REALPrivateSale.sol

• Owner Role / Centralized Risk

Only the owner role has authority over the functions shown below. Any compromise to the _owner account may allow the hacker to take advantage of this authority.

```
50        * @dev Add new beneficiary to vesting contract with some conditions.
51        *  _amount : Total amount include _upfrontAmount
52        */
53       function addBeneficiary(address[] calldata _beneficiarys, uint256[] calldata _amounts) external onlyOwner {
54           require (_beneficiarys.length == _amounts.length,"Input not correct");
55           uint256 totalAmount;
56           for(uint256 i=0; i <_beneficiarys.length; i++){
57               address addr = _beneficiarys[i];
58               require(addr != address(0), "The beneficiary's address cannot be 0");
59               require(!exists[addr],"address was exists");
60               require(_amounts[i] > 0, "Shares amount has to be greater than 0");
61               require(lockTokens[addr].amountLock == 0, "The beneficiary has added to the vesting pool already");
62               listBeneficiaries[totalBeneficiaries.add(i+1)] = addr;
63               lockTokens[addr].amountLock = _amounts[i];
64               lockTokens[addr].nextRelease = START_TIME;
65               totalAmount = totalAmount.add(_amounts[i]);
66               exists[addr] = true;
67           }
68           require(REAL_TOKEN.allowance(ownerToken, address(this)) >= totalAmount,"Can not add more beneficiary");
69           REAL_TOKEN.safeTransferFrom(ownerToken, address(this), totalAmount);
70           totalBeneficiaries = totalBeneficiaries.add(_beneficiarys.length);
71       }
```

✓ Recommendation

The risk describes the current project design and potentially makes iterations to improve the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

REALPublicSale.sol

• Owner Role / Centralized Risk

Only the owner role has authority over the functions shown below. Any compromise to the _owner account may allow the hacker to take advantage of this authority.

```
50        * @dev Add new beneficiary to vesting contract with some conditions.
51        *  _amount : Total amount include _upfrontAmount
52        */
53       function addBeneficiary(address[] calldata _beneficiarys, uint256[] calldata _amounts) external onlyOwner {
54           require (_beneficiarys.length == _amounts.length,"Input not correct");
55           uint256 totalAmount;
56           for(uint256 i=0; i <_beneficiarys.length; i++){
57               address addr = _beneficiarys[i];
58               require(addr != address(0), "The beneficiary's address cannot be 0");
59               require(!exists[addr],"address was exists");
60               require(_amounts[i] > 0, "Shares amount has to be greater than 0");
61               require(lockTokens[addr].amountLock == 0, "The beneficiary has added to the vesting pool already");
62               listBeneficiaries[totalBeneficiaries.add(i+1)] = addr;
63               lockTokens[addr].amountLock = _amounts[i];
64               lockTokens[addr].nextRelease = START_TIME;
65               totalAmount = totalAmount.add(_amounts[i]);
66               exists[addr] = true;
67           }
68           require(REAL_TOKEN.allowance(ownerToken, address(this)) >= totalAmount,"Can not add more beneficiary");
69           REAL_TOKEN.safeTransferFrom(ownerToken, address(this), totalAmount);
70           totalBeneficiaries = totalBeneficiaries.add(_beneficiarys.length);
71       }
```

✓ Recommendation

The risk describes the current project design and potentially makes iterations to improve the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

REALSeedSale.sol

• Owner Role / Centralized Risk

Only the owner role has authority over the functions shown below. Any compromise to the _owner account may allow the hacker to take advantage of this authority.

```
50        * @dev Add new beneficiary to vesting contract with some conditions.
51        *  _amount : Total amount include _upfrontAmount
52        */
53       function addBeneficiary(address[] calldata _beneficiarys, uint256[] calldata _amounts) external onlyOwner {
54           require (_beneficiarys.length == _amounts.length,"Input not correct");
55           uint256 totalAmount;
56           for(uint256 i=0; i <_beneficiarys.length; i++){
57               address addr = _beneficiarys[i];
58               require(addr != address(0), "The beneficiary's address cannot be 0");
59               require(!exists[addr],"address was exists");
60               require(_amounts[i] > 0, "Shares amount has to be greater than 0");
61               require(lockTokens[addr].amountLock == 0, "The beneficiary has added to the vesting pool already");
62               listBeneficiaries[totalBeneficiaries.add(i+1)] = addr;
63               lockTokens[addr].amountLock = _amounts[i];
64               lockTokens[addr].nextRelease = START_TIME;
65               totalAmount = totalAmount.add(_amounts[i]);
66               exists[addr] = true;
67           }
68           require(REAL_TOKEN.allowance(ownerToken, address(this)) >= totalAmount,"Can not add more beneficiary");
69           REAL_TOKEN.safeTransferFrom(ownerToken, address(this), totalAmount);
70           totalBeneficiaries = totalBeneficiaries.add(_beneficiarys.length);
71       }
```

