Coinsult

Advanced Manual Smart Contract Audit

Project: ShiD

Website: https://app.lianziyou.tech:800/

🟢 Low-Risk	🟡 Medium-Risk	🔴 High-Risk
8 low-risk code issues found	2 medium-risk code issues found	0 high-risk code issues found

Contract Address

Not deployed yet

Disclaimer

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Tokenomics

Not deployed yet

Source Code

Coinsult was comissioned by ShiD to perform an audit based on the following smart contract:

Not deployed yet

Note: This project uses openzeppelin imports. While we do check the full contract for vulnerabilities at the time of the audit, we can not ensure the correctness of these imported modules.

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

8 low-risk code
issues found

🟡 Medium-Risk

2 medium-risk code
issues found

🔴 High-Risk

0 high-risk code
issues found

The detailed report continues on the next page...

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Contract contains Reentrancy vulnerabilities

Additional information: This combination increases risk of malicious intent. While it may be justified by some complex mechanics (e.g. rebase, reflections, buyback).

More information: Slither

```
function _transfer(
    address sender,
    address recipient,
    uint256 amount
) internal override {
    if (_noBurnAddress[sender] || _noBurnAddress[recipient]) {
        super._transfer(sender, recipient, amount);
    } else {
        uint256 burnAmount = (amount * 1) / 100; //转账的时候1%黑洞
        super._burn(sender, burnAmount); // 1%销毁
        uint256 marketAmount = (amount * 1) / 100; //1%市场
        super._transfer(sender, marketWallet, marketAmount);

        // 普通转账时LP分红回流底池
        bool isNormalTransfer = !((sender == swapV2Pair ||
            sender == swapV2Router) ||
            (recipient == swapV2Pair || recipient == swapV2Router));
        if (isNormalTransfer &amp;&amp; uniswapV2Pair.totalSupply() &gt; 0) {
            uint256 rewardAmount = reserveLPAwardAmount;
            if (rewardAmount &gt; 0) {
                //分红转给池子
                reserveLPAwardAmount = reserveLPAwardAmount - rewardAmount;
```

Recommendation

Apply the check-effects-interactions pattern.

Exploit scenario

```
function withdrawBalance(){
    // send userBalance[msg.sender] Ether to msg.sender
    // if mgs.sender is a contract, it will call its fallback function
    if( ! (msg.sender.call.value(userBalance[msg.sender])() ) ){
        throw;
    }
    userBalance[msg.sender] = 0;
}
```

Bob uses the re-entrancy bug to call withdrawBalance two times, and withdraw more than its initial deposit to the contract.

● **Low-Risk:** Could be fixed, will not bring problems.

Avoid relying on block.timestamp

block.timestamp can be manipulated by miners.

```
uniswapV2Router02.swapExactTokensForTokensSupportingFeeOnTransferTokens(
    tokenAmount,
    0,
    path,
    address(this),
    block.timestamp
);
```

Recommendation

Do not use `block.timestamp`, now or `blockhash` as a source of randomness

Exploit scenario

```
contract Game {

    uint reward_determining_number;

    function guessing() external{
      reward_determining_number = uint256(block.blockhash(10000)) % 10;
    }
}
```

Eve is a miner. Eve calls `guessing` and re-orders the block containing the transaction. As a result, Eve wins the game.

● **Low-Risk:** Could be fixed, will not bring problems.

Too many digits

Literals with many digits are difficult to read and review.

```
processLP(500000); //最大gaslimt
```

Recommendation

Use: Ether suffix, Time suffix, or The scientific notation

Exploit scenario

```
contract MyContract{
    uint 1_ether = 10000000000000000000;
}
```

While `1_ether` looks like `1 ether`, it is `10 ether`. As a result, it’s likely to be used incorrectly.

● **Low-Risk:** Could be fixed, will not bring problems.

No zero address validation for some functions

Detect missing zero address validation.

```
//设置市场钱包
function setMarketWallet(address marketWallet_)
    public
    onlyRole(CONFIG_ADMIN)
{
    marketWallet = marketWallet_;
}
```

Recommendation

Check that the new address is not zero.

Exploit scenario

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

Bob calls `updateOwner` without specifying the `newOwner`, soBob loses ownership of the contract.

● **Low-Risk:** Could be fixed, will not bring problems.

