Coinsult

Advanced Manual Smart Contract Audit

Project: Get Schiffy Token

Website: https://www.getschiffy.com/en

Low-Risk	Medium-Risk	High-Risk
6 low-risk code issues found	3 medium-risk code issues found	0 high-risk code issues found

Contract Address

0x4aa6027De5acf37eF2221Af51c5a8930E44e52F6

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

Rank	Address	Quantity (Token)	Percentage
1	0x407993575c91ce7643a4d4ccacc9a98c36ee1bbe	746,205,963,000	74.6206%
2	0xc00b613698ba19f82a6e4f47bc2ddae2dcfe4a7b	96,152,856,420.563205638160876875	9.6153%
3	0xd25a3afcec7ff51fa194e09788277918df263ef4	46,957,530,000	4.6958%
4	0xfe64d9b21bc114e6f1ccaef73ba519b327732713	8,364,030,112.2928361424099	0.8364%
5	0x12d0704441bc04c6b661685f70c51993716d9dbc	7,793,611,164.55380353149965	0.7794%

Source Code

Coinsult was comissioned by Get Schiffy Token to perform an audit based on the following smart contract:

https://bscscan.com/address/0x4aa6027De5acf37eF2221Af51c5a8930E44e52F6#code

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

6 low-risk code
issues found

🟡 Medium-Risk

3 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

```
External calls:
- uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2R
State variables written after the call(s):
- setExcludes() (#687)
- _isExcludedFromFee[owner()] = true (#693)
- _isExcludedFromFee[address(this)] = true (#694)
- _isExcludedFromFee[_playToEarnWallet] = true (#695)
- setExcludes() (#687)
- _isExcludedFromMaxTx[owner()] = true (#697)
- _isExcludedFromMaxTx[address(this)] = true (#698)
- _isExcludedFromMaxTx[_playToEarnWallet] = true (#699)
- uniswapV2Router = _uniswapV2Router (#685)
```

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.

```
//Locks the contract for owner for the amount of time provided
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = block.timestamp + time;
    emit OwnershipTransferred(_owner, address(0));
}

//Unlocks the contract for owner when _lockTime is exceeds
function unlock() public virtual {
    require(_previousOwner == msg.sender, "You don't have permission to unlock");
    require(block.timestamp &gt; _lockTime , "Contract is locked until 7 days");
    emit OwnershipTransferred(_owner, _previousOwner);
    _owner = _previousOwner;
}
```

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.

```
uint256 public swapThreshold = (_tTotal * 10) / 100000; // 0.01% of supply
```

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.

```
event UpdateFeeReceiver(address newwallet);
function updateAdminFeeReceirver(address newwallet) external onlyOwner {
    adminFeeReceiver = newwallet;
    emit UpdateFeeReceiver(newwallet);
}
```

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.

Conformance to Solidity naming conventions

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

```
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (#469) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (#470) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (#487) is not in mixedCase
Function IUniswapV2Router01.WETH() (#507) is not in mixedCase
Variable GetSchiffyGold.SWAPTOKEN (#667) is not in mixedCase
```

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.

Costly operations inside a loop

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

```
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already excluded");
    for (uint256 i = 0; i &lt; _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

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.

No check on lock time

```
//Locks the contract for owner for the amount of time provided
function lock(uint256 time) public virtual onlyOwner {
    _previousOwner = _owner;
    _owner = address(0);
    _lockTime = block.timestamp + time;
    emit OwnershipTransferred(_owner, address(0));
}
```

Recommendation

Since you rennounce the ownership of the contract for an X amount of time, you should check this value (x) to be sure you won't unintentionally lock it for too long / infinity and you won't be able to get ownership of the contract back.

✅ Acknowledged by Get Schiffy

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

Wrong event emitted

```
event ExcludeFromReward(address indexed account);
function excludeFromReward(address account) external onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    if(_rOwned[account] &gt; 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
    emit ExcludeFromMaxTx(account);
}
```

Recommendation

When you exclude from rewards, you emitted an excluded from max transaction event. Which is not correct.

✅ Comment by Get Schiffy:
This poses no risk to token holders at all.

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

Unchecked transfer, but event emitted

