CERTIK

Security Assessment

Heroes of Nft

Sept 27th, 2021

Table of Contents

About

Summary

This report has been prepared for Heroes of Nft to discover issues and vulnerabilities in the source code of the Heroes of Nft 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	Heroes of Nft
Platform	Avalanche
Language	Solidity
Codebase	https://github.com/heroesofnft/yield-farm/
Commit	487f51e989c34c08020519a0afe8c70a361e6c40 9e7b4bd404e4e0344313759f864c8de1e578c3df 21c945a6b390ab0e1383735f7e860de173e4b4e1

Audit Summary

Delivery Date	Sept 27, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

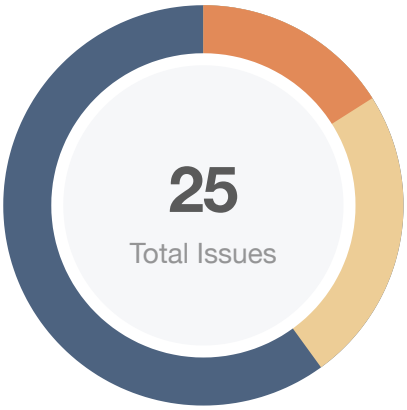
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	4	0	0	2	0	2
Medium	0	0	0	0	0	0
Minor	6	0	0	3	0	3
Informational	15	0	0	3	0	12
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
HTH	HonToken.sol	f155e040272b5168dd08b5c651b5b3b10711b2c74b606d17be2d2b5a352bedcb
MGH	MasterGamer.sol	00784fe9d8422ea6c6bad3ead2fe81a499511d2386bb572e82faae6ff29ae6e8

Findings



Critical	0 (0.00%)
Major	4 (16.00%)
Medium	0 (0.00%)
Minor	6 (24.00%)
Informational	15 (60.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
HTH-01	Unlocked Compiler Version	Language Specific	Informational	Resolved
HTH-02	Third Party Dependencies	Volatile Code	Minor	Acknowledged
HTH-03	Redundant Constant Variable	Volatile Code	Informational	Resolved
HTH-04	Function Visibility Optimization	Gas Optimization	Informational	Resolved
HTH-05	Missing Input Validation	Mathematical Operations, Logical Issue, Gas Optimization	Informational	Resolved
HTH-06	Redundant Modifier	Coding Style	Informational	Resolved
HTH-07	Unclear `deposit()` Logic	Logical Issue	Informational	Acknowledged
HTH-08	Delegation Not Moved Along With Transfer, Minting And Burning	Logical Issue	Major	Resolved
HTH-09	Contract Locks Ether	Logical Issue	Minor	Resolved
HTH-10	Susceptible to Signature Malleability	Volatile Code	Minor	Resolved
MGH-01	Unlocked Compiler Version	Language Specific	Informational	Resolved
MGH-02	Third Party Dependencies	Volatile Code	Minor	Acknowledged
MGH-03	Function Visibility Optimization	Gas Optimization	Informational	Resolved
MGH-04	Delegation Not Moved Along With Transfer, Minting And Burning	Logical Issue	Major	Resolved

ID	Title	Category	Severity	Status
MGH-05	Division Before Multiplication	Logical Issue, Mathematical Operations	Informational	Acknowledged
MGH-06	Comparison to A Boolean Constant	Volatile Code	Informational	Resolved
MGH-07	Comment Typo	Coding Style	Informational	Resolved
MGH-08	Missing Emit Events	Gas Optimization	Informational	Resolved
MGH-09	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	Resolved
MGH-10	Substitution Of `require` Calls With Modifier	Volatile Code	Informational	Resolved
MGH-11	Unknown Implementation Of `migrator.migrate()`	Logical Issue	Minor	Resolved
MGH-12	Unchecked Transfer	Logical Issue	Minor	Acknowledged
MGH-13	Update Logic Of `lastRewardTimestamp` Unclear	Mathematical Operations	Informational	Acknowledged
MGH-14	Centralization Risk With `Dev` Role	**Centralization / Privilege**	**Major**	Acknowledged
MGH-15	Centralization Risk With `owner` Role	**Centralization / Privilege**	**Major**	Acknowledged

HTH-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	HonToken.sol: 2	Resolved

Description

The contract has an 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 differing compiler version numbers. This can lead to 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.