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setMarketWallet(address marketWallet_)
    public
    onlyRole(CONFIG_ADMIN)
{
    marketWallet = marketWallet_;
}
```

Recommendation

Emit an event for critical parameter changes.

Exploit scenario

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

`updateOwner()` has no event, so it is difficult to track off-chain changes in the buy price.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Conformance to Solidity naming conventions

Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_setupRole(DEFAULT_ADMIN_ROLE, msg.sender); //设置默认管理员为合约创建者
_setupRole(CONFIG_ADMIN, msg.sender);
```

Recommendation

Follow the Solidity naming convention.

Rule exceptions

- Allow constant variable name/symbol/decimals to be lowercase (`ERC20`).
- Allow _ at the beginning of the `mixed_case` match for private variables and unused parameters.

● **Low-Risk:** Could be fixed, will not bring problems.

Redundant Statements

Detect the usage of redundant statements that have no effect.

```
address public immutable swapV2Pair;
address private immutable swapV2Router; //
address private immutable BUSD; //稳定币地址 usdt或BUSD
address private immutable WETH;
address private immutable shibAddress;
```

Recommendation

Remove redundant statements if they congest code but offer no value.

Exploit scenario

```
contract RedundantStatementsContract {

    constructor() public {
        uint; // Elementary Type Name
        bool; // Elementary Type Name
        RedundantStatementsContract; // Identifier
    }

    function test() public returns (uint) {
        uint; // Elementary Type Name
        assert; // Identifier
        test; // Identifier
        return 777;
    }
}
```

Each commented line references types/identifiers, but performs no action with them, so no code will be generated for such statements and they can be removed.

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Costly operations inside a loop

Costly operations inside a loop might waste gas, so optimizations are justified.

```
//最多只给列表完整分配一次，iterations &lt; shareholderCount
        while (gasUsed &lt; gas &amp;&amp; iterations = shareholderCount) {
                currentIndex = 0;
            }
            shareHolder = holderProviders[currentIndex];
            //持有的 LP 代币余额，LP 本身也是一种代币
            myTokenBalance = balanceOf(shareHolder);
            //不在排除列表，才分红
            if (
                myTokenBalance &gt; minHoldCoinAmount &amp;&amp;
                !excludeHolderProvider[shareHolder]
            ) {
                amount = (shibBalance * myTokenBalance) / totalSupply();
                //分红大于0进行分配,最小精度
                if (amount &gt; 0) {
                    shibToken.transfer(shareHolder, amount);
                }
            }
```

Recommendation

Use a local variable to hold the loop computation result.

Exploit scenario

```
contract CostlyOperationsInLoop{

    function bad() external{
        for (uint i=0; i < loop_count; i++){
            state_variable++;
        }
    }

    function good() external{
      uint local_variable = state_variable;
      for (uint i=0; i < loop_count; i++){
        local_variable++;
      }
      state_variable = local_variable;
    }
}
```

Incrementing `state_variable` in a loop incurs a lot of gas because of expensive `SSTORE`s, which might lead to an `out-of-gas`.

🟡 **Medium-Risk:** Should be fixed, could bring problems.

Use of chinese characters

```
uint256 public numTokensSellToFund; //合约卖币换shib条件阀值
uint256 public reserveLPAwardAmount; //记录LP分红回流底池的余额
uint256 public swapShibTokenBalance; //持币分红,兑换shib的本币余额
```

Recommendation

Only use utf-8 english character language

🟡 **Medium-Risk:** Should be fixed, could bring problems.

Empty functions

```
// 授权代币转账权限给对应的stake合约，那么授权的数量就是当前可以挖矿的数量
function approveAmountToStake(
    address erc20Address,
    address[] calldata toAddresss_,
    uint256[] calldata values_
) public virtual onlyOwner {}

function increaseAllowanceToStake(
    address erc20Address,
    address[] calldata toAddresss
```

Recommendation

Remove unused functions or give them a value

Owner privileges

- ● Owner cannot set fees higher than 25%
- ● Owner cannot pause trading
- ● Owner cannot change max transaction amount

