MythX

## Analysis 63e57582-c45e-4391-b86e-3bbb8383542c

MythX

| | |
|---|---|
| Started | Wed Jun 02 2021 16:40:05 GMT+0000 (Coordinated Universal Time) |
| Finished | Wed Jun 02 2021 16:55:13 GMT+0000 (Coordinated Universal Time) |
| Mode | Standard |
| Client Tool | Mythx-Vscode-Extension |
| Main Source File | /Contracts/Octopusfriends.Sol |

## DETECTED VULNERABILITIES

| HIGH | MEDIUM | LOW |
|---|---|---|
| 0 | 16 | 4 |

## ISSUES

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

/contracts/octopusfriends.sol

Locations

```
* thereby removing any functionality that is only available to the owner.
*/
function renounceOwnership() public onlyOwner {
emit OwnershipTransferred(_owner, address(0));
_owner = address(0);
}

/**
```

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/octopusfriends.sol

Locations

```
* Can only be called by the current owner.
*/
function transferOwnership(address newOwner) public onlyOwner {
_transferOwnership(newOwner);
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "name" 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/octopusfriends.sol

Locations

```
* @dev Returns the token name.
*/
function name() public override view returns (string memory) {
return _name;
}

/**
```

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

/contracts/octopusfriends.sol

Locations

```
* @dev Returns the token decimals.
*/
function decimals() public override view returns (uint8) {
return _decimals;
}

/**
```

MEDIUM

SWC-000

### Function could be marked as external.

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/octopusfriends.sol`

Locations

```
* @dev Returns the token symbol.
*/
function symbol() public override view returns (string memory) {
return _symbol;
}

/**
```

MEDIUM

SWC-000

### Function could be marked as external.

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/octopusfriends.sol`

Locations

```
* @dev See {BEP20-totalSupply}.
*/
function totalSupply() public override view returns (uint256) {
return _totalSupply;
}

/**
```

MEDIUM

SWC-000

### Function could be marked as external.

The function definition of "balanceOf" 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/octopusfriends.sol`

Locations

```
* @dev See {BEP20-balanceOf}.
*/
function balanceOf(address account) public override view returns (uint256) {
return _balances[account];
}

/**
```

MEDIUM

SWC-000

## Function could be marked as external.

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/octopusfriends.sol

Locations

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

/**
```

MEDIUM

SWC-000

## Function could be marked as external.

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/octopusfriends.sol

Locations

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

/**
```

MEDIUM

SWC-000

## Function could be marked as external.

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/octopusfriends.sol

Locations

```
* - `spender` cannot be the zero address.
*/
function approve(address spender, uint256 amount) public override returns (bool) {
_approve(_msgSender(), spender, amount);
return true;
}

/**
```

MEDIUM

SWC-000

### Function could be marked as external.

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/octopusfriends.sol

Locations

```
* `amount`.
*/
function transferFrom(
address sender,
address recipient,
uint256 amount
) public override returns (bool) {
_transfer(sender, recipient, amount);
_approve(
sender,
_msgSender(),
_allowances[sender][_msgSender()].sub(amount, 'BEP20: transfer amount exceeds allowance')
);
return true;
}

/**
```

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/octopusfriends.sol

Locations

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

/**
```

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/octopusfriends.sol

Locations

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

/**
```

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/octopusfriends.sol

Locations

```
* - `msg.sender` must be the token owner
*/
function mint(uint256 amount) public onlyOwner returns (bool) {
_mint(_msgSender(), amount);
return true;
}

/**
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "setOctopusFriend" 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/octopusfriends.sol`

Locations

```
}

function setOctopusFriend(address farmer, address referrer) public onlyAdmin {
if (referrers[farmer] == address(0) && referrer != address(0)) {
referrers[farmer] = referrer;
referredCount[referrer] += 1;
emit Referral(referrer, farmer);
}
}

function getOctopusFriend(address farmer) public view returns (address) {
```

MEDIUM

SWC-000

## Function could be marked as external.

The function definition of "getOctopusFriend" 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/octopusfriends.sol`

Locations

```
}

function getOctopusFriend(address farmer) public view returns (address) {
return referrers[farmer];
}

// Set admin status.
```

LOW

SWC-103

## A floating pragma is set.

The current pragma Solidity directive is ""^0.6.12"". 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

`/contracts/octopusfriends.sol`

Locations

```
*/

pragma solidity ^0.6.12;

//
```

LOW

SWC-107

## A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Source file

`/contracts/octopusfriends.sol`

Locations

```
function emergencyBEP20Drain(BEP20 token, uint amount) external onlyOwner {
emit EmergencyBEP20Drain(address(token), owner, amount);
token.transfer(owner, amount);
}
}
```

LOW

SWC-123

### Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

Source file

/contracts/octopusfriends.sol

Locations

```
function emergencyBEP20Drain(BEP20 token, uint amount) external onlyOwner {
emit EmergencyBEP20Drain(address(token), owner, amount);
token.transfer(owner, amount);
}
}
```

Source file

/contracts/octopusfriends.sol

Locations

```
}
}
contract OctopusFriends {
using SafeBEP20 for IBEP20;


mapping(address => address) public referrers; // account_address -> referrer_address
mapping(address => uint256) public referredCount; // referrer_address -> num_of_referred

event Referral(address indexed referrer, address indexed farmer);

// Standard contract ownership transfer.
address public owner;
address private nextOwner;

mapping(address => bool) public isAdmin;

constructor () public {
owner = msg.sender;
}

// Standard modifier on methods invokable only by contract owner.
modifier onlyOwner {
require(msg.sender == owner, "OnlyOwner methods called by non-owner.");
_;
}

modifier onlyAdmin {
require(isAdmin[msg.sender], "OnlyAdmin methods called by non-admin.");
_;
}

// Standard contract ownership transfer implementation,
function approveNextOwner(address _nextOwner) external onlyOwner {
require(_nextOwner != owner, "Cannot approve current owner.");
nextOwner = _nextOwner;
}

function acceptNextOwner() external {
require(msg.sender == nextOwner, "Can only accept preapproved new owner.");
owner = nextOwner;
}


```

```
function setOctopusFriend(address farmer, address referrer) public onlyAdmin {
if (referrers[farmer] == address(0) && referrer != address(0)) {
referrers[farmer] = referrer;
referredCount[referrer] += 1;
emit Referral(referrer, farmer);
}
}

function getOctopusFriend(address farmer) public view returns (address) {
return referrers[farmer];
}

// Set admin status.
function setAdminStatus(address _admin, bool _status) external onlyOwner {
isAdmin[_admin] = _status;
}

event EmergencyBEP20Drain(address token, address owner, uint256 amount);

// owner can drain tokens that are sent here by mistake
function emergencyBEP20Drain(BEP20 token, uint amount) external onlyOwner {
emit EmergencyBEP20Drain(address(token), owner, amount);
token.transfer(owner, amount);
}
}
```

LOW

SWC-128

## Loop over unbounded data structure.

Gas consumption in function "sqrt" in contract "SafeMath" 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/octopusfriends.sol`

Locations

```
z = y;
uint256 x = y / 2 + 1;
while (x < z) {
z = x;
x = (y / x + x) / 2;
```