Coinsult

Request your audit at coinsult.net

# Advanced Manual
# **Smart Contract Audit**

September 5, 2022

Audit requested by

Social2E

0xa05f9Fb902FE039E33E75347460bf3189024B98e

# Table of Contents

# Audit Summary

## Audit Scope

| | |
|---|---|
| **Project Name** | Social2E |
| **Website** | https://social2e.net/ |
| **Blockchain** | Binance Smart Chain |
| **Smart Contract Language** | Solidity |
| **Contract Address** | 0xa05f9Fb902FE039E33E75347460bf3189024B98e |
| **Audit Method** | Static Analysis, Manual Review |
| **Date of Audit** | 5 September 2022 |

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

## Tokenomics

| Rank | Address | Quantity (Token) | Percentage |
|---|---|---|---|
| 1 | 0x71a94fd00e001d0001f50e82171fe36fc4bbd298 | 1,000,000,000 | 100.0000% |

## Source Code

Coinsult was comissioned by Social2E to perform an audit based on the following code:

https://bscscan.com/address/0xa05f9Fb902FE039E33E75347460bf3189024B98e#code

# Disclaimer

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

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.

# Global Overview

## Manual Code Review

In this audit report we will highlight the following issues:

| Vulnerability Level | Total | Pending | Acknowledged | Resolved |
|---|---|---|---|---|
| Informational | 0 | 0 | 0 | 0 |
| Low-Risk | 4 | 4 | 0 | 0 |
| Medium-Risk | 0 | 0 | 0 | 0 |
| High-Risk | 0 | 0 | 0 | 0 |

## Privilege Overview

Coinsult checked the following privileges:

| Contract Privilege | Description |
|---|---|
| Owner can mint? | Owner cannot mint new tokens |
| Owner can blacklist? | Owner cannot blacklist addresses |
| Owner can set fees > 25%? | Owner cannot set the sell fee to 25% or higher |
| Owner can exclude from fees? | Owner can exclude from fees |
| Owner can pause trading? | Owner cannot pause the contract |
| Owner can set Max TX amount? | Owner cannot set max transaction amount |

More owner priviliges are listed later in the report.

🟢 **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 from,
        address to,
        uint256 amount
    ) internal  override {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");

        if(amount == 0) {
            super._transfer(from, to, 0);
            return;
        }

        if (launchedAt == 0 &amp;amp;&amp;amp; to == (uniswapV2Pair)) {
                launchedAt = block.timestamp;
        }

        if(antibotEnabled &amp;amp;&amp;amp; !isExcludedFromFees(from) &amp;amp;&amp;amp; !isExcludedFromFees(to)) {
             if(launchedAt + 1 hours &amp;gt; block.timestamp) {
                if(from == uniswapV2Pair) {
                    lastTransactionTime[to] = block.timestamp;
                } else {
```

### 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.

## Unchecked transfer

The return value of an external transfer/transferFrom call is not checked.

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "Owner cannot claim native tokens");
    if (token == address(0x0)) {
        payable(msg.sender).transfer(address(this).balance);
        return;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);
}
```

## Recommendation

Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked.

## Exploit scenario

```
contract Token {
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
}
contract MyBank{
    mapping(address => uint) balances;
    Token token;
    function deposit(uint amount) public{
        token.transferFrom(msg.sender, address(this), amount);
        balances[msg.sender] += amount;
    }
}
```

Several tokens do not revert in case of failure and return false. If one of these tokens is used in `MyBank`, `deposit` will not revert if the transfer fails, and an attacker can call `deposit` for free..

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

### Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner {
    require(
        newAmount &gt; totalSupply() / 1_000_000,
        "New Amount must more than 0.0001% of total supply"
    );
    swapTokensAtAmount = newAmount;
}
```

### 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.

### Redundant Statements

Detect the usage of redundant statements that have no effect.

