freshcoins

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Fides Token

$FIDES

21/02/2022

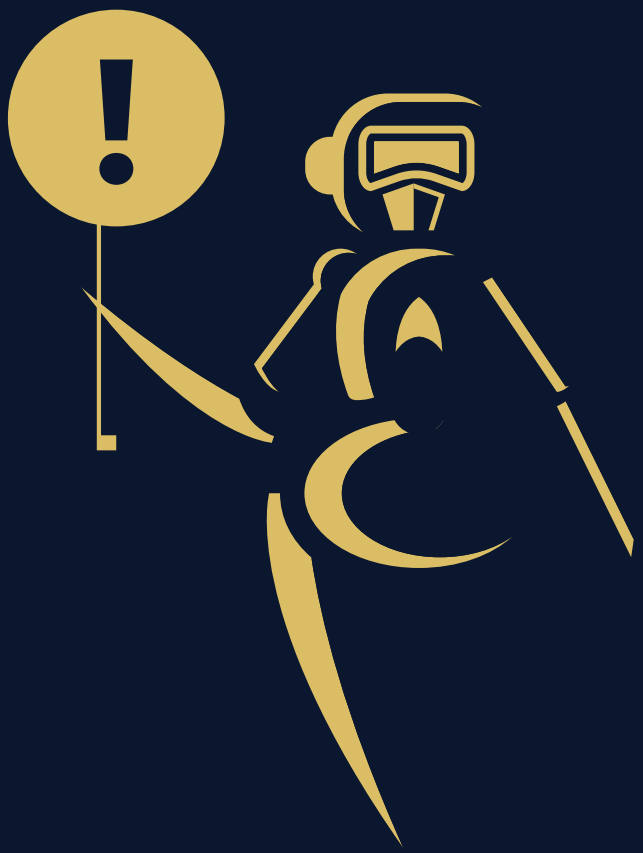
TABLE OF CONTENTS

DISCLAIMER

The information provided on this analysis document is only for general information and should not be used as a reason to invest.

FreshCoins Team will take no payment for manipulating the results of this audit.

The score and the result will stay on this project page information on our website https://freshcoins.io
FreshCoins Team does not guarantees that a project will not sell off team supply, or any other scam strategy (RUG or Honeypot etc)



INTRODUCTION

FreshCoins (Consultant) was contracted by
Fides Token (Customer) to conduct a Smart Contract Code Review
and Security Analysis.

0xb876567c9212b4dcf1bfC3B9b55Ac6931De3aa41

Network: Binance Smart Chain (BNB)

This report presents the findings of the security assessment of
Customer's smart contract and its code review conducted on 21/02/2022

WEBSITE DIAGNOSTIC

https://www.fidestoken.com/

0-49 50-89 90-100

Performance	Accessibility	Best Practices	SEO	Progressive Web App
77	90	93	98	NA

Metrics

First Contentful Paint
3.2 s

Time to interactive
48.2 s

Speed Index
24.4 s

Total Blocking Time
3,104 ms

Large Contentful Paint
25.1 s

Cumulative Layout Shift
0

WEBSITE IMPROVEMENTS

Use video formats for animated content

Reduce initial server response time

Reduce unused CSS

Reduce unused JavaScript

Reduce the impact of third-party code Third-party code blocked the main thread for 6k ms

Avoid enormous network payloads Total size was 11,587 KiB

Image elements do not have explicit width and height

Ensure text remains visible during webfont load

Reduce JavaScript execution time 10.8 s

Buttons do not have an accessible name

Background and foreground colors do not have a sufficient contrast ratio

Some elements have a [tabindex] value greater than 0

Heading elements are not in a sequentially-descending order

Browser errors were logged to the console

AUDIT OVERVIEW

96

Security Score

94

Static Scan
Automatic scanning for common vulnerabilities

98

ERC Scan
Automatic checks for ERC's conformance

0 High

0 Medium

0 Low

0 Optimizations

0 Informational

No.	Issue description	Checking Status
1	Compiler Errors / Warnings	Passed
2	Reentrancy and Cross-function	Passed
3	Front running	Passed
4	Timestamp dependence	Passed
5	Integer Overflow and Underflow	Passed
6	Reverted DoS	Passed
7	DoS with block gas limit	Passed
8	Methods execution permissions	Passed
9	Exchange rate impact	Passed
10	Malicious Event	Passed
11	Scoping and Declarations	Passed
12	Uninitialized storage pointers	Passed
13	Design Logic	Passed
14	Safe Zeppelin module	Passed

OWNER PRIVILEGES

Contract owner can't exclude an address from transactions.