Alleviation

[`Heroes Of NFT`]: Selected compiler version is now 0.8.3.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

HTH-02 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	HonToken.sol: 4~6, 37, 53	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party `NativeAssets` protocols and libraries. 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, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the logic of `HonToken` requires interaction with `NativeAssets` and many libraries. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[`Heroes Of NFT`]: NativeAssets library enables Hon contract to interact with Avalanche's core precompiled contracts which resides in :

- `0x0100000000000000000000000000000000000001` and
- `0x0100000000000000000000000000000000000002`.

The first one returns ANT (Avalanche Native Token) balance while the latter transfers ANT between x-chain and c-chain. The official Avalanche document states that these cross chain transfers are atomic.

HTH-03 | Redundant Constant Variable

Category	Severity	Location	Status
Volatile Code	Informational	HonToken.sol: 14, 18, 21	Resolved

Description

The `maxSupply` parameter is needed in the constructor for the deployment. However, this constant is never used except in a modifier which is never used in that contract.

Recommendation

We advise rewriting the contract to use this constant or removed it completely.

Alleviation

[`Heroes Of NFT`]: require statement is added for maxSupply check in `deposit()` function.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df and
21c945a6b390ab0e1383735f7e860de173e4b4e1

HTH-04 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	HonToken.sol: 36	Resolved

Description

The linked functions are declared as `public`, but are never called internally within the contract.

Recommendation

We advise that the functions' visibility specifiers are set to `external`, optimizing the gas cost of the function.

Alleviation

[`Heroes Of NFT`]:

- Modifier of the function deposit is changed to external.
- The function migrate is removed.

Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

HTH-05 | Missing Input Validation

Category	Severity	Location	Status
Mathematical Operations, Logical Issue, Gas Optimization	Informational	HonToken.sol: 47~55	Resolved

Description

The function `withdraw()` does not check if :

```
uint256 native_amount = amount / 1 gwei;
```

is greater than 0. If

```
NativeAssets.assetCall(msg.sender, _assetID, 0, "");
```

does not revert, the user will spend gas on a function with no usability.

Recommendation

We advise adding a check to avoid this kinds of error.

Alleviation

[`Heroes Of NFT`]: require statement is added for `native_amount` check in `withdraw()` function.

Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

HTH-06 | Redundant Modifier

Category	Severity	Location	Status
Coding Style	Informational	HonToken.sol: 26~29	Resolved

Description

The modifier `limitSupply` is never used.

Recommendation

We advise removing the modifier `limitSupply` or using it in the contract.
The logic of this modifier lets us thinks that it should not be possible to have a supply exceeding `maxSupply`. However, this could easily be done by using the `deposit()` function if the amount of `asset` exceeds the `maxSupply`.

Alleviation

[`Heroes Of NFT`]: Modifier `limitSupply` is deleted.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

HTH-07 | Unclear `deposit()` Logic

Category	Severity	Location	Status
Logical Issue	Informational	HonToken.sol: 36~44	Acknowledged

Description

The logic of the function `deposit()` implies that :

- more token than the `maxSupply` can be minted, indeed the function calculates the amount to mint without any consideration for the `maxSupply`.
- anyone can call this function because it is `public`. This function is callable if the value `(updatedBalance * 1 gwei)` is greater than the `totalSupply` of Token. In this case, it can be called only once.

```
39 uint256 depositAmount = (updatedBalance * 1 gwei) - totalSupply();
40 require(depositAmount > 0, "Deposit amount should be more than zero");
```

This means that if there are enough `NativeAssets` in the contract `HonToken` at a time **t**, a user can mint all the tokens available (which can be more than the `maxSupply`). Moreover, it will be necessary to wait until someone provides more `NativeAssets` to the `HonContract`, to call this function again.