```
function _msgData() internal view virtual returns (bytes calldata) {
    this; // silence state mutability warning without generating bytecode - see https://github.com/ether
    return msg.data;
}
```

### 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.

# Contract Privileges

## Maximum Fee Limit Check

Coinsult tests if the owner of the smart contract can set the transfer, buy or sell fee to 25% or more. It is bad practice to set the fees to 25% or more, because owners can prevent healthy trading or even stop trading when the fees are set too high.

| Type of fee | Description |
| --- | --- |
| Transfer fee | 🟢 Owner cannot set the transfer fee to 25% or higher |
| Buy fee | 🟢 Owner cannot set the buy fee to 25% or higher |
| Sell fee | 🟢 Owner cannot set the sell fee to 25% or higher |

Note: this is a boolean check to 25%, we will not change this value in the report.

# Contract Pausability Check

Coinsult tests if the owner of the smart contract has the ability to pause the contract. If this is the case, users can no longer interact with the smart contract; users can no longer trade the token.

| Privilege Check | Description |
| --- | --- |
| Can owner pause the contract? | Owner cannot pause the contract |

# Max Transaction Amount Check

Coinsult tests if the owner of the smart contract can set the maximum amount of a transaction. If the transaction exceeds this limit, the transaction will revert. Owners could prevent normal transactions to take place if they abuse this function.

| Privilege Check | Description |
| --- | --- |
| Can owner set max tx amount? | 🟢 Owner cannot set max transaction amount |

# Exclude From Fees Check

Coinsult tests if the owner of the smart contract can exclude addresses from paying tax fees. If the owner of the smart contract can exclude from fees, they could set high tax fees and exclude themselves from fees and benefit from 0% trading fees. However, some smart contracts require this function to exclude routers, dex, cex or other contracts / wallets from fees.

| Privilege Check | Description |
| --- | --- |
| Can owner exclude from fees? | Owner can exclude from fees |

# Ability To Mint Check

Coinsult tests if the owner of the smart contract can mint new tokens. If the contract contains a mint function, we refer to the token's total supply as non-fixed, allowing the token owner to "mint" more tokens whenever they want.

A mint function in the smart contract allows minting tokens at a later stage. A method to disable minting can also be added to stop the minting process irreversibly.

