BlockSAFU

ADVANCE MANUAL SMART CONTRACT AUDIT

Project: SatoshiSwap

Website: https://satoshiswap.exchange

PASSED

BlockSAFU Score:

79

Contract Address:

0x16d93ab88024607b52cC02b0e12D05aAdd0eec13

DISCLAMER

BlockSAFU has completed this report to provide a summary of the Smart Contract functions, and any security, dependency, or cybersecurity vulnerabilities. This is often a constrained report on our discoveries based on our investigation and understanding of the current programming versions as of this report's date. To understand the full scope of our analysis, it is vital for you to at the date of this report. To understand the full scope of our analysis, you need to review the complete report. Although we have done our best in conducting our investigation and creating this report, it is vital to note that you should not depend on this report and cannot make any claim against BlockSAFU or its Subsidiaries and Team members on the premise of what has or has not been included in the report. Please remember to conduct your independent examinations before making any investment choices. We do not provide investment advice or in any way claim to determine if the project will be successful or not.

By perusing this report or any portion of it, you concur to the terms of this disclaimer. In the unlikely situation where you do not concur with the terms, you should immediately terminate reading this report, and erase and discard any duplicates of this report downloaded and/or printed by you. This report is given for data purposes as it were and on a non-reliance premise and does not constitute speculation counsel. No one should have any right to depend on the report or its substance, and BlockSAFU and its members (including holding companies, shareholders, backups, representatives, chiefs, officers, and other agents) BlockSAFU and its subsidiaries owe no obligation of care towards you or any other person, nor does BlockSAFU make any guarantee or representation to any individual on the precision or completeness of the report.

ABOUT THE AUDITOR:

BlockSAFU (BSAFU) is an Anti-Scam Token Utility that reviews Smart Contracts and Token information to Identify Rug Pull and Honey Pot scamming activity. BlockSAFUs Development Team consists of several Smart Contract creators, Auditors Developers, and Blockchain experts. BlockSAFU provides solutions, prevents, and hunts down scammers. BSAFU is a utility token with features Audit, KYC, Token Generators, and Bounty Scammers. It will enrich the crypto ecosystem.

SATOSHISWAP - TOKEN

OVERVIEW

Mint Function

- The owner can mint after the initial deployment.

SMART CONTRACT REVIEW

Token Name	SatoshiSwap
Token Symbol	SAT
Token Decimal	18
Network	Core DAO
Contract Address	0xcbBf51F6d153f2745a9763515dAbcb05F1B69701
Deployer Address	0x2a95ae0163434f9FE47230562F560ff938900B3E
Owner Address	0x939258404cd941f1506eECe44eE289656105a584
Contract Created	2023-04-04T08:55:42.000+0000
Verified CA	Yes
Compiler	v0.8.19+commit.7dd6d404
Optimization	Yes with runs
Sol License	MIT License
Other	evmVersion MIT License (MIT)

MANUAL CODE REVIEW

● Minor-risk

1 minor-risk code issue found

Could be fixed, and will not bring problems.

1. A floating pragma is set. [SWC-103]

   Risk Scenario: The current pragma Solidity directive is " ^0.8.1, ^0.8.2, ^0.8.0,"

   Risk Recommendation: 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.

```
Pragma solidity ^0.8.1;
pragma solidity ^0.8.2;
pragma solidity ^0.8.0;
```

● Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

● High-Risk

0 high-risk code issues found

Must be fixed, and will bring problems.

Infomational

2 Things you need to know.

1. Mint Fuction

   Risk Scenario: The owner can mint after the initial deployment.
   Risk Recommendation: Investors must be aware that this function exists and know the risks.

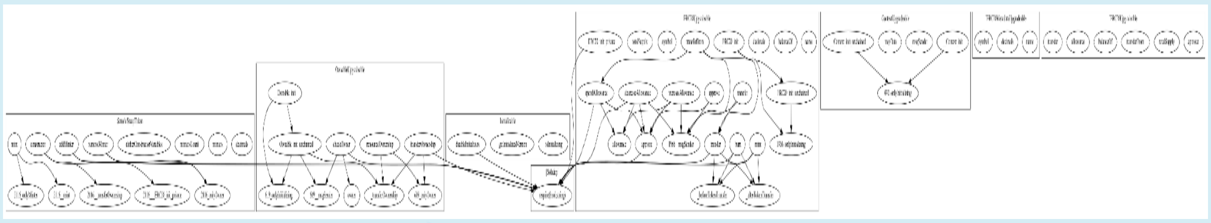
```
function mint(address _to, uint256 _amount) external onlyMinter {
   _mint(_to, _amount);
 }
```