This could lead to problems related to the `withdraw()` function: if Bob sends `nativeAssets` to the `HonToken` contract, then Alice call `deposit()` before Bob. She will get the corresponding amount of `HON`. She can now call `withdraw()` and get some `nativeAssets` Bob has sent.

Recommendation

This logic seems unusual. Since it is related to other tokens: `NativeAssets`, we cannot be sure of the purpose of this function.

We advise checking if the logic of the function works as intended and rewriting it if it is not the case.

Alleviation

[`Heroes Of NFT`]: `maxSupply` check is added to `deposit()` function.
Hon Token is intended to be an Avalanche Native Token (ANT) and created as a fixed supply asset in X-Chain of Avalanche network. Then it'll be wrapped as an ARC-20 (ERC-20 equivalent) to be used in smart contracts on C-Chain.

The official documents of Avalanche state that ‘Unlike ERC-20s, Avalanche Native Tokens (ANTs) are stored directly on the account that owns them.’. This means that for Bob to use his Hon-ANT tokens in C-Chain he has to transfer these to the Hon Token contract’s address. ‘The C-Chain keeps a mapping `[assetID -> balance]` in each account's storage to support ANTs.’. This means that Bob’s transferred Hon-ANT Tokens are tracked by Avalanche’s nodes. After this transaction, Bob simply calls the Hon Token contract’s deposit function just to update the contract’s states. The NativeAssets library returns the ANT amount stored in the contract's address and the contract's `totalSupply` and `balances` are updated accordingly. The reference Avalanche document link is provided.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

[`CertiK`]: In the commit 9e7b4bd404e4e0344313759f864c8de1e578c3df, there is no require statement added to the `deposit()` function.

HTH-08 | Delegation Not Moved Along With Transfer, Minting And Burning

Category	Severity	Location	Status
Logical Issue	Major	HonToken.sol: 42, 52	Resolved

Description

Since `HON` is a Token with Governance, the voting power of delegation should be changed when minting, burning and using `transfer()` and `transferFrom()` functions. Here the `_mint()`, `_burn()`, `transfer()` and `transferFrom()` functions are from `ERC20` protocol and do not invoke `_moveDelegates()`.

Recommendation

We advise checking that the logic of these functions works as intended and rewriting them if it's not the case.

Alleviation

[`Heroes Of NFT`]: Inherited transfer, minting and burning functions from Openzeppelin's ERC20 contract are overridden to support `moveDelegates()` function.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

HTH-09 | Contract Locks Ether

Category	Severity	Location	Status
Logical Issue	Minor	HonToken.sol: 272~276	Resolved

Description

The contract has payable fallback functions but without a withdrawal capacity. Hence every Ether sent to the contract will be lost.

Recommendation

We advise rewriting the functions to avoid loss of Ether or adding an ether withdrawal feature.

Alleviation

[`Heroes Of NFT`]: Withdraw feature is added for an account that is determined at the deploy time. This account is controlled by the team to retrieve the stucked Ether. The receive function cannot be altered for this purpose because it allows secure ANT transfer to the contract.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

HTH-10 | Susceptible to Signature Malleability

Category	Severity	Location	Status
Volatile Code	Minor	HonToken.sol: 145	Resolved

Description

The signature malleability is possible within the Elliptic Curve cryptographic system. An Elliptic Curve is symmetric on the X-axis, meaning two points can exist with the same `X` value. In the `r`, `s` and `v` representation this permits us to carefully adjust `s` to produce a second valid signature for the same `r`, thus breaking the assumption that a signature cannot be replayed in what is known as a replay-attack.

