MythX

REPORT 60688534DD06F7001141532C

Created	Sat Apr 03 2021 15:09:40 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	admin@kobito.finance

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
b36392f5-a5c4-456d-a9fa-2e1361d8c50d	contracts/MasterChef.sol	57

Analysis b36392f5-a5c4-456d-a9fa-2e1361d8c50d MythX

Started	Sat Apr 03 2021 15:09:49 GMT+0000 (Coordinated Universal Time)
Finished	Sat Apr 03 2021 15:25:46 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Remythx
Main Source File	Contracts/MasterChef.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	26	31

ISSUES

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/SafeBEP20.sol
Locations

```
11 * @dev Wrappers around BEP20 operations that throw on failure (when the token
12 * contract returns false). Tokens that return no value (and instead revert or
13 * throw on failure) are also supported, non-reverting calls are assumed to be
14 * successful.
15 * To use this library you can add a `using SafeBEP20 for IBEP20;` statement to your contract,
16 * which allows you to call the safe operations as `token.safeTransfer(...)`, etc.
17 */
```

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/MasterChef.sol
Locations

```
42 // Info of each pool.
43 struct PoolInfo {
44 IBEP20 lpToken; // Address of LP token contract.
45 uint256 allocPoint; // How many allocation points assigned to this pool. KBTs to distribute per block.
46 uint256 lastRewardBlock; // Last block number that KBTs distribution occurs.
47 uint256 accKobitoPerShare; // Accumulated KBTs per share, times 1e12. See below.
```

MEDIUM

SWC-000

Function could be marked as external.

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/MasterChef.sol

Locations

```
45  uint256 allocPoint; // How many allocation points assigned to this pool. KBTs to distribute per block.
46  uint256 lastRewardBlock; // Last block number that KBTs distribution occurs.
47  uint256 accKobitoPerShare; // Accumulated KBTs per share, times 1e12. See below.
48  uint16 depositFeeBP; // Deposit fee in basis points
49  }
50
51  // The KBT TOKEN!
52  KobitoToken public kobito;
53  // Dev address.
54  address public devAddress;
55  // Deposit Fee address
56  address public feeAddress;
57  // KBT tokens created per block.
```

MEDIUM

SWC-000

Function could be marked as external.

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/math/SafeMath.sol

Locations

```
79  * Counterpart to Solidity's `+` operator.
80  *
81  * Requirements:
82  *
83  * - Addition cannot overflow.
84  */
85  function add(uint256 a, uint256 b) internal pure returns (uint256) {
86  uint256 c = a + b;
87  require(c >= a, "SafeMath: addition overflow");
```

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/math/SafeMath.sol

Locations

```
84  */
85  function add(uint256 a, uint256 b) internal pure returns (uint256) {
86  uint256 c = a + b;
87  require(c >= a, "SafeMath: addition overflow");
88  return c;
89  }
90
91  /**
92  * @dev Returns the subtraction of two unsigned integers, reverting on
93  * overflow (when the result is negative).
94  *
```

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/math/SafeMath.sol

Locations

```
90
91  /**
92  * @dev Returns the subtraction of two unsigned integers, reverting on
93  * overflow (when the result is negative).
94  *
95  * Counterpart to Solidity's `-` operator.
96  *
97  * Requirements:
```

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/math/SafeMath.sol

Locations

```
108 * overflow.
109 *
110 * Counterpart to Solidity's `*` operator.
111 *
112 * Requirements:
113 *
114 * - Multiplication cannot overflow.
115 */
116 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
117 if (a == 0) return 0;
118 uint256 c = a * b;
```

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/math/SafeMath.sol

Locations

```
115 */
116 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
117 if (a == 0) return 0;
118 uint256 c = a * b;
119 require(c / a == b, "SafeMath: multiplication overflow");
120 return c;
121 }
122
123 /**
124 * @dev Returns the integer division of two unsigned integers, reverting on
125 * division by zero. The result is rounded towards zero.
126 *
```

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/math/SafeMath.sol

Locations

```
124 * @dev Returns the integer division of two unsigned integers, reverting on
125 * division by zero. The result is rounded towards zero.
126 *
127 * Counterpart to Solidity's `/` operator. Note: this function uses a
128 * `revert` opcode (which leaves remaining gas untouched) while Solidity
129 * uses an invalid opcode to revert (consuming all remaining gas).
130 *
131 * Requirements:
```

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/math/SafeMath.sol

Locations

```
142 * reverting when dividing by zero.
143 *
144 * Counterpart to Solidity's `%` operator. This function uses a `revert`
145 * opcode (which leaves remaining gas untouched) while Solidity uses an
146 * invalid opcode to revert (consuming all remaining gas).
147 *
148 * Requirements:
149 *
150 * - The divisor cannot be zero.
151 */
152 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
153 require(b > 0, "SafeMath: modulo by zero");
154 return a % b;
155 }
```

MEDIUM

SWC-000

Function could be marked as external.

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/math/SafeMath.sol

Locations

```
162 * message unnecessarily. For custom revert reasons use {trySub}.
163 *
164 * Counterpart to Solidity's `-` operator.
165 *
166 * Requirements:
167 *
168 * - Subtraction cannot overflow.
169 */
170 function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
171 require(b <= a, errorMessage);
172 return a - b;
```

MEDIUM

SWC-000

Function could be marked as external.

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/math/SafeMath.sol

Locations

```
180 * message unnecessarily. For custom revert reasons use {tryDiv}.
181 *
182 * Counterpart to Solidity's `/` operator. Note: this function uses a
183 * `revert` opcode (which leaves remaining gas untouched) while Solidity
184 * uses an invalid opcode to revert (consuming all remaining gas).
185 *
186 * Requirements:
187 *
188 * - The divisor cannot be zero.
189 */
190 function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
191 require(b > 0, errorMessage);
192 return a / b;
```

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

@openzeppelin/contracts/math/SafeMath.sol

Locations

```
194
195 /**
196 * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
197 * reverting with custom message when dividing by zero.
198 *
199 * CAUTION: This function is deprecated because it requires allocating memory for the error
200 * message unnecessarily. For custom revert reasons use {tryMod}.
201 *
```