# Security Assessment

# MetaHero

Jul 12th, 2021

# Table of Contents

# Summary

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

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

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

# Overview

## Project Summary

| | |
|---|---|
| Project Name | MetaHero |
| Platform | BSC |
| Language | Solidity |
| Codebase | https://github.com/metahero-token/metahero-contracts/tree/7b843bd8965d01f36373dfc5dfc292fa4e495fb0 |
| Commit | 7b843bd8965d01f36373dfc5dfc292fa4e495fb0 |

## Audit Summary

| | |
|---|---|
| Delivery Date | Jul 12, 2021 |
| Audit Methodology | Static Analysis, Manual Review |
| Key Components | |

## Vulnerability Summary

| Vulnerability Level | Total | Pending | Partially Resolved | Resolved | Acknowledged | Declined |
|---|---|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 | 0 | 0 |
| Major | 2 | 0 | 0 | 0 | 2 | 0 |
| Medium | 0 | 0 | 0 | 0 | 0 | 0 |
| Minor | 6 | 0 | 0 | 1 | 5 | 0 |
| Informational | 2 | 0 | 0 | 1 | 1 | 0 |
| Discussion | 0 | 0 | 0 | 0 | 0 | 0 |

## Audit Scope

| ID | file | SHA256 Checksum |
|---|---|---|
| CCK | common/access/Controlled.sol | 5303a14ccdd33123b54d2c02cb73934cdab9d11cf1dd431cc3782597504c41a2 |
| LCK | common/access/Lockable.sol | 000e12ade538b0ba64fc3fea0b9168227c9a4360b0ad5e4f82a8b98d6c4d87d9 |
| OCK | common/access/Owned.sol | 650e7f258b479536a1da72f4ec32fe92b8328e3ce81afd1c980e73c3bc38fb07 |
| ERC | common/erc20/ERC20.sol | ccce5c3b585eb171b848be9192382ef99e12acf831dedf3415e1c68cd8cef5f3 |
| ICK | common/lifecycle/Initializable.sol | c676a1222f60736d327ad287a721f19ef1d47425f750aef71aedcd4deeab28ca |
| MLC | common/math/MathLib.sol | c731f2ad102e1d66681909eaf765ea0e155cabe9620b54bbfff43a293a9d1516 |
| SML | common/math/SafeMathLib.sol | 09675f7f288f08ee0a81e10c3ed16852072b1f6de1ae8655f5c56bb7dafddaa3 |
| HER | lpManager/HEROLPManager.sol | e38bf0beac6116cac316ab22a4159c9a632cc2307478ffb8d94bb7a6fc3d150b |
| HEO | lpManager/HEROLPManagerForUniswapV2.sol | 5d55e209d0ce4f3b996fb21c99e81af50fb8575c019ccd6d5a1b2e9e13d5e491 |
| HEP | presale/HEROPresale.sol | 51a66fbf9fc0847cc6fb9d46ced8c7639f67cb1cd4d41ba43340eac746cfe728 |
| HET | token/HEROToken.sol | e7927ff5a6745a239c995c9ded52cf4b1917b42796ae958ad7b26028118b0977 |

## Centralized Risks

The owner of contract `HEROToken` has the permission to:

1. exclude/include addresses from rewards/fees, hence holding accounts cannot get minted tokens/burn tokens. Holding account cannot transfer tokens in presale period,
2. mint any amount of tokens to any excluded accounts,
3. set `taxFee`, `liquidityFee` , `_maxTxAmount` and `presaleFinished`,
4. enable `swapAndLiquifyEnabled`

without obtaining the consensus of the community.

The owner of contract `HEROPresale` has the permission to:

1. change the presale period(deadline) via `updateDeadline()`,
2. exclude/include addresses from `whitelist`, hence only addresses in `whitelist` can buy HEROToken,
3. set `tokensAmountPerNative`, `maxPurchasePrice`

without obtaining the consensus of the community.