2. Add Minter

   Risk Scenario: The owner can mint after the initial deployment.
   Risk Recommendation: Investors must be aware that this function exists and know the risks.

```
function addMinter(address _minter) external onlyOwner {
   require(_minter != address(0), "ZERO_ADDRESS");
   require(!_minters.contains(_minter), "DUPLICATED");
   _minters.add(_minter);
 }
```

CONTRACT INHERITANCE



RUG-PULL REVIEW

Based on the available information analyzed by us, we come to the following conclusions:

🔴 There is no information about the KYC team yet.

🔴 The owner can mint after the initial deployment.

HONEYPOT REVIEW

🟢 No Honeypot scenario.

SATOSHISWAP - ROUTER

OVERVIEW

External Call

- External call to core contract

SMART CONTRACT REVIEW

Network	Core DAO
Contract Name	SatoshiSwapRouter
Contract Address	0x5A49d47eb0a8Acf531d3eEA917E9096D13772908
Deployer Address	0x2a95ae0163434f9FE47230562F560ff938900B3E
Contract Created	2023-04-04T08:47:33.000+0000
Verified CA	Yes
Compiler	v0.8.19+commit.7dd6d404
Optimization	Yes with runs
Sol License	MIT License
Other	evmVersion MIT License (MIT)

MANUAL CODE REVIEW

● Minor-risk

0 minor-risk code issue found

Could be fixed, and will not bring problems.

● Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

● High-Risk

0 high-risk code issues found

Must be fixed, and will bring problems.

Infomational

2 Things you need to know.

1. Fallback Receive

   Risk Scenario: Only weth contract can receive fallback weth

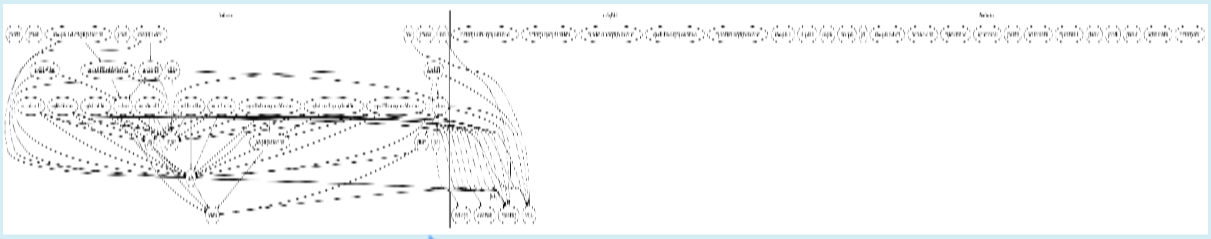
```
receive() external payable {
        assert(msg.sender == WETH); // only accept ETH via fallback from the WETH contract
    }
```

2. Deadline Mechanism

   Risk Scenario: it will be return expired if deadline >= block.timestamp (used in add liquidity and remove liquidity, etc)

```
modifier ensure(uint deadline) {
        require(deadline == 0 || deadline >= block.timestamp, 'EXPIRED');
        _;
    }
```

CONTRACT INHERITANCE



SATOSHISWAP - TIME LOCK

OVERVIEW

SMART CONTRACT REVIEW

Network	Core DAO
Contract Name	Timelock
Contract Address	0x5A49d47eb0a8Acf531d3eEA917E9096D13772908
Deployer Address	0x2a95ae0163434f9FE47230562F560ff938900B3E
Contract Created	2023-04-04T08:47:33.000+0000
Verified CA	Yes
Compiler	v0.8.19+commit.7dd6d404
Optimization	Yes with runs
Sol License	MIT License
Other	evmVersion MIT License (MIT)

MANUAL CODE REVIEW

● Minor-risk

1 minor-risk code issue found

Could be fixed, and will not bring problems.

1. A control flow decision is made based on The block.timestamp environment variable. [SWC-116]

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

   Risk Recommendation: 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.

```
require(c >= a, "SafeMath: addition overflow");
require(msg.sender == admin, "Timelock::queueTransaction: Call
must come from admin.");
        require(
```

● Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

● High-Risk

0 high-risk code issues found

Must be fixed, and will bring problems.

