MythX

REPORT 604AAD0E5F8BEB0019899BF4

Created	Thu Mar 11 2021 23:51:42 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	team@hyruleswap.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
15763c01-8acf-41a8-808d-74a4485f413f	contracts/RupeeToken.sol	19

Analysis 15763c01-8acf-41a8-808d-74a4485f413f MythX

Started	Thu Mar 11 2021 23:51:51 GMT+0000 (Coordinated Universal Time)
Finished	Fri Mar 12 2021 00:37:36 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Cli-0.6.22
Main Source File	Contracts/RupeeToken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	13	6

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/RupeeToken.sol
Locations

```
7  contract RupeeToken is BEP20('Rupee Token', 'RUPEE') {
8  /// @dev Creates `_amount` token to `_to`. Must only be called by the owner (Link).
9  function mint(address _to, uint256 _amount) public onlyOwner {
10 _mint(_to, _amount);
11 _moveDelegates(address(0), _delegates[_to], _amount);
12 }
13
14 // Copied and modified from YAM code:
```

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
contracts/libs/BEP20.sol
Locations

```
78 * name.
79 */
80 function symbol() public override view returns (string memory) {
81 return _symbol;
82 }
83
84 /**
```

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

contracts/libs/BEP20.sol

Locations

```
85  * @dev Returns the number of decimals used to get its user representation.
86  */
87  function decimals() public override view returns (uint8) {
88  return _decimals;
89  }
90
91  /**
```

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

contracts/libs/BEP20.sol

Locations

```
92  * @dev See {BEP20-totalSupply}.
93  */
94  function totalSupply() public override view returns (uint256) {
95  return _totalSupply;
96  }
97
98  /**
```

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

contracts/libs/BEP20.sol

Locations

```
111 * - the caller must have a balance of at least `amount`.
112 */
113 function transfer(address recipient, uint256 amount) public override returns (bool) {
114 _transfer(_msgSender(), recipient, amount);
115 return true;
116 }
117
118 /**
```

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

contracts/libs/BEP20.sol

Locations

```
119 * @dev See {BEP20-allowance}.
120 */
121 function allowance(address owner, address spender) public override view returns (uint256) {
122 return _allowances[owner][spender];
123 }
124
125 /**
```

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

contracts/libs/BEP20.sol

Locations

```
130 * - `spender` cannot be the zero address.
131 */
132 function approve(address spender, uint256 amount) public override returns (bool) {
133 _approve(_msgSender(), spender, amount);
134 return true;
135 }
136
137 /**
```

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

contracts/libs/BEP20.sol

Locations

```
147 * `amount`.
148 */
149 function transferFrom (address sender, address recipient, uint256 amount) public override returns (bool) {
150 _transfer(sender, recipient, amount);
151 _approve(
152 sender,
153 _msgSender(),
154 _allowances[sender][_msgSender()].sub(amount, 'BEP20: transfer amount exceeds allowance')
155 );
156 return true;
157 }
158
159 /**
```

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

contracts/libs/BEP20.sol

Locations

```
169 * - `spender` cannot be the zero address.
170 */
171 function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
172 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
173 return true;
174 }
175
176 /**
```

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

contracts/libs/BEP20.sol

Locations

```
188 * `subtractedValue`.
189 */
190 function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
191 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, 'BEP20: decreased allowance below zero'));
192 return true;
193 }
194
195 /**
```

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

contracts/libs/BEP20.sol

Locations

```
201 * - `msg.sender` must be the token owner
202 */
203 function mint(uint256 amount) public onlyOwner returns (bool) {
204 _mint(_msgSender(), amount);
205 return true;
206 }
207
208 /**
```

MEDIUM

SWC-000

Function could be marked as external.

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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
52  * thereby removing any functionality that is only available to the owner.
53  */
54  function renounceOwnership() public virtual onlyOwner {
55  emit OwnershipTransferred(_owner, address(0));
56  _owner = address(0);
57  }
58
59  /**
```

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

node_modules/@openzeppelin/contracts/access/Ownable.sol

Locations

```
61  * Can only be called by the current owner.
62  */
63  function transferOwnership(address newOwner) public virtual onlyOwner {
64  require(newOwner != address(0), "Ownable: new owner is the zero address");
65  emit OwnershipTransferred(_owner, newOwner);
66  _owner = newOwner;
67  }
68  }
```

LOW

SWC-116

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. 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

contracts/RupeeToken.sol

Locations

```
116 require(signatory != address(0), "TOKEN::delegateBySig: invalid signature");
117 require(nonce == nonces[signatory]++, "TOKEN::delegateBySig: invalid nonce");
118 require(now <= expiry, "TOKEN::delegateBySig: signature expired");
119 return _delegate(signatory, delegatee);
120 }
```

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

contracts/RupeeToken.sol

Locations

```
146 returns (uint256)
147 {
148 require(blockNumber < block.number, "TOKEN::getPriorVotes: not yet determined");
149
150 uint32 nCheckpoints = numCheckpoints[account];
```

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

contracts/RupeeToken.sol

Locations

```
219 internal
220 {
221 uint32 blockNumber = safe32(block.number, "TOKEN::_writeCheckpoint: block number exceeds 32 bits");
222
223 if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
```

LOW

SWC-120

A control flow decision is made based on The block.number environment variable.

The block.number environment variable is used to determine a control flow decision. 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

contracts/RupeeToken.sol

Locations

```
146 returns (uint256)
147 {
148 require(blockNumber < block.number, "TOKEN::getPriorVotes: not yet determined");
149
150 uint32 nCheckpoints = numCheckpoints[account];
```

LOW

SWC-128

Potentially unbounded data structure passed to builtin.

Gas consumption in function "delegateBySig" in contract "RupeeToken" depends on the size of data structures that may grow unboundedly. Specifically the "1-st" argument to builtin "keccak256" may be able to grow unboundedly causing the builtin to consume more gas than the block gas limit, effectively causing a denial-of-service condition.Consider that an attacker might attempt to cause this condition on purpose.

Source file

contracts/RupeeToken.sol

Locations

```
90  abi.encode(
91  DOMAIN_TYPEHASH,
92  keccak256(bytes(name())),
93  getChainId(),
94  address(this)
```

LOW

SWC-128

Loop over unbounded data structure.

Gas consumption in function "getPriorVotes" in contract "RupeeToken" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.

Source file

contracts/RupeeToken.sol

Locations

```
165  uint32 lower = 0;
166  uint32 upper = nCheckpoints - 1;
167  while (upper > lower) {
168  uint32 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
169  Checkpoint memory cp = checkpoints[account][center];
```