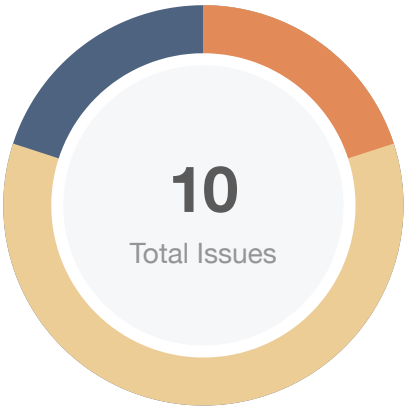
## Financial Models

Financial models of Metahero protocols need to be resilient to attacks. It needs to pass simulations and verifications to guarantee the security of the overall protocol.

As per the white paper, the HEROToken contract is a deflationary token, and 1% of each transaction amount is proportionally distributed among all holders as a passive reward. However, we are unable to find the reward distribution logic. The mechanism is just to move 1% of transaction amount from holding accounts to total rewards, and user's balance is keep shrinking. Thus, the reward fee is also acting like burning fee.

The scope of this audit is not including tokenomics, hence we strongly recommend the team to take serious considerations.

# Findings




| | |
|---|---|
| **Critical** | **0** (0.00%) |
| **Major** | **2** (20.00%) |
| **Medium** | **0** (0.00%) |
| **Minor** | **6** (60.00%) |
| **Informational** | **2** (20.00%) |
| **Discussion** | **0** (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| GLOBAL-01 | Missing emit events | Logical Issue | Minor | Resolved |
| **HEO-01** | Privileged ownership | **Centralization / Privilege** | **Minor** | **Acknowledged** |
| HEO-02 | Logic issue on reward fee | Logical Issue | Major | Acknowledged |
| HEO-03 | Missing logic in function `_transferFromExcludedAccount` | Logical Issue | Major | Acknowledged |
| HEP-01 | Issue in receive functions | Logical Issue | Minor | Acknowledged |
| **HEP-02** | Privileged ownership | **Centralization / Privilege** | **Minor** | **Acknowledged** |
| HER-01 | External dependencies risk | Control Flow | Informational | Acknowledged |
| HER-02 | Issue in receive functions | Logical Issue | Minor | Acknowledged |
| HER-03 | Code redundancy | Coding Style, Gas Optimization | Informational | Resolved |
| HER-04 | Potential sandwich attack | Logical Issue | Minor | Acknowledged |

## GLOBAL-01 | Missing emit events

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | Global | Resolved |

### Description

Function `initialize` in contract `HEROToken` affects the value of sensitive variables including `settings` and `lpManager`, should be able to emit events as notifications to customers.

Function `initialize` in contract `HEROLPManagerForUniswapV2` affects the value of sensitive variables including `uniswapRouter`, `wrappedNative` and `uniswapPair`, should be able to emit events as notifications to customers.

Function `initialize` in contract `HEROPresale` should also be able to emit events as notifications to customers.

### Recommendation

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

### Alleviation

The team heeded our advice addressed the issue and reflected in commit 63b63d16d52fa1da6f5bb3bc7a23ac4c283ea8a5.

## HEO-01 | Privileged ownership

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| **Centralization / Privilege** | **Minor** | contracts/metahero/token/HEROToken.sol | **Acknowledged** |

## Description

The owner of contract `HEROToken` has the permission to:

1. exclude/include addresses from rewards/fees, hence holding accounts cannot get minted tokens/burn tokens. Holding account cannot transfer tokens in presale period,
2. mint any amount of tokens to any excluded accounts,
3. set `taxFee`, `liquidityFee` , `_maxTxAmount` and `presaleFinished`,

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.

## Alleviation

The development team replied that

1. The presale is finished. The owner can still exclude fresh (without balance) accounts - this is required for future integrations.
2. The controller was turned off during the initialization process (there is no possibility of mining new tokens)
3. Only DAO can manipulate fees.

They are planning to migrate to DAO in the future - DAO will take ownership of the contract.

