MythX

REPORT 60F9088552E9EC001996220A

Created	Thu Jul 22 2021 05:56:21 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	60f906b2a6e184dcafc6e947

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
42d78dff-5b85-4293-858c-ba8570989011	BaronReferral.sol	8

Analysis 42d78dff-5b85-4293-858c-ba8570989011 MythX

Started	Thu Jul 22 2021 05:56:27 GMT+0000 (Coordinated Universal Time)
Finished	Thu Jul 22 2021 06:41:46 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	BaronReferral.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
1	5	2

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

BaronReferral.sol

Locations

```
706 function recordReferralCommission(address _referrer, uint256 _commission) public override onlyOperator {
707 if (_referrer != address(0) && _commission > 0) {
708 totalReferralCommissions[_referrer] += _commission;
709 emit ReferralCommissionRecorded(_referrer, _commission);
710 }
```

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

BaronReferral.sol

Locations

```
659 * thereby removing any functionality that is only available to the owner.
660 */
661 function renounceOwnership() public virtual onlyOwner {
662 emit OwnershipTransferred(_owner, address(0));
663 _owner = address(0);
664 }
665
666 /**
```

MEDIUM

Function could be marked as external.

SWC-000

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

BaronReferral.sol

Locations

```
668 * Can only be called by the current owner.
669 */
670 function transferOwnership(address newOwner) public virtual onlyOwner {
671 require(newOwner != address(0), "Ownable: new owner is the zero address");
672 emit OwnershipTransferred(_owner, newOwner);
673 _owner = newOwner;
674 }
675 }
676
```

MEDIUM

Function could be marked as external.

SWC-000

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

BaronReferral.sol

Locations

```
692 }
693
694 function recordReferral(address _user, address _referrer) public override onlyOperator {
695 if (_user != address(0)
696 && _referrer != address(0)
697 && _user != _referrer
698 && referrers[_user] == address(0)
699 ) {
700 referrers[_user] = _referrer;
701 referralsCount[_referrer] += 1;
702 emit ReferralRecorded(_user, _referrer);
703 }
704 }
705
706 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

BaronReferral.sol

Locations

```
704 }
705
706 function recordReferralCommission(address _referrer, uint256 _commission) public override onlyOperator {
707 if (_referrer != address(0) && _commission > 0) {
708 totalReferralCommissions[_referrer] += _commission;
709 emit ReferralCommissionRecorded(_referrer, _commission);
710 }
711 }
712
713 // 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

BaronReferral.sol

Locations

```
712
713 // Get the referrer address that referred the user
714 function getReferrer(address _user) public override view returns (address) {
715 return referrers[_user];
716 }
717
718 // Update the status of the operator
```

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

BaronReferral.sol

Locations

```
414
415 // solhint-disable-next-line avoid-low-level-calls
416 (bool success, bytes memory returndata) = target.call{ value: value }(data);
417 return _verifyCallResult(success, returndata, errorMessage);
418 }
```

LOW

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).

SWC-123

Source file

BaronReferral.sol

Locations

```
414
415 // solhint-disable-next-line avoid-low-level-calls
416 (bool success, bytes memory returndata) = target.call{ value: value }(data);
417 return _verifyCallResult(success, returndata, errorMessage);
418 }
```

Source file

BaronReferral.sol

Locations

```
675 }
676
677 contract BaronReferral is IBaronReferral, Ownable {
678 using SafeBEP20 for IBEP20;
679
680 mapping(address => bool) public operators;
681 mapping(address => address) public referrers; // user address => referrer address
682 mapping(address => uint256) public referralsCount; // referrer address => referrals count
683 mapping(address => uint256) public totalReferralCommissions; // referrer address => total referral commissions
684
685 event ReferralRecorded(address indexed user, address indexed referrer);
686 event ReferralCommissionRecorded(address indexed referrer, uint256 commission);
687 event OperatorUpdated(address indexed operator, bool indexed status);
688
689 modifier onlyOperator {
690 require(operators[msg.sender], "Operator: caller is not the operator");
691 _;
692 }
693
694 function recordReferral(address _user, address _referrer) public override onlyOperator {
695 if (_user != address(0)
696 && _referrer != address(0)
697 && _user != _referrer
698 && referrers[_user] == address(0)
699 ) {
700 referrers[_user] = _referrer;
701 referralsCount[_referrer] += 1;
702 emit ReferralRecorded(_user, _referrer);
703 }
704 }
705
706 function recordReferralCommission(address _referrer, uint256 _commission) public override onlyOperator {
707 if (_referrer != address(0) && _commission > 0) {
708 totalReferralCommissions[_referrer] += _commission;
709 emit ReferralCommissionRecorded(_referrer, _commission);
710 }
711 }
712
713 // Get the referrer address that referred the user
714 function getReferrer(address _user) public override view returns (address) {
715 return referrers[_user];
716 }
717
718 // Update the status of the operator
719 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);
}
}
```