MythX

Analysis 2443ef85-e7ed-4afa-b826-bd5388fd727a

MythX

Started	Mon Jul 12 2021 18:49:25 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 12 2021 18:51:38 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Brownie-1.14.6
Main Source File	Contracts/HeartToken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

ISSUES

UNKNOWN Arithmetic operation "-" discovered

SWC-101

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

Source file

contracts/HeartToken.sol

Locations

```
return 0;
} else {
return ids[ids.length - 1];
}
}
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

SWC-101

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

Source file

contracts/HeartToken.sol

Locations

```
return 0;
} else {
return ids[ids.length - 1];
}
}
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/structs/EnumerableSet.sol

Locations

```
// This modifies the order of the array, as noted in {at}.

uint256 toDeleteIndex = valueIndex - 1;
uint256 lastIndex = set._values.length - 1;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/structs/EnumerableSet.sol

Locations

```

uint256 toDeleteIndex = valueIndex - 1;
uint256 lastIndex = set._values.length - 1;

if (lastIndex != toDeleteIndex) {
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/structs/EnumerableSet.sol

Locations

```
// This modifies the order of the array, as noted in {at}.

uint256 toDeleteIndex = valueIndex - 1;
uint256 lastIndex = set._values.length - 1;
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/structs/EnumerableSet.sol

Locations

```

uint256 toDeleteIndex = valueIndex - 1;
uint256 lastIndex = set._values.length - 1;

if (lastIndex != toDeleteIndex) {
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Arrays.sol

Locations

```
high = mid;
} else {
low = mid + 1;
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Arrays.sol

Locations

```

// At this point `low` is the exclusive upper bound. We will return the inclusive upper bound.
if (low > 0 && array[low - 1] == element) {
return low - 1;
} else {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Arrays.sol

Locations

```
// At this point `low` is the exclusive upper bound. We will return the inclusive upper bound.
if (low > 0 && array[low - 1] == element) {
return low - 1;
} else {
return low;
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Arrays.sol

Locations

```

// At this point `low` is the exclusive upper bound. We will return the inclusive upper bound.
if (low > 0 && array[low - 1] == element) {
return low - 1;
} else {
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Arrays.sol

Locations

```
// At this point `low` is the exclusive upper bound. We will return the inclusive upper bound.
if (low > 0 && array[low - 1] == element) {
return low - 1;
} else {
return low;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol

Locations

```
function average(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b) / 2 can overflow, so we distribute.
return (a / 2) + (b / 2) + (((a % 2) + (b % 2)) / 2);
}
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol

Locations

```
function average(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b) / 2 can overflow, so we distribute.
return (a / 2) + (b / 2) + (((a % 2) + (b % 2)) / 2);
}
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol

Locations

```
function average(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b) / 2 can overflow, so we distribute.
return (a / 2) + (b / 2) + (((a % 2) + (b % 2)) / 2);
}
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol

Locations

```
function average(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b) / 2 can overflow, so we distribute.
return (a / 2) + (b / 2) + (((a % 2) + (b % 2)) / 2);
}
```

UNKNOWN SWC-101

Arithmetic operation "/" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol

Locations

```
function average(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b) / 2 can overflow, so we distribute.
return (a / 2) + (b / 2) + (((a % 2) + (b % 2)) / 2);
}
```

UNKNOWN SWC-101

Arithmetic operation "+" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol

Locations

```
function average(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b) / 2 can overflow, so we distribute.
return (a / 2) + (b / 2) + (((a % 2) + (b % 2)) / 2);
}
```

UNKNOWN SWC-101

Arithmetic operation "%" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol

Locations

```
function average(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b) / 2 can overflow, so we distribute.
return (a / 2) + (b / 2) + (((a % 2) + (b % 2)) / 2);
}
```

UNKNOWN SWC-101

Arithmetic operation "%" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol

Locations

```
function average(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b) / 2 can overflow, so we distribute.
return (a / 2) + (b / 2) + (((a % 2) + (b % 2)) / 2);
}
```

UNKNOWN
SWC-101

Arithmetic operation "+" discovered

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

Source file
/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol
Locations

```
function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b - 1) / b can overflow on addition, so we distribute.
return a / b + (a % b == 0 ? 0 : 1);
}
}
```

UNKNOWN
SWC-101

Arithmetic operation "/" discovered

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

Source file
/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol
Locations

```
function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b - 1) / b can overflow on addition, so we distribute.
return a / b + (a % b == 0 ? 0 : 1);
}
}
```