## HEO-02 | Logic issue on reward fee

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Major | contracts/metahero/token/HEROToken.sol | Acknowledged |

### Description

As per the white paper said, 1% of each transaction is proportionally distributed among all holders as a passive reward. However, there is no reward distribution logic.

```
    if (summary.totalRewards != 0) {
      uint256 totalHoldingWithRewards = summary.totalHolding.add(summary.totalRewards);

      senderAmount = senderAmount.mul(summary.totalHolding).div(totalHoldingWithRewards);
      recipientAmount =
recipientAmount.mul(summary.totalHolding).div(totalHoldingWithRewards);
      totalFee = totalFee.mul(summary.totalHolding).div(totalHoldingWithRewards);
    }
```

The above code is the only usage of the totalRewards, but it only reduces the transfer amount and fee (the reduced fee is matching the reduced transfer amount, so cannot be considered as rewards). The total rewards have never been distributed among all holders. Thus, the `rewardFee` is not actually a reward, it acts like a transaction fee.

### Recommendation

Consider to implement the reward distribution logic.

Or just remove the reward fee and update the white paper, because it seems not a helpful incentive to motivate people to buy Hero token. The reward fee is indicating that when user want to transfer his/her money to someone else, or receive money from others, he/she has to pay every one in the market a small portion of his money. This is not reasonable.

### Alleviation

**[MetaHero]**:

According to our whitepaper, 1% of each transaction is proportionally distributed among all holders as a passive reward.

**[CertiK Response]**:

We noticed the above-mentioned statement in the white paper, here we are asking the distribution logic. According to the code snippet in our finding, the transfer amount and fee will be multiplied by same rate calculated by reward fee. This is not a proper way of reward distribution. The reduced fee is only due to the reduced transfer amount, so both the sender and receiver do not get the reward. Reward fee keep accumulating, and never get distributed.

**[MetaHero]**:

1. We can not distribute rewards to all holder accounts each time rewards are updated - it would require a loop.

Instead of this all holders related variables are multiple by TH / (TH + TR) multiplier, where:

- TH = total balance of all holders
- TR = total accumulated rewards

https://github.com/metahero-io/metahero-contracts/blob/master/src/MetaheroToken.sol#L872
https://github.com/metahero-io/metahero-contracts/blob/master/src/MetaheroToken.sol#L875
https://github.com/metahero-io/metahero-contracts/blob/master/src/MetaheroToken.sol#L878

In the end, the holder can spend more than owns in the current account balance (proportionally to owned rewards)

2. Holder rewards are added to balance in the "balanceOf" method.
3. To be more transparent for holders, we created the "getBalanceSummary" method, which returns all balance components.

## HEO-03 | Missing logic in function `_transferFromExcludedAccount`

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Major | contracts/metahero/token/HEROToken.sol: 654 | Acknowledged |

## Description

As `_transferBetweenHolderAccounts` and `_transferToExcludedAccount`, there are some logics recalculate the transfer/receive amount for holding accounts based on the `totalHolding` and `totalRewards`. However, the function `_transferFromExcludedAccount` seems lack of these logics.

## Recommendation

Consider to add the missing logic, since in function `_transferFromExcludedAccount`, the recipient is a holding account.

## Alleviation

**[MetaHero]**:

That's correct because the excluded account is excluded from both fees and rewards. Please check the transfers scenario at:https://docs.google.com/spreadsheets/d/1YrzJ9mWM5QDxjdGkeYvy6baH2GgLIN_k_-wcCRFI8n0/edit?usp=sharing

**[CertiK Response]**:

`_transferFromExcludedAccount` is to transfer token from an excluded account to a holding account, so there is holding account involved. Thus, the logic mentioned in the finding is needed. Consider to add the similar code like the one in function `_transferToExcludedAccount`:

```
if (summary.totalRewards != 0) {
  uint256 totalHoldingWithRewards = summary.totalHolding.add(
    summary.totalRewards
  );

  senderAmount = senderAmount.mul(summary.totalHolding).div(
    totalHoldingWithRewards
  );
}
```