● Infomational

2 Things you need to know.

1. Set delay

   Risk Scenario: The owner can’t set delay transaction over 30 Days

```
uint256 public constant MAXIMUM_DELAY = 30 days;
…
function setDelay(uint256 delay_) external {
       require(msg.sender == address(this), "Timelock::setDelay:
Call must come from Timelock.");
       require(delay_ <= MAXIMUM_DELAY, "Timelock::setDelay: Delay
must not exceed maximum delay.");
       delay = delay_;       emit NewDelay(delay);
   }
```

2. Set Operator

   Risk Scenario: The owner can set operator for do change in contract, like set delay etc

```
function acceptAdmin() external {
       require(msg.sender == pendingAdmin, "Timelock::acceptAdmin:
Call must come from pendingAdmin.");
       admin = msg.sender;
       pendingAdmin = address(0);       emit NewAdmin(admin);
   }   function setPendingAdmin(address pendingAdmin_) external {
       // allows one time setting of admin for deployment purposes
       if (admin_initialized) {
           require(msg.sender == address(this),
"Timelock::setPendingAdmin: Call must come from Timelock.");
       } else {
              require(msg.sender == admin,
"Timelock::setPendingAdmin: First call must come from admin.");
              admin_initialized = true;
          }
          pendingAdmin = pendingAdmin_;       emit
   NewPendingAdmin(pendingAdmin);
        }
```

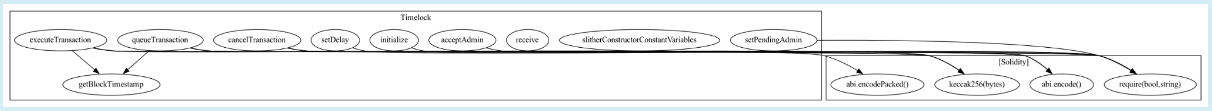
3. Grace Period

Risk Scenario: The owner (admin), can set again estimate time 14 Days after transaction delay expired

```
uint256 public constant GRACE_PERIOD = 14 days;

require(getBlockTimestamp() <= eta.add(GRACE_PERIOD),
"Timelock::executeTransaction: Transaction is stale.");
```

CONTRACT INHERITANCE



SATOSHISWAP - FACTORY

OVERVIEW

- External call to core contract

SMART CONTRACT REVIEW

Token Name	SatoshiSwap LP
Token Symbol	Satoshi-LP
Token Decimal	18
Network	Core DAO
Contract Name	SatoshiSwapFactory
Contract Address	0x8f5c03a1c86bf79ae0baC0D72E75aee662083e26
Deployer Address	0x2a95ae0163434f9FE47230562F560ff938900B3E
Owner Address	
Contract Created	2023-04-04T08:45:57.000+0000
Verified CA	Yes
Compiler	v0.8.19+commit.7dd6d404
Optimization	Yes with runs
Sol License	MIT License
Other	evmVersion MIT License (MIT)

MANUAL CODE REVIEW

● Minor-risk

2 minor-risk code issue found

Could be fixed, and will not bring problems.

1. Requirement violation.[SWC-123]

   Risk Scenario: A requirement was violated in a nested call and the call was reverted as a result.

   Risk Recommendation: Make sure valid inputs are provided to the nested call (for instance, via passed arguments)

```
contract SatoshiSwapFactory is ISatoshiSwapFactory {
uint bal = IERC20(_token).balanceOf(address(this));
```

2. Read of persistent state following external call. [SWC-107]

   Risk Scenario: The contract account state is accessed after an external call.

   Risk Recommendation: To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state

```
contract SatoshiSwapFactory is ISatoshiSwapFactory {
uint bal = IERC20(_token).balanceOf(address(this));
```

● Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

● High-Risk

0 high-risk code issues found

Must be fixed, and will bring problems.

● Infomational

2 Things you need to know.

1. Auto Lock

   Risk Scenario: After mint LP, Liquidity Protocol will be locked automatic

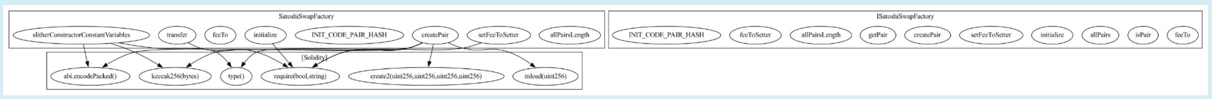
```
function mint(address to) external override lock returns (uint liquidity) {
       uint _totalSupply = totalSupply();
       (uint _liquidity, uint amount0, uint amount1) = core.mint(_totalSupply);
       if (_totalSupply == 0) {
           _mint(address(0), MINIMUM_LIQUIDITY); // permanently lock the first MINIMUM_LIQUIDITY tokens
       }
       liquidity = _liquidity;
       _mint(to, _liquidity);
       _update();
          emit Mint(msg.sender, amount0, amount1);
          core.emitMint(msg.sender, amount0, amount1);
      }
```