```
event ClaimedBUSD(address indexed to, uint256 amount);
function claimBusd() external onlyOwner {
    emit ClaimedBUSD(_msgSender(), SWAPTOKEN.balanceOf(address(this)));
    SWAPTOKEN.transfer(_msgSender(), SWAPTOKEN.balanceOf(address(this)));
}
```

Recommendation

First perform the transfer, then check successful transfer, then emit event.

✅ Comment by Get Schiffy

This poses no risk to token holders at all.

Owner privileges

- Owner cannot set fees higher than 25%
- Owner cannot pause trading
- Owner can change max transaction amount
- Owner can exclude from fees

Extra notes by the team

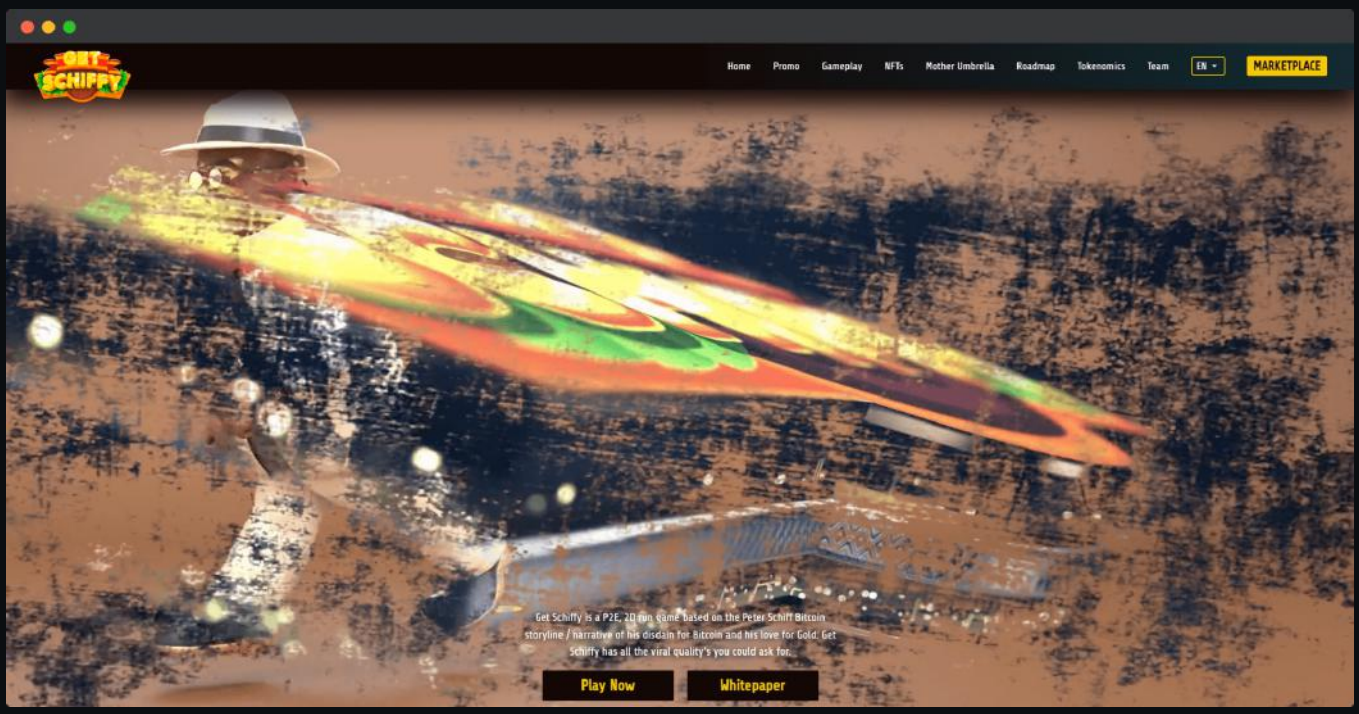
No notes

Contract Snapshot

```
contract GetSchiffyGold is Context, IGOLD, Ownable {
using SafeMath for uint256;
using Address for address;

mapping (address =&gt; uint256) private _rOwned;
mapping (address =&gt; uint256) private _tOwned;
mapping (address =&gt; mapping (address =&gt; uint256)) private _allowances;
mapping (address =&gt; bool) private _isExcludedFromFee;
mapping (address =&gt; bool) private _isExcluded;
mapping (address =&gt; bool) private _isExcludedFromMaxTx;

address[] private _excluded;
uint256 private constant MAX = ~uint256(0);
uint256 private _tTotal = 1 * 10**12 * 10**18;
uint256 private _rTotal = (MAX - (MAX % _tTotal));
string private _name;
string private _symbol;
uint8 private _decimals = 18;

uint256 private _playToEarn = 50;
uint256 private _previousPlayToEarn = _playToEarn;
uint256 private _taxFee = 25;
uint256 private _previousTaxFee = _taxFee;
uint256 private _tFeeTotal;

IUniswapV2Router02 public uniswapV2Router;
address public uniswapV2Pair;
IGOLD public SWAPTOKEN = IGOLD(0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56);
uint256 public _maxTxAmount;
uint256 public swapThreshold = (_tTotal * 10) / 100000; // 0.01% of supply
address public adminFeeReceiver;

constructor() {
    _name = "Get Schiffy Gold";
    _symbol = "GOLD";
    _maxTxAmount = _tTotal.mul(5).div(10**3);
    _rOwned[_msgSender()] = _rTotal;
```

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.



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

Project Overview

Not KYC verified by Coinsult

Get Schiffy Token

Audited by Coinsult.net

Date: 25 June 2022

Advanced Manual Smart Contract Audit