MythX

REPORT 652621710CD13D00197D7936

Created	Wed Oct 11 2023 04:15:45 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	637bc5fd8288ab39b9a0547b

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
04acc8c0-a6fe-4cf9-b226-efed44210dea	contracts/Beatsminer.sol	2

Analysis 04acc8c0-a6fe-4cf9-b226-efed44210dea MythX

Started	Wed Oct 11 2023 04:15:50 GMT+0000 (Coordinated Universal Time)
Finished	Wed Oct 11 2023 04:15:55 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Contracts/Beatsminer.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	2

ISSUES

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

@openzeppelin/contracts/utils/math/SafeMath.sol

Locations

```
117 * Requirements:
118 *
119 * - Multiplication cannot overflow.
120 */
121 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
```

UNKNOWN Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

@openzeppelin/contracts/utils/math/SafeMath.sol

Locations

```
119 * - Multiplication cannot overflow.
120 */
121 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
122 return a * b;
123 }
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

@openzeppelin/contracts/utils/math/SafeMath.sol

Locations

```
117 * Requirements:
118 *
119 * - Multiplication cannot overflow.
120 */
121 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

@openzeppelin/contracts/utils/math/SafeMath.sol

Locations

```
119 * - Multiplication cannot overflow.
120 */
121 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
122 return a * b;
123 }
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.14"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

@openzeppelin/contracts/access/Ownable.sol

Locations

```
59 * thereby removing any functionality that is only available to the owner.
60 */
61 function renounceOwnership() public virtual onlyOwner {
62 _transferOwnership(address(0));
63 }
```

LOW

SWC-108

State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "TOKEN" is internal. Other possible visibility settings are public and private.

Source file

@openzeppelin/contracts/access/Ownable.sol

Locations

```
39
40  /**
41  * @dev Returns the address of the current owner.
42  */
43  function owner() public view virtual returns (address) {
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

@openzeppelin/contracts/utils/math/SafeMath.sol

Locations

```
124
125 /**
126 * @dev Returns the integer division of two unsigned integers, reverting on
127 * division by zero. The result is rounded towards zero.
128 *
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

@openzeppelin/contracts/utils/math/SafeMath.sol

Locations

```
125 /**
126 * @dev Returns the integer division of two unsigned integers, reverting on
127 * division by zero. The result is rounded towards zero.
128 *
129 * Counterpart to Solidity's `/` operator.
130 *
131 * Requirements:
```

UNKNOWN

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

SWC-110

Source file

@openzeppelin/contracts/utils/math/SafeMath.sol

Locations

```
166 * - Subtraction cannot overflow.
167 */
168 function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
169 unchecked {
170 require(b <= a, errorMessage);
```