**[MetaHero]**:

This is a part of our model from google sheets. Please check the transfers scenario at:https://docs.google.com/spreadsheets/d/1YrzJ9mWM5QDxjdGkeYvy6baH2GgLIN_k_-wcCRFI8n0/edit?usp=sharing

## HEP-01 | Issue in receive functions

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | contracts/metahero/presale/HEROPresale.sol | Acknowledged |

### Description

In the Ethereum, `send/transfer/call` can be used for ETH transferrings. In the worst case, the receive function can only rely on 2300 gas being available (for example when `send` or `transfer` is used), leaving little room to perform other operations except basic logging. Therefore, the callback function of the current contract is not suitable for doing much. Contract `HEROPresale` and `HEROLPManagerForUniswapV2` have the same issue.

```
62  receive() external payable {
63          require(
64              block.timestamp < deadline, // solhint-disable-line not-rely-on-time
65              'HEROPresale#1'
66          );
67
68          require(whitelist[msg.sender], 'HEROPresale#2');
69
70          require(msg.value != 0, 'HEROPresale#3');
71
72          require(msg.value <= settings.maxPurchasePrice, 'HEROPresale#4');
73
74          uint256 tokensAmount = msg.value.mul(settings.tokensAmountPerNative);
75          // bnb amount * x = token amount
76
77          require(tokensAmount <= summary.totalTokens, 'HEROPresale#5');
78
79          whitelist[msg.sender] = false;
80
81          summary.totalAccounts = summary.totalAccounts.sub(1);// meimaide ren
82          summary.totalTokens = summary.totalTokens.sub(tokensAmount);
83
84          token.transfer(msg.sender, tokensAmount);
85
86          emit TokensPurchased(msg.sender, msg.value, tokensAmount);
87      }
```

The same parameter is used for Binance Smart Chain. Refer to: https://github.com/binance-chain/bsc/blob/46d185b4cfed54436f526b24c47b15ed58a5e1bb/params/protocol_params.go#L38

### Recommendation

Consider to test the gas consumption of above-mentioned codes.

Each opcode supported by the EVM has an associated gas cost. Pay attention the gas costs aren't arbitrary. Gas costs can and will change.

## Alleviation

For `Presale` contract, the development team replied that the `presale` contract has been removed from the audit scope.

For `MetaheroLPMForUniswapV2`(`HEROLPManagerForUniswapV2`) contract, the development team replied that the contract should be connected with HEO, not HER contract. This function is only used for testing purposes (converting BNB to WBNB).

## HEP-02 | Privileged ownership

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| **Centralization / Privilege** | **Minor** | contracts/metahero/presale/HEROPresale.sol | **Acknowledged** |

### Description

The owner of contract `HEROPresale` has the permission to:

1. changes the presale period(deadline) via `updateDeadline()`,
2. exclude/include addresses from `whitelist`, hence only addresses in `whitelist` can buy HEROToken,
3. set `tokensAmountPerNative`, `maxPurchasePrice`

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.

### Alleviation

The development team replied that the `presale` contract has been removed from the audit scope.

## HER-01 | External dependencies risk

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Control Flow | Informational | contracts/metahero/lpManager/HEROLPManagerForUniswapV2.sol | Acknowledged |

### Description

The contract is serving as the underlying entity to interact with third party PancakeSwap protocols. These external contracts are initialized by passing their contract addresses without emitting events, so they are unknown implementations for us. The scope of the audit would treat those external dependencies entities as black boxes and assume the functional correctness. In fact, any external dependencies might be compromised that led to assets lost or stolen.

### Recommendation

We understand that the business logic of the HERO protocol requires the interaction PancakeSwap protocol for adding liquidity to HERO-BNB pool and swap tokens. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate the side effects when unexpected activities are observed.

Make sure the external addresses are correct, and those contracts are credible, and the third-party implementations and the way these functions are called can meet the requirements.

### Alleviation

The development team understand the risks, but stated that there is no (gas-efficient) way to verify external calls to PancakeSwap contracts.