2. Minimum Add Liquidity

   Risk Scenario: Minimum Liquidity is 1000

```
uint public constant MINIMUM_LIQUIDITY = 10**3;
```

CONTRACT INHERITANCE



SATOSHISWAP - MASTER CHEF

OVERVIEW

Fees

- DEV FEE 0% (owner can’t set over 10%)
- DEPOSIT 0% (owner can’t set over 10%)

SMART CONTRACT REVIEW

Network	**Core DAO**
Contract Name	SatoshiSwapFactory
Contract Address	0x8f5c03a1c86bf79ae0baC0D72E75aee662083e26
Deployer Address	0x2a95ae0163434f9FE47230562F560ff938900B3E
Contract Created	2023-04-04T09:08:30.000+0000
Verified CA	Yes
Compiler	v0.8.19+commit.7dd6d404
Optimization	Yes with runs
Sol License	MIT License
Other	evmVersion MIT License (MIT)

FEEs

Name	Value	address
DEV FEE	0%	0x1f353B28589c071eAD756342bE037363Bc61493
DEPOSIT	0%	0x1f353B28589c071eAD756342bE037363Bc61493

MANUAL CODE REVIEW

● Minor-risk

2 minor-risk code issue found

Could be fixed, and will not bring problems.

1. A floating pragma is set. [SWC-103]

   Risk Scenario: The current pragma Solidity directive is ""^0.8.0", ^O.8.1".

   Risk Recommendation: 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.

```
1. pragma solidity ^0.8.0;
2. pragma solidity ^0.8.1;
```

2. Potential use of "block.number" as source of randonmness. [SWC-120]

   Risk Scenario: The environment variable "block.number" looks like it might be used as a source of randomness.

   Risk Recommendation: 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.

```
1. if (block.number < pool.startBlock) {
2. if (block.number > pool.lastRewardBlock && lpSupply != 0) {
3. uint reward = (block.number - pool.lastRewardBlock) *
   rewardPerBlock * pool.allocPoint / totalAllocPoint;
4. if (block.number <= pool.lastRewardBlock) {
5. if (lpSupply == 0 || pool.allocPoint == 0 || block.number <
   pool.startBlock) {
6. pool.lastRewardBlock = block.number;
7. uint reward = (block.number - pool.lastRewardBlock) *
   rewardPerBlock * pool.allocPoint / totalAllocPoint;
8. pool.lastRewardBlock = block.number;
```

3. Requirement violation. [SWC-123]

Risk Scenario: A requirement was violated in a nested call and the call was reverted as a result.

Risk Recommendation: Make sure valid inputs

```
contract MasterChef is Ownable {
```

● Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

● High-Risk

0 high-risk code issues found

Must be fixed, and will bring problems.

● Infomational

2 Things you need to know.

1. Withdraw

   Risk Scenario: Withdraw lp function from masterchef

```
// Withdraw LP tokens from masterchef
 function withdraw(uint _pid, uint _amount) external {
   require(_pid < poolInfo.length, "INVALID_POOL");
   PoolInfo storage pool = poolInfo[_pid];
   UserInfo storage user = userInfo[_pid][msg.sender];
   require(user.amount >= _amount, "WITHDRAW_NOT_GOOD");
   updatePool(_pid);
   uint pending = (user.amount * pool.accRewardPerShare / 1e12) -
user.rewardDebt;
   if (pending > 0) {
        rewardToken.mint(msg.sender, pending);
      }
      if (_amount > 0) {
        user.amount -= _amount;
        pool.lpToken.safeTransfer(address(msg.sender), _amount);
      }
      user.rewardDebt = user.amount * pool.accRewardPerShare /
1e12;
      emit Withdraw(msg.sender, _pid, _amount);
    }
```

