MythX

REPORT 607941A42A50CA00110D75AC

Created	Fri Apr 16 2021 07:49:56 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	admin@axedefi.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
fb58156f-fbfd-4b6b-a19d-71679888baf1	contracts/MasterChef.sol	47

Analysis fb58156f-fbfd-4b6b-a19d-71679888baf1 MythX

Started	Fri Apr 16 2021 07:49:57 GMT+0000 (Coordinated Universal Time)
Finished	Fri Apr 16 2021 08:05:38 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Remythx
Main Source File	Contracts/MasterChef.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	24	23

ISSUES

MEDIUM Function could be marked as external.

SWC-000 The function definition of "add" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/math/SafeMath.sol

Locations

```
110 *
111 * Counterpart to Solidity's `/` operator. Note: this function uses a
112 * `revert` opcode (which leaves remaining gas untouched) while Solidity
113 * uses an invalid opcode to revert (consuming all remaining gas).
114 *
115 * Requirements:
116 *
117 * - The divisor cannot be zero.
118 */
119 function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
120 require(b > 0, errorMessage);
121 uint256 c = a / b;
122 // assert(a == b * c + a % b); // There is no case in which this doesn't hold
123
124 return c;
125 }
126
127 /**
128 * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
129 * Reverts when dividing by zero.
130 *
131 * Counterpart to Solidity's `%` operator. This function uses a `revert`
132 * opcode (which leaves remaining gas untouched) while Solidity uses an
133 * invalid opcode to revert (consuming all remaining gas).
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "set" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/math/SafeMath.sol

Locations

```
130 *
131 * Counterpart to Solidity's `%` operator. This function uses a `revert`
132 * opcode (which leaves remaining gas untouched) while Solidity uses an
133 * invalid opcode to revert (consuming all remaining gas).
134 *
135 * Requirements:
136 *
137 * - The divisor cannot be zero.
138 */
139 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
140 return mod(a, b, "SafeMath: modulo by zero");
141 }
142
143 /**
144 * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
145 * Reverts with custom message when dividing by zero.
146 *
147 * Counterpart to Solidity's `%` operator. This function uses a `revert`
148 * opcode (which leaves remaining gas untouched) while Solidity uses an
149 * invalid opcode to revert (consuming all remaining gas).
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "mint" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/libs/IBEP20.sol

Locations

```
9 /**
10 * @dev Returns the token decimals.
11 */
12 function decimals() external view returns (uint8);
13
14 /**
15 * @dev Returns the token symbol.
16 */
17 function symbol() external view returns (string memory);
18
19 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "renounceOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/AxeToken.sol

Locations

```
36
37  /// @notice The EIP-712 typehash for the delegation struct used by the contract
38  bytes32 public constant DELEGATION_TYPEHASH = keccak256("Delegation(address delegatee,uint256 nonce,uint256 expiry)");
39
40  /// @notice A record of states for signing / validating signatures
41  mapping (address => uint) public nonces;
42
43  /// @notice An event thats emitted when an account changes its delegate
44  event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "transferOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/AxeToken.sol

Locations

```
42
43  /// @notice An event thats emitted when an account changes its delegate
44  event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);
45
46  /// @notice An event thats emitted when a delegate account's vote balance changes
47  event DelegateVotesChanged(address indexed delegate, uint previousBalance, uint newBalance);
48
49  /**
50  * @notice Delegate votes from `msg.sender` to `delegatee`
51  * @param delegator The address to get delegatee for
52  */
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "symbol" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
62  * @dev Performs a Solidity function call using a low level `call`. A
63  * plain`call` is an unsafe replacement for a function call: use this
64  * function instead.
65  *
66  * If `target` reverts with a revert reason, it is bubbled up by this
67  * function (like regular Solidity function calls).
68  *
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "decimals" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
67 * function (like regular Solidity function calls).
68 *
69 * Returns the raw returned data. To convert to the expected return value,
70 * use https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlight=abi.decode#abi-encoding-and-decoding-functions[`abi.decode`].
71 *
72 * Requirements:
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "totalSupply" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
68 *
69 * Returns the raw returned data. To convert to the expected return value,
70 * use https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlight=abi.decode#abi-encoding-and-decoding-functions[`abi.decode`].
71 *
72 * Requirements:
73 *
74 * - `target` must be a contract.
75 * - calling `target` with `data` must not revert.
76 *
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
82
83 /**
84 * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with
85 * `errorMessage` as a fallback revert reason when `target` reverts.
86 *
87 * _Available since v3.1._
88 */
89 function functionCall(address target, bytes memory data, string memory errorMessage) internal returns (bytes memory) {
90 return functionCallWithValue(target, data, 0, errorMessage);
91 }
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "allowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
87  * _Available since v3.1._
88  */
89  function functionCall(address target, bytes memory data, string memory errorMessage) internal returns (bytes memory) {
90  return functionCallWithValue(target, data, 0, errorMessage);
91  }
92
93  /**
94  * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
95  * but also transferring `value` wei to `target`.
96  *
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "approve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
94  * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
95  * but also transferring `value` wei to `target`.
96  *
97  * Requirements:
98  *
99  * - the calling contract must have an ETH balance of at least `value`.
100 * - the called Solidity function must be `payable`.
101 *
102 * _Available since v3.1._
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
108 /**
109 * @dev Same as {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`], but
110 * with `errorMessage` as a fallback revert reason when `target` reverts.
111 *
112 * _Available since v3.1._
113 */
114 function functionCallWithValue(address target, bytes memory data, uint256 value, string memory errorMessage) internal returns (bytes memory) {
115 require(address(this).balance >= value, "Address: insufficient balance for call");
116 require(isContract(target), "Address: call to non-contract");
117
118 // solhint-disable-next-line avoid-low-level-calls
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "increaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
125 * but performing a static call.
126 *
127 * _Available since v3.3._
128 */
129 function functionStaticCall(address target, bytes memory data) internal view returns (bytes memory) {
130 return functionStaticCall(target, data, "Address: low-level static call failed");
131 }
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "decreaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
141
142 // solhint-disable-next-line avoid-low-level-calls
143 (bool success, bytes memory returndata) = target.staticcall(data);
144 return _verifyCallResult(success, returndata, errorMessage);
145 }
146
147 function _verifyCallResult(bool success, bytes memory returndata, string memory errorMessage) private pure returns(bytes memory) {
148 if (success) {
149 return returndata;
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

@openzeppelin/contracts/utils/Address.sol

Locations

```
151 // Look for revert reason and bubble it up if present
152 if (returndata.length > 0) {
153 // The easiest way to bubble the revert reason is using memory via assembly
154
155 // solhint-disable-next-line no-inline-assembly
156 assembly {
157 let returndata_size := mload(returndata)
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

@openzeppelin/contracts/math/SafeMath.sol

Locations

```
118 */
119 function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
120 require(b > 0, errorMessage);
121 uint256 c = a / b;
122 // assert(a == b * c + a % b); // There is no case in which this doesn't hold
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

@openzeppelin/contracts/math/SafeMath.sol

Locations

```
119 function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
120 require(b > 0, errorMessage);
121 uint256 c = a / b;
122 // assert(a == b * c + a % b); // There is no case in which this doesn't hold
```