Recommendation

We advise to utilize a `recover()` function similar to that of the `ECDSA.sol` implementation of OpenZeppelin.

Alleviation

[`Heroes of NFT`]: Suggested 'recover' function is implemented then used as solidity's `ecrecover()` function.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

MGH-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	MasterGamer.sol: 2	Resolved

Description

The contract has an 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 differing compiler version numbers. This can lead to 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.

Alleviation

[`Heroes Of NFT`]: Selected compiler version is now 0.8.3.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

MGH-02 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	MasterGamer.sol: 4~9	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party `NativeAssets` protocols and libraries. 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, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the logic of `HonToken` requires interaction with `NativeAssets` and many libraries. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[`Heroes Of NFT`]: NativeAssets library enables Hon contract to interact with Avalanche's core precompiled contracts which resides in :

- `0x0100000000000000000000000000000000000001` and
- `0x0100000000000000000000000000000000000002`.

The first one returns ANT (Avalanche Native Token) balance while the latter transfers ANT between x-chain and c-chain. The official Avalanche document states that these cross chain transfers are atomic.

MGH-03 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	MasterGamer.sol: 189	Resolved

Description

The linked functions are declared as `public`, but are never called internally within the contract.

Recommendation

We advise that the functions' visibility specifiers are set to `external`, optimizing the gas cost of the function.

Alleviation

[`Heroes Of NFT`]:

- Modifier of the function deposit is changed to external.
- The function migrate is removed.

Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

MGH-04 | Delegation Not Moved Along With Transfer, Minting And Burning

Category	Severity	Location	Status
Logical Issue	Major	MasterGamer.sol: 267, 268, 305, 323	Resolved

Description

Since `HON` is a Token with Governance, the voting power of delegation should be changed when minting, burning and using `transfer()` and `transferFrom()` functions. Here the `_mint()`, `_burn()`, `transfer()` and `transferFrom()` functions are from `ERC20` protocol and do not invoke `_moveDelegates()`.

Recommendation

We advise checking that the logic of these functions works as intended and rewriting them if it's not the case.

Alleviation

[`Heroes Of NFT`]: Inherited transfer, minting and burning functions from Openzeppelin's ERC20 contract are overridden to support `moveDelegates()` function.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

MGH-05 | Division Before Multiplication

Category	Severity	Location	Status
Logical Issue, Mathematical Operations	Informational	MasterGamer.sol: 215~216, 264, 267~269	Acknowledged

Description

Mathematical operations in the linked functions perform divisions before multiplications. Performing multiplication before division can sometimes avoid loss of precision.

Recommendation

We advise the client to apply multiplications before divisions.

Alleviation

[`Heroes Of NFT`]: Added parentheses mathematical operations linked before `div()` operation to ensure the order of execution.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

[`Certik`]: It didn't change the fact that some divisions happen before multiplication. For example, in :

```
203  uint256 honReward =
(multiplier.mul(honPerPeriod).mul(pool.allocPoint)).div(totalAllocPoint);
204  accHonPerShare = accHonPerShare.add((honReward.mul(1e12)).div(lpSupply));
```

There is a division before a multiplication because of `honReward.mul(1e12)` which could be rewritten as :

```
(multiplier.mul(honPerPeriod).mul(pool.allocPoint)).div(totalAllocPoint).mul(1e12);
```

MGH-06 | Comparison to A Boolean Constant

Category	Severity	Location	Status
Volatile Code	Informational	MasterGamer.sol: 140	Resolved

Description

A boolean is compared to a boolean constant.

```
128  require(poolExists[address(_lpToken)] == false, "LP token has already been added");
```

Boolean constants can be used directly and do not need to be compared to true or false.

Recommendation

We advise removing the comparison to the boolean constant. For example:

```
require(!poolExists[address(_lpToken)], "LP token has already been added");
```

Alleviation

[Heroes Of NFT]: Suggested code has been applied.

Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

MGH-07 | Comment Typo

Category	Severity	Location	Status
Coding Style	Informational	MasterGamer.sol: 90	Resolved

Description

The linked comment statement contains a typo in its body.

Recommendation

We advise that the comment text is corrected.

Alleviation

[`Heroes Of NFT`]: Typo has been corrected.

Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

MGH-08 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	MasterGamer.sol: 184, 333, 340, 347	Resolved

Description

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

- `setMigrator()`
- `dev()`
- `treasury()`
- `fee()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function. For example for `dev()`:

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

function dev(address _devaddr) external {
  require(_devaddr != address(0), "dev address must not be address(0)");
  require(msg.sender == devaddr, "dev: you shall not pass !");
  devaddr = _devaddr;
}
```

Alleviation

[`Heroes Of NFT`]: Events `MigratorChanged`, `DevChanged`, `TreasuryChanged` and `FeeChanged` have been added. The affected functions are changed to emit these accordingly.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

[`Certik`]: Events `MigratorChanged` has not been added but `setMigrator()` has been removed.

MGH-09 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	MasterGamer.sol: 168, 189, 207, 240, 274, 299, 313	Resolved

Description

There is no sanity check to validate if a pool is existing.

Recommendation

We advise to adopt the following modifier :

```
modifier validatePoolByPid(uint256 _pid) {
    require (_pid < poolInfo . length , "Pool does not exist") ;
    _;
}
```

for the functions:

- `set()`;
- `migrate()`;
- `deposit()`;
- `withdraw()`;
- `emergencyWithdraw()`;
- `pendingHon()`;
- `updatePool()`.

Alleviation

[`Heroes Of NFT`]: Suggested modifier has been defined in the contract then added to the affected functions.

Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

MGH-10 | Substitution Of `require` Calls With Modifier

Category	Severity	Location	Status
Volatile Code	Informational	MasterGamer.sol: 335, 342, 349	Resolved

Description

The required statements on the linked lines can be converted into a modifier to avoid code duplication and increase the legibility of the code.

Recommendation

We advise using the following modifier instead of repeating the same require statements.

```
modifier onlyDev {
    require(msg.sender == devaddr, "dev: you shall not pass !");
    _;
}
```

Alleviation

[`Heroes Of NFT`]: Suggested modifier has been defined in the contract then added to the affected functions.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

MGH-11 | Unknown Implementation Of `migrator.migrate()`

Category	Severity	Location	Status
Logical Issue	Minor	MasterGamer.sol: 185, 195	Resolved

Description

The `setMigrator()` function can set migrator contract to any contract that is implemented from `IMigratorChef` interface by the owner. As result, invocation of `migrator.migrate()` in function migrate() may bring dangerous effects as it is unknown to the user. However, the project may lose the ability to upgrade and migrate if `setMigrator()` and `migrate()` are removed.

Recommendation

We advise planning to prevent abuse of the migrate functionality, and also to be completely transparent about this feature.

Alleviation

[`Heroes Of NFT`]: Intended use of migrator is, in case of necessary Hon Token update, it provides the ability to easily migrate a new token distribution contract while keeping everybody's pool share. But for the sake of keeping community trust and possible risks we are removing this ability from the contract.
Commit: 9e7b4bd404e4e0344313759f864c8de1e578c3df

MGH-12 | Unchecked Transfer

Category	Severity	Location	Status
Logical Issue	Minor	MasterGamer.sol: 323~330	Acknowledged

Description

The return value of `hon.transfer()` call is not checked. Several tokens do not revert in case of failure and return false. If one of these tokens is used as `hon`, this function will not revert. The functions `withdraw()`, `deposit()` and `update()` call the `hon.transfer()` function, in the above case it could lead to errors.

Moreover, the `transfer` function from the `HonToken` contract will revert if there are not enough tokens. This implies that the function `safeHonTransfer` will revert too. As a spill effect it could also make the functions `withdraw()`, `deposit()` and `update()` revert.

Recommendation

We advise checking if the logic of this function is as intended and rewriting it if it is not.

Alleviation

[`Heroes Of NFT`]: For the first matter, the audited Hon Token contract will be used as the `hon` object. Hon Token `transfer()` function reverts and this is intended in the code. The token distribution is decided to be done in five years. The last reward will be unsuccessful as the `transfer()` function reverts.

MGH-13 | Update Logic Of `lastRewardTimestamp` Unclear

Category	Severity	Location	Status
Mathematical Operations	Informational	MasterGamer.sol: 251~256	Acknowledged

Description

Because of how the function is implemented and how the `SafeMath` functions behave, there are two cases :

(1) if the distance between `block.timestamp` and `lastRewardTimestamp` is greater than the `rewardPeriodTime`, then `tmpLastRewardTimestamp` will be equal to