2. Emergency Withdraw

   Risk Scenario: Emergency Withdraw

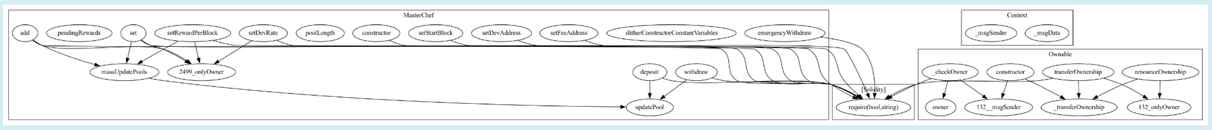
```
function emergencyWithdraw(uint _pid) external {
   require(_pid < poolInfo.length, "INVALID_POOL");
   PoolInfo storage pool = poolInfo[_pid];
   UserInfo storage user = userInfo[_pid][msg.sender];
   uint amount = user.amount;
   user.amount = 0;
   user.rewardDebt = 0;
   pool.lpToken.safeTransfer(address(msg.sender), amount);
   emit EmergencyWithdraw(msg.sender, _pid, amount);
    }
```

3. Set Fees

Risk Scenario: The owner can't set fees over 10%

```
uint private constant MAX_DEV_RATE = 1100;
function setDevRate(uint _devRate) external onlyOwner {
   require(_devRate < MAX_DEV_RATE, "INVALID_RATE");
   devRate = _devRate;
 }
```

CONTRACT INHERITANCE



SATOSHISWAP - CORE

OVERVIEW

Fees

- DEV FEE 0% (owner can’t set over 19.99%)
- DEPOSIT 0% (owner can’t set over 19.99%)

SMART CONTRACT REVIEW

Network	**Core DAO**
Contract Name	SatoshiSwapCore
Contract Address	0x5AACAbc9A29B9E5893f9F6A9ae7c8DFbCe4d8833
Deployer Address	0x2a95ae0163434f9FE47230562F560ff938900B3E
Contract Created	2023-04-04T08:48:48.000+0000
Verified CA	Yes
Compiler	v0.8.19+commit.7dd6d404
Optimization	Yes with runs
Sol License	MIT License
Other	evmVersion MIT License (MIT)

FEEs

Name	Value	address
DEV FEE	0%	0x1f353B28589c071eAD756342bE037363Bc61493
DEPOSIT	0%	0x1f353B28589c071eAD756342bE037363Bc61493

MANUAL CODE REVIEW

● Minor-risk

0 minor-risk code issue found

Could be fixed, and will not bring problems.

● Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

● High-Risk

0 high-risk code issues found

Must be fixed, and will bring problems.

● Infomational

1 Things you need to know.

1. Set fees

Risk Scenario: Setter can't set fees over 11%

```
uint private constant MAX_FEE_POINT = 1100;

function setFeePoint(uint _feePoint) external onlySetter {
        require(_feePoint < MAX_FEE_POINT, 'Hard fee limit exceed');
        feePoint = _feePoint;
        emit UpdateFeePoint(_feePoint);
    }
```

2. Withdraw Token

Risk Scenario: Setter can withdraw token

```
function transfer(address _token) external onlySetter {
        uint bal = IERC20(_token).balanceOf(address(this));
        IERC20(_token).safeTransfer(msg.sender, bal);
    }
```

3. Reset Price

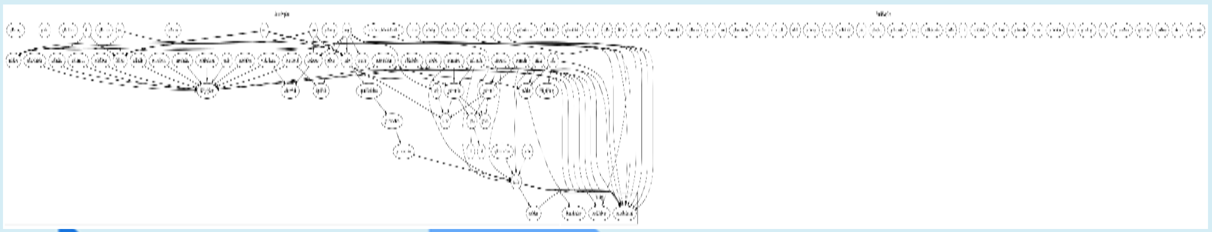
Risk Scenario: If address in badPairs, price in oracle will be reset to zero

```
function addBlacklist(address pairAddress) external onlySetter {
        badPairs.add(pairAddress);
    }

if (!badPairs.contains(pairAddress) && balance0 > minimumLiq[pair.token0] && balance1 > minimumLiq[pair.token1]) {
            if (pair.pairType == PairType.Stable) {
                // call getAmount only if _getAmountOut would not revert with tokenDecimals[pair.token0] input
                if (balance0 > tokenDecimals[pair.token0] * 2 && balance1 > tokenDecimals[pair.token1] * 2) {
                    // Using decimal balance for input amount, necessary liquidity is required
```

```
                      unchecked {
                          pairPrices[pairAddress].price0 = _getAmountOut(tokenDecimals[pair.token0], pair.token0, pairAddress) * 1e18 / tokenDecimals[pair.token0];
                          pairPrices[pairAddress].price1 = _getAmountOut(tokenDecimals[pair.token1], pair.token1, pairAddress) * 1e18 / tokenDecimals[pair.token1];
                      }
                  } else {
                      // Mark price with default due to low liq
                      unchecked {
                          pairPrices[pairAddress].price0 = tokenDecimals[pair.token1] * 1e18 / tokenDecimals[pair.token0];
                          pairPrices[pairAddress].price1 = tokenDecimals[pair.token0] * 1e18 / tokenDecimals[pair.token1];
                      }
                  }
              } else {
                  // * never overflows, and + overflow is desired
                  unchecked {
                      pairPrices[pairAddress].price0 = balance1 * 1e18 / balance0;
                      pairPrices[pairAddress].price1 = balance0 * 1e18 / balance1;
                  }
              }
          } else {
              // Reset price for pairs with low liq
              pairPrices[pairAddress].price0 = 0;
              pairPrices[pairAddress].price1 = 0;
          }
```

