Audit Report for **SAFUU**

Date: 06 August 2024

Audit result: **Passed**

**Token Address:** 0xC53Eb9eCc247465e2E1fb16B2EdE3A6446053170

**Name:** SAFUU 2.0

**Symbol:** SAFUU

**Decimals**: 18

**Network:** Ethereum Network

**Token Type**: ERC-20

**Owner**: 0x5fA28b1227432FA1BC4bd88073f37063381c04dC

**Deployer:** 0x5fA28b1227432FA1BC4bd88073f37063381c04dC

**Token Supply:** 25,890,650

**Checksum:** 76d9051a5c26a4a92f7159096df9df6d

**Testnet:**

<https://testnet.bscscan.com/address/0xfcf8e661354c81615f5ceaaadcc8dbfcab41eaa5#writeContract>

**Token Overview:**

**Buy tax:** 1% - 10%

**Sell tax:** 1% - 10%

**Transfer Fee:** 0-0%

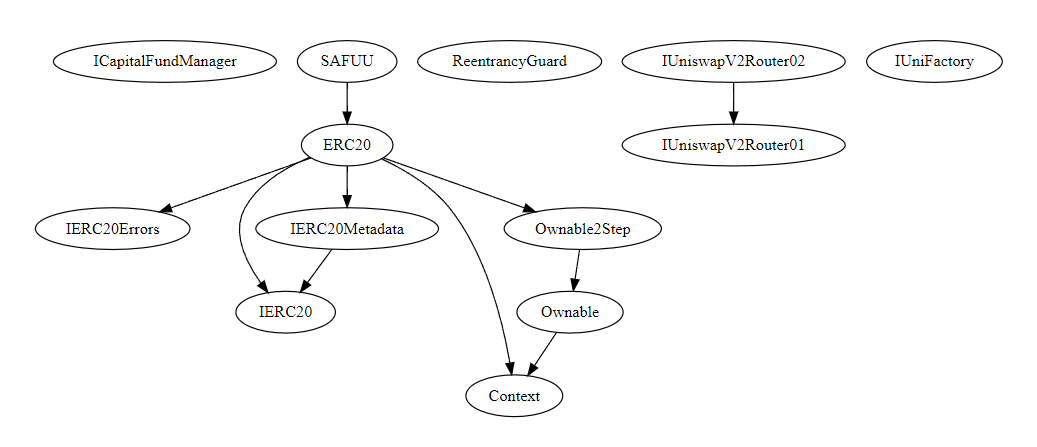
**Fee Privilege:** Owner

**Ownership:** Owned

**Minting:** None

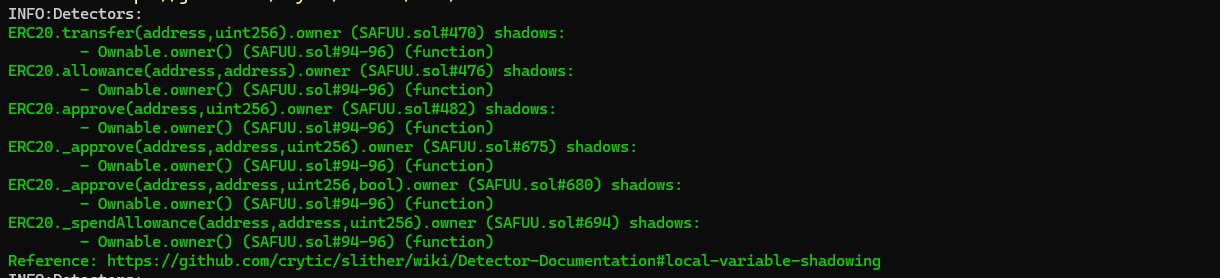
**Max Tx:** No

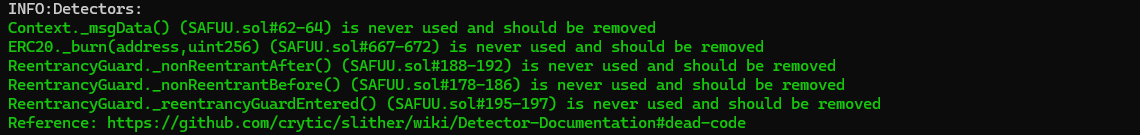
**Blacklist:** No

**Inheritance Tree**  ****

**Static Analysis**

A static analysis of the code was performed using Slither. No issues were found.




**Functional Tests**

**Router (PCS V2):**

**1- Set Fees (passed):**

<https://testnet.bscscan.com/tx/0x44f38c527fdbaabe44689aa2fc07fddc506158ea0bdc18a19720ef8685f3951b>

**2- Set Staking (passed):**

<https://testnet.bscscan.com/tx/0xb6b6c5595ca242dac9e296de2ae99ef32887001593d24171d2b0d18a2054cc33>

**3- Set Treasury Address (passed):**

<https://testnet.bscscan.com/tx/0x5eacbd5dc4fedadd6c98ce88731d9a33149b831c2e746b506bbe87ccfda9313d>

**4- Set Capital Fund (passed):**

<https://testnet.bscscan.com/tx/0x870691de2d8617ce9d7c20afcb2b664ef26defcd364a9273c246b4c365d47ef9>

**Ownership Privileges:**

- The owner can transfer/renounce the ownership.

- The owner can set treasury/staking/capital fund address.

- The owner can set MinTokens for Swap greater than 0.

- The owner can set a buy and sell tax not to more than 10%

- The owner can setFeeDistribution.

- The owner can rescue ETH.