Extra notes by the team

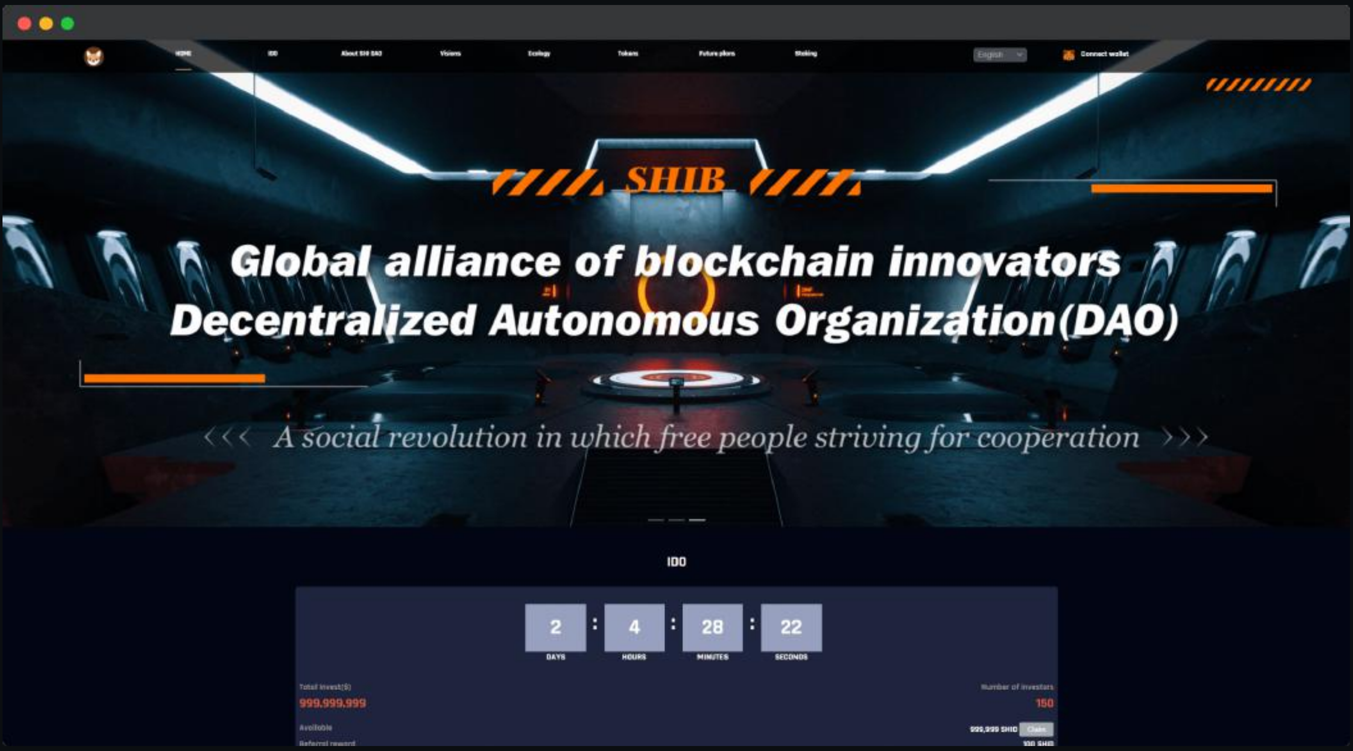
No notes

Contract Snapshot

```
contract ShiDToken is ERC20, AccessControl, IUserParent {
bytes32 public constant CONFIG_ADMIN = keccak256("CONFIG_ADMIN");
mapping(address =&gt; bool) public _noBurnAddress;
address _stakeAddress = address(0); // 质押合约地址
address public immutable swapV2Pair;
address private immutable swapV2Router; //
address private immutable BUSD; //稳定币地址 usdt或BUSD
address private immutable WETH;
address private immutable shibAddress;
address topAddress; // 推荐关系 顶层地址可以在没有上线的情况下进行关联关系
address public marketWallet; // 市场钱包
mapping(address =&gt; address) public _addressParentInfo; //保存推荐关系
IUniswapV2Router02 immutable uniswapV2Router02;
IUniswapV2Pair immutable uniswapV2Pair;
```

Website Review

Coinsult checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.


SHIB
Global alliance of blockchain innovators
Decentralized Autonomous Organization(DAO)
A social revolution in which free people striving for cooperation
IDO
2 DAYS
4 HOURS
28 MINUTES
22 SECONDS
999,999,999
150
Connect wallet

- ● Mobile Friendly
- ● Does not contain jQuery errors
- ● SSL Secured
- ● No major spelling errors

Project Overview

● Not KYC verified by Coinsult