CONTRACT INHERITANCE



SATOSHISWAP - MULTICALL2

SMART CONTRACT REVIEW

Network	**Core DAO**
Contract Name	Multicall2
Contract Address	0xCdeCd2D5D33Dc6c361c3f05Bc862a95a77AA1D8d
Deployer Address	0x2a95ae0163434f9FE47230562F560ff938900B3E
Contract Created	2023-04-04T08:24:27.000+0000
Verified CA	Yes
Compiler	v0.8.19+commit.7dd6d404
Optimization	Yes with runs
Sol License	MIT License
Other	evmVersion MIT License (MIT)

MANUAL CODE REVIEW

● Minor-risk

2 minor-risk code issue found

Could be fixed, and will not bring problems.

1. A floating pragma is set. [SWC-103]

   Risk Scenario: The current pragma Solidity directive is "">=0.8.0""

   Risk Recommendation: 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.

```
pragma solidity ^0.8.0;
```

2. Potential use of "block.number" as source of randonmness. [SWC-120]

   Risk Scenario: The environment variable "block.number" looks like it might be used as a source of randomness.

   Risk Recommendation: 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.

```
blockNumber = block.number;
```

3. Potential use of "blockhash" as source of randonmness. [SWC-120]

   Risk Scenario: The environment variable "blockhash" looks like it might be used as a source of randomness.

   Risk Recommendation: 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.

```
blockHash = blockhash(blockNumber);
blockHash = blockhash(block.number - 1);
```

Medium-risk

1 medium-risk code issues found

Should be fixed, could bring problems.

1. Multiple calls are executed in the same transaction. [SWC-113]

   Risk Scenario: This call is executed following another call within the same transaction.

   Risk Recommendation: It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).

```
(bool success, bytes memory ret) =
calls[i].target.call(calls[i].callData);
```

High-Risk

0 high-risk code issues found

Must be fixed, and will bring problems.

● Infomational

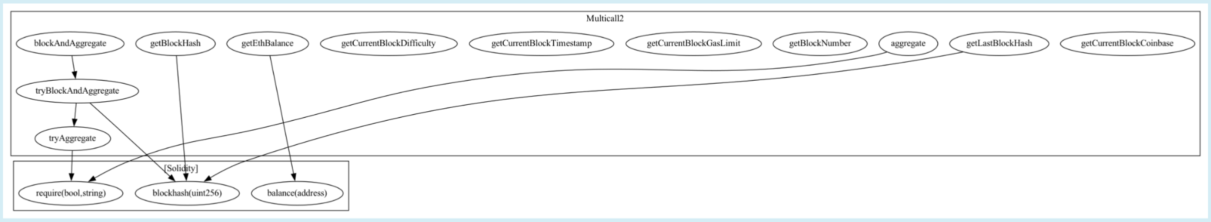
1 Things you need to know.

1. Read Balance

   Risk Scenario: For read eth balance

```
function getEthBalance(address addr) public view returns (uint256
balance) {
        balance = addr.balance;
    }
```

CONTRACT INHERITANCE



Note: Please check the disclaimer above and note that the audit makes no statements or warranties on the business model, investment attractiveness, or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by the project owner.