MythX

REPORT 60F85312D4B5330019E7B9C1

Created	Wed Jul 21 2021 17:02:10 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	60b6a744a6e1845c77c6e3dc

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
68421950-51cd-4056-9399-28148b16f935	/contracts-v1/flashreferral.sol	14

Analysis 68421950-51cd-4056-9399-28148b16f935

MythX

Started	Wed Jul 21 2021 17:02:12 GMT+0000 (Coordinated Universal Time)
Finished	Wed Jul 21 2021 17:17:28 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts-V1/Flashreferral.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
1	5	8

ISSUES

HIGH

SWC-101

The arithmetic operation can overflow.

It is possible to cause an arithmetic overflow. Prevent the overflow by constraining inputs using the require() statement or use the OpenZeppelin SafeMath library for integer arithmetic operations. Refer to the transaction trace generated for this issue to reproduce the overflow.

Source file

/contracts-v1/flashreferral.sol

Locations

```
function recordReferralCommission(address _referrer, uint256 _commission) public override onlyOperator {
if (_referrer != address(0) && _commission > 0) {
totalReferralCommissions[_referrer] += _commission;
emit ReferralCommissionRecorded(_referrer, _commission);
}
```

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-v1/flashreferral.sol

Locations

```
* thereby removing any functionality that is only available to the owner.
*/
function renounceOwnership() public virtual 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-v1/flashreferral.sol`

Locations

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

```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "recordReferral" 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-v1/flashreferral.sol`

Locations

```
}

function recordReferral(address _user, address _referrer) public override onlyOperator {
if (_user != address(0)
&& _referrer != address(0)
&& _user != _referrer
&& referrers[_user] == address(0)
) {
referrers[_user] = _referrer;
referralsCount[_referrer] += 1;
emit ReferralRecorded(_user, _referrer);
}
}

function recordReferralCommission(address _referrer, uint256 _commission) public override onlyOperator {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "recordReferralCommission" 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-v1/flashreferral.sol`

Locations

```
}

function recordReferralCommission(address _referrer, uint256 _commission) public override onlyOperator {
if (_referrer != address(0) && _commission > 0) {
totalReferralCommissions[_referrer] += _commission;
emit ReferralCommissionRecorded(_referrer, _commission);
}
}

// Get the referrer address that referred the user
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getReferrer" 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-v1/flashreferral.sol`

Locations

```

// Get the referrer address that referred the user
function getReferrer(address _user) public override view returns (address) {
return referrers[_user];
}

// Update the status of the operator
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.0<0.8.0"". 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-v1/flashreferral.sol`

Locations

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.6.0 <0.8.0;

/*
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.0<0.8.0"". 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-v1/flashreferral.sol`

Locations

```
// File: @openzeppelin/contracts/access/Ownable.sol

pragma solidity >=0.6.0 <0.8.0;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.2<0.8.0"". 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-v1/flashreferral.sol`

Locations

```
// File: @openzeppelin/contracts/utils/Address.sol

pragma solidity >=0.6.2 <0.8.0;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.0<0.8.0"". 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-v1/flashreferral.sol`

Locations

```
// File: @openzeppelin/contracts/math/SafeMath.sol

pragma solidity >=0.6.0 <0.8.0;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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-v1/flashreferral.sol`

Locations

```
// File: contracts/libs/SafeBEP20.sol

pragma solidity ^0.6.0;


```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.4.0"". 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-v1/flashreferral.sol`

Locations

```
// File: contracts/libs/IBEP20.sol

pragma solidity >=0.4.0;

interface IBEP20 {
```

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-v1/flashreferral.sol`

Locations

```

// solhint-disable-next-line avoid-low-level-calls
(bool success, bytes memory returndata) = target.call{ value: value }(data);
return _verifyCallResult(success, returndata, errorMessage);
}
```

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-v1/flashreferral.sol

Locations

```

// solhint-disable-next-line avoid-low-level-calls
(bool success, bytes memory returndata) = target.call{ value: value }(data);
return _verifyCallResult(success, returndata, errorMessage);
}
```

Source file

/contracts-v1/flashreferral.sol

Locations

```


contract FlashReferral is IFlashReferral, Ownable {
using SafeBEP20 for IBEP20;

mapping(address => bool) public operators;
mapping(address => address) public referrers; // user address => referrer address
mapping(address => uint256) public referralsCount; // referrer address => referrals count
mapping(address => uint256) public totalReferralCommissions; // referrer address => total referral commissions

event ReferralRecorded(address indexed user, address indexed referrer);
event ReferralCommissionRecorded(address indexed referrer, uint256 commission);
event OperatorUpdated(address indexed operator, bool indexed status);

modifier onlyOperator {
require(operators[msg.sender], "Operator: caller is not the operator");
_;
}

function recordReferral(address _user, address _referrer) public override onlyOperator {
if (_user != address(0)
&& _referrer != address(0)
&& _user != _referrer
&& referrers[_user] == address(0)
) {
referrers[_user] = _referrer;
referralsCount[_referrer] += 1;
emit ReferralRecorded(_user, _referrer);
}
}

function recordReferralCommission(address _referrer, uint256 _commission) public override onlyOperator {
if (_referrer != address(0) && _commission > 0) {
totalReferralCommissions[_referrer] += _commission;
emit ReferralCommissionRecorded(_referrer, _commission);
}
}

// Get the referrer address that referred the user
function getReferrer(address _user) public override view returns (address) {
return referrers[_user];
}

// Update the status of the operator
function updateOperator(address _operator, bool _status) external onlyOwner {
```

```
operators[_operator] = _status;
emit OperatorUpdated(_operator, _status);
}

// Owner can drain tokens that are sent here by mistake
function drainBEP20Token(IBEP20 _token, uint256 _amount, address _to) external onlyOwner {
_token.safeTransfer(_to, _amount);
}
}
```