functions like below are missing zero address validation when critical addresses are initialized or set.

```
StratX2.setGov(address)
StratX2.setUniRouterAddress(address)
StratX2.setFomoAddress(address,address)
StratX2.setFomoAddress(address,address)
StratX2.setDevAddress(address)
HeroFarm.initialize(address,uint256,address[])
HeroFarm.setWithdrawFee(address,bool)
```

### Recommendation

Recommend applying `require` statements to all important functions make sure critical state variables are not set to `address(0)`.

## SXH-02 | Unused function parameters

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization, Volatile Code | Informational | StratX2_HERO.sol: 2252, 2310 | Pending |

### Description

Parameters of functions like onERC721Received(), deposit() and withdraw() are not used.

### Recommendation

We recommend to remove all unused function parameters.

## SXH-03 | Unused State Variable

| Category | Severity | Location | Status |
|---|---|---|---|
| Gas Optimization | Informational | StratX2_HERO.sol: 2199 | Pending |

### Description

`HEROAddress` is not used for logical function.

### Recommendation

We recommend removing unused variable to save gas.

# 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.

## 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 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.

CERTIK