✓ Recommendation

The risk describes the current project design and potentially makes iterations to improve the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

RealTeamAndAdvisor.sol

• Owner Role / Centralized Risk

Only the owner role has authority over the functions shown below. Any compromise to the _owner account may allow the hacker to take advantage of this authority.

```
50        * @dev Add new beneficiary to vesting contract with some conditions.
51        *  _amount : Total amount include _upfrontAmount
52        */
53       function addBeneficiary(address[] calldata _beneficiarys, uint256[] calldata _amounts) external onlyOwner {
54           require (_beneficiarys.length == _amounts.length,"Input not correct");
55           uint256 totalAmount;
56           for(uint256 i=0; i <_beneficiarys.length; i++){
57               address addr = _beneficiarys[i];
58               require(addr != address(0), "The beneficiary's address cannot be 0");
59               require(!exists[addr],"address was exists");
60               require(_amounts[i] > 0, "Shares amount has to be greater than 0");
61               require(lockTokens[addr].amountLock == 0, "The beneficiary has added to the vesting pool already");
62               listBeneficiaries[totalBeneficiaries.add(i+1)] = addr;
63               lockTokens[addr].amountLock = _amounts[i];
64               lockTokens[addr].nextRelease = START_TIME;
65               totalAmount = totalAmount.add(_amounts[i]);
66               exists[addr] = true;
67           }
68           require(REAL_TOKEN.allowance(ownerToken, address(this)) >= totalAmount,"Can not add more beneficiary");
69           REAL_TOKEN.safeTransferFrom(ownerToken, address(this), totalAmount);
70           totalBeneficiaries = totalBeneficiaries.add(_beneficiarys.length);
71       }
```

✓ Recommendation

The risk describes the current project design and potentially makes iterations to improve the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

REALREALM-token.sol

Medium severity issues

• Initial Token Distribution / Centralization

In the constructor function, the deployer is minting tokens for multiple hardcoded addresses. This could be a centralization risk, the deployer can distribute those tokens without obtaining the consensus of the community.

```
29          0x47dA2d7674FdFa8e16ef0EE306c337f0F4548806,
30          250000000 * 10**decimals()
31        ); // Game Rewards - 250.000.000
32        _mint(
33          0x2d082504DF6d138Bdc669cc7914D0B16df863dd9,
34          93000000 * 10**decimals()
35        ); // Staking Rewards - 93.000.000
36        _mint(
37          0x76B7d23274B73801327F22C2D2BA79E316500750,
38          95000000 * 10**decimals()
39        ); // Marketing - 95.000.000
40        _mint(
41          0x76B7d23274B73801327F22C2D2BA79E316500750,
42          100000000 * 10**decimals()
43        ); // Ecosystem - 100.000.000
44        _mint(
45          0x5A4010D01377A515F64633403F9cdaef91f8a631,
46          100000000 * 10**decimals()
47        ); // Liquidity - 100.000.000
48        _mint(
49          0x0D1Ba337c0f17322155EDDbc1c2f246d61698D40,
50          150000000 * 10**decimals()
51        ); // Team - 150.000.000
```

✓ Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key, We also advise the client to adopt Multisig, Timelock, and/or DAO in the project to manage the specific account in this case.

• Owner Role / Centralized Risk

Only the owner role has authority over the functions shown below. Any compromise to the _owner account may allow the hacker to take advantage of this authority.

```
62      function pause() public onlyOwner {
63        _pause();
64      }
65
66      function unpause() public onlyOwner {
67        _unpause();
68      }
```

```
104       Users make mistake by transfering usdt/usdt ... to contract address.
105       This function allows contract owner to withdraw those tokens and send back to users.
106       */
107     function rescueStuckErc20(address _token) external onlyOwner {
108       uint256 _amount = ERC20(_token).balanceOf(address(this));
109       ERC20(_token).transfer(owner(), _amount);
110     }
```

✓ Recommendation

The risk describes the current project design and potentially makes iterations to improve the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Low severity issues

• Unchecked Value of ERC-20 transfer()

In function incurDebt,the external call to transfer of ERC20 contracts and the return value is not checked in either case.

```
104       Users make mistake by transfering usdt/usdt ... to contract address.
105       This function allows contract owner to withdraw those tokens and send back to users.
106       */
107     function rescueStuckErc20(address _token) external onlyOwner {
108       uint256 _amount = ERC20(_token).balanceOf(address(this));
109       ERC20(_token).transfer(owner(), _amount);
110     }
```

✓ Recommendation

We advise the team to use SafeERC20 or make sure that the value returned from transfer()' is checked.