Minting tokens is done by sending a transaction that creates new tokens inside of the token smart contract. With the help of the smart contract function, an unlimited number of tokens can be created without spending additional energy or money.

| Privilege Check | Description |
| --- | --- |
| Can owner mint? | 🟢 Owner cannot mint new tokens |

# Ability To Blacklist Check

Coinsult tests if the owner of the smart contract can blacklist accounts from interacting with the smart contract. Blacklisting methods allow the contract owner to enter wallet addresses which are not allowed to interact with the smart contract.

This method can be abused by token owners to prevent certain / all holders from trading the token. However, blacklists might be good for tokens that want to rule out certain addresses from interacting with a smart contract.

| Privilege Check | Description |
| --- | --- |
| Can owner blacklist? | 🟢 Owner cannot blacklist addresses |

# Other Owner Privileges Check

Coinsult lists all important contract methods which the owner can interact with.

✅ No other important owner privileges to mention.

# Notes

## Notes by Social2E

No notes provided by the team.

## Notes by Coinsult

✅ No notes provided by Coinsult

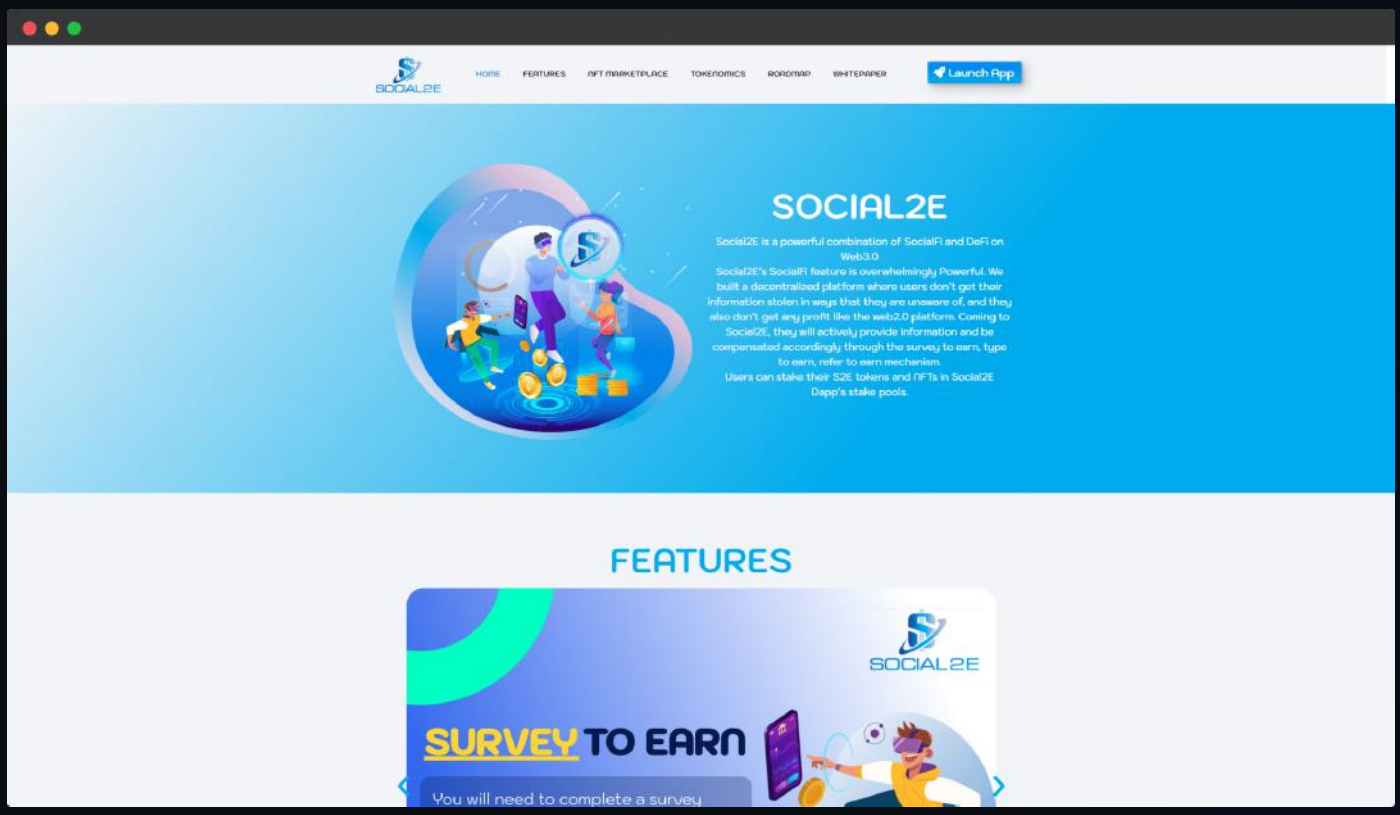
# Contract Snapshot

This is how the constructor of the contract looked at the time of auditing the smart contract.

```
contract Social2E is ERC20, Ownable {
uint256 public liquidityFeeOnBuy  = 0;
uint256 public liquidityFeeOnSell = 0;

uint256 public taxDuringAntiBot   = 5;

IUniswapV2Router02 public uniswapV2Router;
address public  uniswapV2Pair;

address private DEAD = 0x000000000000000000000000000000000000dEaD;
```

# 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.




| Type of check | Description |
| --- | --- |
| Mobile friendly? | The website is mobile friendly |
| Contains jQuery errors? | The website does not contain jQuery errors |
| Is SSL secured? | The website is SSL secured |
| Contains spelling errors? | The website does not contain spelling errors |

# Certificate of Proof

- KYC verified by Coinsult partner
- Not KYC verified by Coinsult

## Social2E

### Completed KYC Verification at a Coinsult partner

✔ Project Owner Identified

✔ Contract: 0xa05f9Fb902FE039E33E75347460bf3189024B98e

## Social2E

### Audited by Coinsult.net

Date: 5 September 2022

✔ Advanced Manual Smart Contract Audit

Coinsult

coinsult.net

End of report

# Smart Contract Audit