MythX

REPORT 609B63EB556A8D0011DE1461

Created	Wed May 12 2021 05:13:15 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	609b61c3e7c926a8cc011f6b

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
4c9db1c4-c731-4914-accf-f562e9aa4bda	Token.sol	6

Analysis 4c9db1c4-c731-4914-accf-f562e9aa4bda MythX

Started	Wed May 12 2021 05:13:17 GMT+0000 (Coordinated Universal Time)
Finished	Wed May 12 2021 05:58:35 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Token.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	1	5

ISSUES

MEDIUM

SWC-000

Incorrect ERC20 implementation

Contract "BasicTaxableToken" looks like its trying to implement the ERC20 standard, but its missing a required event with signature "event Approval(address indexed, address indexed, uint256)"

Source file

Token.sol

Locations

```
68  * @dev Basic version of StandardToken, with no allowances.
69  */
70  contract BasicTaxableToken is ERC20Basic {
71  using SafeMath for uint256;
72
73  mapping(address => uint256) balances;
74
75  uint256 totalSupply_;
76  uint256 taxRateDivisor_;
77
78  /**
79  * @dev total number of tokens in existence
80  */
81  function totalSupply() public view returns (uint256) {
82  return totalSupply_;
83  }
84
85  function _transfer (
86  address _from,
87  address _to,
88  uint256 _value
89  )
90  internal
91  returns (bool)
92  {
93  uint256 _tax = _value.div(taxRateDivisor_);
94  uint256 _valueAfterTax = _value.sub(_tax);
95
96  balances[_from] = balances[_from].sub(_value);
97  balances[0x0000000000000000000000000000000000000000] = balances[0x0000000000000000000000000000000000000000].add(_tax);
98  balances[_to] = balances[_to].add(_valueAfterTax);
99  emit Transfer(_from, _to, _valueAfterTax);
100 emit Transfer(_from, 0x0000000000000000000000000000000000000000, _tax);
101 return true;
102 }
103
104 /**
105 * @dev transfer token for a specified address
106 * @param _to The address to transfer to.
107 * @param _value The amount to be transferred.
108 */
109 function transfer(address _to, uint256 _value) public returns (bool) {
110 require(_to != address(0));
111 require(_value <= balances[msg.sender]);
112
113 return _transfer(msg.sender, _to, _value);
114 }
115
116 /**
117 * @dev Gets the balance of the specified address.
118 * @param _owner The address to query the the balance of.
119 * @return An uint256 representing the amount owned by the passed address.
120 */
121 function balanceOf(address _owner) public view returns (uint256) {
122 return balances[_owner];
```

```
}
}

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.4.23"". 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

Token.sol

Locations

```
3  */
4
5  pragma solidity ^0.4.23;
6
7  // based on https://github.com/OpenZeppelin/openzeppelin-solidity/tree/v1.10.0
```

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 "balances" is internal. Other possible visibility settings are public and private.

Source file

Token.sol

Locations

```
71  using SafeMath for uint256;
72
73  mapping(address => uint256) balances;
74
75  uint256 totalSupply_;
```

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 "totalSupply_" is internal. Other possible visibility settings are public and private.

Source file

Token.sol

Locations

```
73  mapping(address => uint256) balances;
74
75  uint256 totalSupply_;
76  uint256 taxRateDivisor_;
```

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 "taxRateDivisor_" is internal. Other possible visibility settings are public and private.

Source file

Token.sol

Locations

```
74
75  uint256 totalSupply_;
76  uint256 taxRateDivisor_;
77
78  /**
```

LOW

SWC-110

An assertion violation was triggered.

It is possible to cause an assertion violation. Note that Solidity assert() statements should only be used to check invariants. Review the transaction trace generated for this issue and either make sure your program logic is correct, or use require() instead of assert() if your goal is to constrain user inputs or enforce preconditions. Remember to validate inputs from both callers (for instance, via passed arguments) and callees (for instance, via return values).

Source file

Token.sol

Locations

```
47  function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
48  c = a + b;
49  assert(c >= a);
50  return c;
51  }
```