Coinsult

Advanced Manual Smart Contract Audit

Project: Get Schiffy Token

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

Low-Risk	Medium-Risk	High-Risk
5 low-risk code issues found	1 medium-risk code issues found	0 high-risk code issues found

Contract Address

0x194460B46315D4C61F52E5100560E62EB958D182

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

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	465,057,567,651.7247210291060875	46.5058%
2	0x99e9f4b484fb704f16ec56f7d0cb59e215b8ff3f	329,200,000,000	32.9200%
3	0x88809be08dff03509894b901c52d94289ad13868	48,506,294,498.9975	4.8506%
4	Null Address: 0x000...dEaD	42,600,000,000	4.2600%
5	0xfe64d9b21bc114e6f1ccaef73ba519b327732713	8,364,030,112.2928361424099	0.8364%

Source Code

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

https://bscscan.com/address/0x194460B46315D4C61F52E5100560E62EB958D182#code

Manual Code Review

In this audit report we will highlight all these issues:

Low-Risk

5 low-risk code
issues found

Medium-Risk

1 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.

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}

function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}

function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}

function setPlayToEarnFeePercent(uint256 liquidityFee) external onlyOwner() {
    _playToEarn = liquidityFee;
}

function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
    _maxTxAmount = _tTotal.mul(maxTxPercent).div(10**2);
}
```

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.

```
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (#471) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (#472) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (#489) is not in mixedCase
Function IUniswapV2Router01.WETH() (#509) is not in mixedCase
Parameter GetSchiffyGold.calculateTaxFee(uint256)._amount (#894) is not in mixedCase
Parameter GetSchiffyGold.calculatePlayToEarn(uint256)._amount (#898) is not in mixedCase
Variable GetSchiffyGold._maxTxAmount (#671) 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

Owner privileges

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

Extra notes by the team

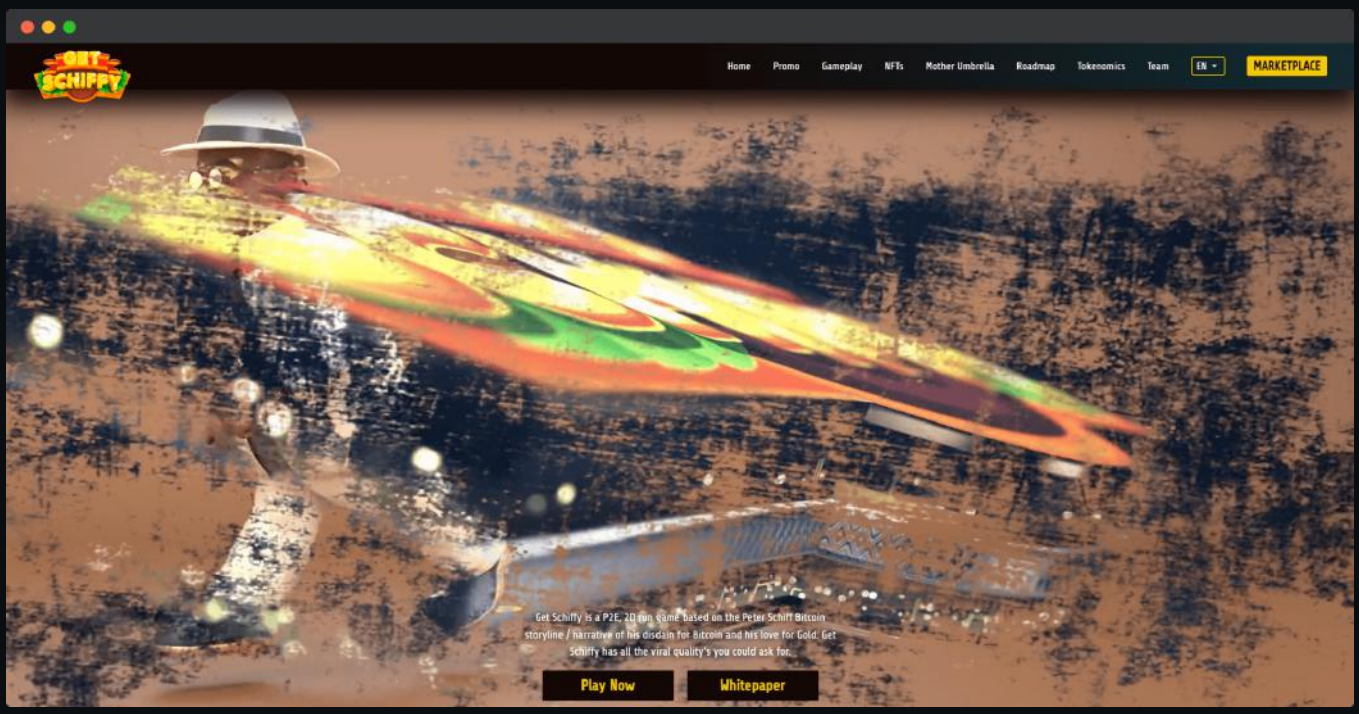
No notes

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));
uint256 private _tFeeTotal;
string private _name;
string private _symbol;
uint8 private _decimals = 18;

uint256 private _playToEarn = 75;
uint256 private _previousPlayToEarn = _playToEarn;
address payable private _playToEarnWallet;
uint256 private _taxFee = 0;
uint256 private _previousTaxFee = _taxFee;

IUniswapV2Router02 public immutable uniswapV2Router;
address public immutable uniswapV2Pair;

uint256 public _maxTxAmount;

constructor(address payable playToEarnWallet_, address dexRouter_)
{
    _name = "Get Schiffy Gold";
    _symbol = "GOLD";
    _playToEarnWallet = playToEarnWallet_;
    _maxTxAmount = _tTotal.mul(5).div(10**3);
```

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