UNKNOWN
SWC-101

Arithmetic operation "%" discovered

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

Source file
/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/math/Math.sol
Locations

```
function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256) {
// (a + b - 1) / b can overflow on addition, so we distribute.
return a / b + (a % b == 0 ? 0 : 1);
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
uint256 digits;
while (temp != 0) {
digits++;
temp /= 10;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/=" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
while (temp != 0) {
digits++;
temp /= 10;
}
bytes memory buffer = new bytes(digits);
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
bytes memory buffer = new bytes(digits);
while (value != 0) {
digits -= 1;
buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
value /= 10;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
while (value != 0) {
digits -= 1;
buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
value /= 10;
}
```

UNKNOWN

SWC-101

Arithmetic operation "%" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
while (value != 0) {
digits -= 1;
buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
value /= 10;
}
```

UNKNOWN

SWC-101

Arithmetic operation "/=" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
digits -= 1;
buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
value /= 10;
}
return string(buffer);
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
uint256 length = 0;
while (temp != 0) {
length++;
temp >>= 8;
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
*/
function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
bytes memory buffer = new bytes(2 * length + 2);
buffer[0] = "0";
buffer[1] = "x";
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
*/
function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
bytes memory buffer = new bytes(2 * length + 2);
buffer[0] = "0";
buffer[1] = "x";
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
buffer[0] = "0";
buffer[1] = "x";
for (uint256 i = 2 * length + 1; i > 1; --i) {
buffer[i] = _HEX_SYMBOLS[value & 0xf];
value >>= 4;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
buffer[0] = "0";
buffer[1] = "x";
for (uint256 i = 2 * length + 1; i > 1; --i) {
buffer[i] = _HEX_SYMBOLS[value & 0xf];
value >>= 4;
```

UNKNOWN Arithmetic operation "--" discovered

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

SWC-101

Source file

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
buffer[0] = "0";
buffer[1] = "x";
for (uint256 i = 2 * length + 1; i > 1; --i) {
buffer[i] = _HEX_SYMBOLS[value & 0xf];
value >>= 4;
```

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

`contracts/HeartToken.sol`

Locations

```
return (false, 0);
} else {
return (true, snapshots.values[index]);
}
}
```

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

`contracts/HeartToken.sol`

Locations

```
return 0;
} else {
return ids[ids.length - 1];
}
}
```

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

`/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/structs/EnumerableSet.sol`

Locations

```

if (lastIndex != toDeleteIndex) {
bytes32 lastvalue = set._values[lastIndex];

// Move the last value to the index where the value to delete is
```

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

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/structs/EnumerableSet.sol

Locations

```

// Move the last value to the index where the value to delete is
set._values[toDeleteIndex] = lastvalue;
// Update the index for the moved value
set._indexes[lastvalue] = valueIndex; // Replace lastvalue's index to valueIndex
```

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

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/structs/EnumerableSet.sol

Locations

```
*/
function _at(Set storage set, uint256 index) private view returns (bytes32) {
return set._values[index];
}
```

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

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Arrays.sol

Locations

```
// Note that mid will always be strictly less than high (i.e. it will be a valid array index)
// because Math.average rounds down (it does integer division with truncation).
if (array[mid] > element) {
high = mid;
} else {
```

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

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Arrays.sol

Locations

```

// At this point `low` is the exclusive upper bound. We will return the inclusive upper bound.
if (low > 0 && array[low - 1] == element) {
return low - 1;
} else {
```

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

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
while (value != 0) {
digits -= 1;
buffer[digits] = bytes1(uint8(48 + uint256(value % 10)));
value /= 10;
}
```

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

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
function toHexString(uint256 value, uint256 length) internal pure returns (string memory) {
bytes memory buffer = new bytes(2 * length + 2);
buffer[0] = "0";
buffer[1] = "x";
for (uint256 i = 2 * length + 1; i > 1; --i) {
```

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

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
bytes memory buffer = new bytes(2 * length + 2);
buffer[0] = "0";
buffer[1] = "x";
for (uint256 i = 2 * length + 1; i > 1; --i) {
buffer[i] = _HEX_SYMBOLS[value & 0xf];
```

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

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
buffer[1] = "x";
for (uint256 i = 2 * length + 1; i > 1; --i) {
buffer[i] = _HEX_SYMBOLS[value & 0xf];
value >>= 4;
}
```

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

/Users/nick/.brownie/packages/OpenZeppelin/openzeppelin-contracts@4.2.0/contracts/utils/Strings.sol

Locations

```
buffer[1] = "x";
for (uint256 i = 2 * length + 1; i > 1; --i) {
buffer[i] = _HEX_SYMBOLS[value & 0xf];
value >>= 4;
}
```