```
block.timestamp + rewardPeriodTime
```

(2) if `rewardPeriodTime` is greater or equal to the distance between `block.timestamp` and `lastRewardTimestamp`, then `tmpLastRewardTimestamp` will be equal to

```
lastRewardTimestamp + rewardPeriodTime
```

Since `rewardPeriodTime` is calculated with immutable parameters, it is possible that if wrong parameters are used for the deployment then it could have bad consequences on the `rewardPeriodTimes` calculation and therefore on the rewards calculation too.

Recommendation

We advise checking if the function behaves as intended and to changes it if it is not the case. If the logic is good, this finding has no resaon to be anymore.

Alleviation

[`Heroes Of NFT`]: The farm rewards are distributed in periods which are specified at deployment. In this way future reward times are set and cannot be changed after deployment. This ensures yield farming protocol to be trusted.
Let's say that farming starts at t0, after a period p the first reward will be distributed, that makes t1. When the reward is distributed at t1 lastRewardTimestamp is set to t1 and the users have to wait for t1 + p timestamp to get the next reward.

MGH-14 | Centralization Risk With `Dev` Role

Category	Severity	Location	Status
Centralization / Privilege	**Major**	MasterGamer.sol: 333~337, 340~344, 347~351	Acknowledged

Description

In the contract `MasterGamer`, the `Dev` account : `devaddr`, has the authority over the following function:

- `dev()`;
- `treasury()`;
- `fee()`;

Any compromise to the `Dev` account may allow the hacker to take advantage of this and :

- set an address he controls as the new `devaddr`, allowing this address to have the authority over the aforementioned functions and to receive all the tokens the `Dev` is supposed to have;
- set an address he controls as the new `treasuryaddr`, allowing the address to receive all the tokens the `Treasury` is supposed to have;
- set an address he controls as the new `feeaddr`, allowing the address to receive all the fee;

Recommendation

We advise the client to carefully manage the `Dev` 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

[`Heroes Of NFT`]: A multisignature wallet will be assigned as dev role. Initially this will be 4 private keys with 3 votes, after the expansion of the project it'll be increased to 9 private keys with 7 votes.

MGH-15 | Centralization Risk With `owner` Role

Category	Severity	Location	Status
Centralization / Privilege	**Major**	MasterGamer.sol: 134~138, 168~173, 184	Acknowledged

Description

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

- `transferOwnership` (in the `Ownable` library);
- `renounceOwnership` (in the `Ownable` library);
- `add()`;
- `set()`;
- `setMigrator()`;

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

- transfer the `owner` role to an address he controls;
- renounce the ownership and leave the contract without `owner`;
- create a new liquidity pool with the parameters he wants, or change the parameters of an already existing pool;
- migrate lp token to another lp contract, which could cause a devastating loss for the contract.

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;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[`Heroes Of NFT`]: As a recommended action the MasterGamer contract will be owned by a `Timelock` contract inherited from OpenZeppelin's `TimelockController`.

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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.