## HER-02 | Issue in receive functions

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | contracts/metahero/lpManager/HEROLPManagerForUniswapV2.sol | Acknowledged |

### Description

In the Ethereum, `send/transfer/call` can be used for ETH transferrings. In the worst case, the receive function can only rely on 2300 gas being available (for example when `send` or `transfer` is used), leaving little room to perform other operations except basic logging. Therefore, the callback function of the current contract is not suitable for doing much. Contract `HEROPresale` and `HEROLPManagerForUniswapV2` have the same issue.

```
62  receive() external payable {
63          require(
64              block.timestamp < deadline, // solhint-disable-line not-rely-on-time
65              'HEROPresale#1'
66          );
67
68          require(whitelist[msg.sender], 'HEROPresale#2');
69
70          require(msg.value != 0, 'HEROPresale#3');
71
72          require(msg.value <= settings.maxPurchasePrice, 'HEROPresale#4');
73
74          uint256 tokensAmount = msg.value.mul(settings.tokensAmountPerNative);
75          // bnb amount * x = token amount
76
77          require(tokensAmount <= summary.totalTokens, 'HEROPresale#5');
78
79          whitelist[msg.sender] = false;
80
81          summary.totalAccounts = summary.totalAccounts.sub(1);// meimaide ren
82          summary.totalTokens = summary.totalTokens.sub(tokensAmount);
83
84          token.transfer(msg.sender, tokensAmount);
85
86          emit TokensPurchased(msg.sender, msg.value, tokensAmount);
87      }
```

The same parameter is used for Binance Smart Chain. Refer to: https://github.com/binance-chain/bsc/blob/46d185b4cfed54436f526b24c47b15ed58a5e1bb/params/protocol_params.go#L38

### Recommendation

Consider to test the gas consumption of above-mentioned codes.

Each opcode supported by the EVM has an associated gas cost. Pay attention the gas costs aren't arbitrary. Gas costs can and will change.

## Alleviation

For `Presale` contract, the development team replied that the `presale` contract has been removed from the audit scope.

For `MetaheroLPMForUniswapV2`(`HEROLPManagerForUniswapV2`) contract, the development team replied that the contract should be connected with HEO, not HER contract. This function is only used for testing purposes (converting BNB to WBNB).

## HER-03 | Code redundancy

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Coding Style, Gas Optimization | Informational | contracts/metahero/lpManager/HEROLPManagerForUniswapV2.sol: 28, 239~246, 277~282 | Resolved |

### Description

Function in `uniswapRouter.addLiquidity` and `uniswapRouter.removeLiquidity` of Pancakeswap have already utilize the `sortTokens` function in the library, to correct the order of the input tokens. Thus, the state variable `correctPairOrder` and corresponding logic of sorting the tokens is redundant.

### Recommendation

Consider to remove redundant codes, and just pass correct tokens into functions `uniswapRouter.addLiquidity` and `uniswapRouter.removeLiquidity`.

### Alleviation

The team heeded our advice addressed the issue and reflected in commit 6b2e7eb6d401023d67a48df2ed73ccfff91dce8d.

# HER-04 | Potential sandwich attack

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | contracts/metahero/lpManager/HEROLPManagerForUniswapV2.sol | Acknowledged |

## Description

Potential sandwich attacks could happen if calling `uniswapV2Router.swapExactTokensForTokens` and `uniswapV2Router.addLiquidity` without setting restrictions on slippage.

For example, when we want to make a transaction of swapping 100 HERO for 1 BNB, an attacker could raise the price of BNB by adding HERO into the pool before the transaction so we might only get 0.1 BNB. After the transaction, the attacker would be able to withdraw more than he deposited because the total value of the pool increases by 0.9 BNB.

## Recommendation

We recommend using Oracle to get an estimation of prices and setting minimum amounts based on the prices when calling the aforementioned functions.

## Alleviation

The development team understand the risks, but stated that there is no (gas-efficient) way to verify external calls to PancakeSwap contracts, and calling oracle will have an impact on gas usage.

# 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.

### Control Flow

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

### 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 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.