- The owner can remove/add a pair address.

- The owner can setFeeExempt.

- The owner can setTimeExpire.

**Findings:**

**Critical**: 0

**High**: 0

**Medium**: 0

**Low**: 5

**Informational** **&** **Optimizations**: 1

**Centralization – The owner can regain ownership.**

**Severity**: **Low**

**Function**: pendingOwner

**Status:** Open

**Overview:**

The owner can regain ownership after transferring it with the following steps:

1. Call lock function to set previous owner to the own address

2. Call unlock function to get ownership back

3. Transfer/renounce ownership

4. Call unlock function to get ownership back

abstract contract Ownable2Step is Ownable {

    address private \_pendingOwner;

    event OwnershipTransferStarted(address indexed previousOwner, address indexed newOwner);

    function pendingOwner() public view virtual returns (address) {

        return \_pendingOwner;

    }

    function transferOwnership(address newOwner) public virtual override onlyOwner {

        \_pendingOwner = newOwner;

        emit OwnershipTransferStarted(owner(), newOwner);

    }

    function \_transferOwnership(address newOwner) internal virtual override {

        delete \_pendingOwner;

        super.\_transferOwnership(newOwner);

    }

    function acceptOwnership() public virtual {

        address sender = \_msgSender();

        if (pendingOwner() != sender) {

            revert OwnableUnauthorizedAccount(sender);

        }

        \_transferOwnership(sender);

    }

}

**Suggestion:** Make sure to set the previous ownership back to address zero after using the unlock function.

**Centralization – Missing Require Check.**

**Severity**: **Low**

**Function**: setStaking,setCapitalFund, setTreasuryAddress.

**Status:** Open

**Overview:**

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner will set the address to the contract address, then the Eth will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

function setTreasuryAddress(address \_newTreasuryWallet) external onlyOwner {

        require(\_newTreasuryWallet != address(0), "Zero Wallets");

        TREASURY = \_newTreasuryWallet;

    }

function setStaking(address pAddress) external onlyOwner {

        STAKING = pAddress;//can be set to zero to deactivate sending to Staking

    }

    function setCapitalFund(address pAddress) external onlyOwner {

        CAPITAL\_FUND\_ADDRESS = pAddress;

        if(CAPITAL\_FUND\_ADDRESS != address(0)) CAPITAL\_FUND = ICapitalFundManager(CAPITAL\_FUND\_ADDRESS);//we allow the address to be address(0) so you can deactivate the CAPITAL\_FUND deposit

    }

**Suggestion:** It is recommended that the address should not be able to set as a contract address.  
  
**Centralization – Missing Events**

**Severity**: **Low**

**Subject**: Missing Events

**Status:** Open

**Overview:**

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

function setTreasuryAddress(address \_newTreasuryWallet) external onlyOwner {

        require(\_newTreasuryWallet != address(0), "Zero Wallets");

        TREASURY = \_newTreasuryWallet;

    }

function setMinTokensForSwap(uint256 \_newMinTokens) external onlyOwner {

        require(\_newMinTokens > 0, "New minimum tokens must be greater than 0");

        MIN\_TOKENS\_FOR\_SWAP = \_newMinTokens;

    }

function setFees(uint256 \_BUYTax, uint256 \_SELLTax) external onlyOwner {

        require(

           \_BUYTax >= MIN\_FEE\_RATE && \_BUYTax <= MAX\_FEE\_RATE &&

           \_SELLTax >= MIN\_FEE\_RATE && \_SELLTax <= MAX\_FEE\_RATE,

           "MAX\_FEE\_RATE or MIN\_FEE\_RATE not matched"

        );

        BUYTax          = \_BUYTax;

        SELLTax         = \_SELLTax;

    }

function setFeeDistribution(uint256 \_TREASURYPart, uint256 \_SHERIFFPart, uint256 \_BurnPart) external onlyOwner {

        require(

            (\_TREASURYPart + \_SHERIFFPart + \_BurnPart) == 1000, "3 sums not matched"

        );

        TREASURYPart    = \_TREASURYPart;

        SHERIFFPart     = \_SHERIFFPart;

        BurnPart        = \_BurnPart;

    }

**Suggestion:**

Emit an event for critical changes.

**Centralization – Local variable Shadowing**

**Severity**: **Low**

**Subject**: Variable Shadowing

**Status:** Open

**Overview:**

function transfer(address to, uint256 value) public virtual returns (bool) {

        address owner = \_msgSender();

        \_transfer(owner, to, value);

        return true;

    }

function allowance(address owner, address spender) public view virtual returns (uint256) {

        return \_allowances[owner][spender];

    }

    function approve(address spender, uint256 value) public virtual returns (bool) {

        address owner = \_msgSender();

        \_approve(owner, spender, value);

        return true;

    }

**Suggestion:**

Rename the local variables that shadow another component.

**Centralization – Missing Zero Address**

**Severity**: **Low**

**Subject**: Zero Check

**Status:** Open

**Overview:**

Functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

function setStaking(address pAddress) external onlyOwner {

        STAKING = pAddress;//can be set to zero to deactivate sending to Staking

    }

**Suggestion:**

It is suggested that the address should not be zero or dead.

**Optimization**

**Severity**: **Optimization**

**Subject**: Remove unused code.

**Status:** Open

**Overview:**

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice. though to avoid them.

function \_burn(address account, uint256 value) internal {

        if (account == address(0)) {

            revert ERC20InvalidSender(address(0));

        }

        \_simpleUpdate(account, address(0), value);

    }