Contract owner can't mint tokens after initial contract deploy

Contract owner can exclude/include wallet from fees

```
function setIsFeeExempt(address holder, bool exempt) external authorized {
        isFeeExempt[holder] = exempt;
}
```

Contract owner can exclude/include wallet from dividends

```
function setIsDividendExempt(address holder, bool exempt)
        external
        authorized
    {
        require(holder != address(this) && holder != pair);
        isDividendExempt[holder] = exempt;
        if (exempt) {
            distributor.setShare(holder, 0);
        } else {
            distributor.setShare(holder, _balances[holder]);
        }
}
```

Contract owner can change swapback settings

```
function setSwapBackSettings(bool _enabled, uint256 _amount)
        external
        authorized
    {
        swapEnabled = _enabled;
        swapThreshold = _amount;
}
```

Contract owner can change distribution settings

```
function setDistributionCriteria(
        uint256 _minPeriod,
        uint256 _minDistribution
    ) external authorized {
        distributor.setDistributionCriteria(_minPeriod, _minDistribution);
}

function setDistributorSettings(uint256 gas) external authorized {
        require(gas < 750000, "Gas must be lower than 750000");
        distributorGas = gas;
}
```

Contract owner can change autoLiquidityReceiver and marketingFeeReceiver addresses

Current value:

autoLiquidityReceiver : 0x625f30563989857cd592566723ba5dffbed08b37

marketingFeeReceiver : 0x625f30563989857cd592566723ba5dffbed08b37

```
function setFeeReceivers(
        address _autoLiquidityReceiver,
        address _marketingFeeReceiver
    ) external authorized {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        marketingFeeReceiver = _marketingFeeReceiver;
}
```

Contract owner can change buyback settings

```
function setAutoBuybackSettings(
        bool _enabled,
        uint256 _cap,
        uint256 _amount,
        uint256 _period
    ) external authorized {
        autoBuybackEnabled = _enabled;
        autoBuybackCap = _cap;
        autoBuybackAccumulator = 0;
        autoBuybackAmount = _amount;
        autoBuybackBlockPeriod = _period;
        autoBuybackBlockLast = block.number;
}

function setBuybackMultiplierSettings(
        uint256 numerator,
        uint256 denominator,
        uint256 length
    ) external authorized {
        require(numerator / denominator <= 2 && numerator > denominator);
        buybackMultiplierNumerator = numerator;
        buybackMultiplierDenominator = denominator;
        buybackMultiplierLength = length;
}

function setBuyBacker(address acc, bool add) external authorized {
        buyBacker[acc] = add;
}
```

Contract owner can transfer ownership

```
function transferOwnership(address payable adr) public onlyOwner {
        owner = adr;
        authorizations[adr] = true;
        emit OwnershipTransferred(adr);
}
```

Contract owner can change the fees up to 25%

```
function _initializeFees(uint256[5] memory feeSettings_) internal {
        _setFees(
            feeSettings_[0], // liquidityFee
            feeSettings_[1], // buybackFee
            feeSettings_[2], // reflectionFee
            feeSettings_[3], // marketingFee
            feeSettings_[4] // feeDenominator
        );
    }

    function _setFees(
        uint256 _liquidityFee,
        uint256 _buybackFee,
        uint256 _reflectionFee,
        uint256 _marketingFee,
        uint256 _feeDenominator
    ) internal {
        liquidityFee = _liquidityFee;
        buybackFee = _buybackFee;
        reflectionFee = _reflectionFee;
        marketingFee = _marketingFee;
        totalFee = _liquidityFee.add(_buybackFee).add(_reflectionFee).add(
            _marketingFee
        );
        feeDenominator = _feeDenominator;
        require(
            totalFee < feeDenominator / 4,
            "Total fee should not be greater than 1/4 of fee denominator"
        );
    }
```

Recommendation:

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. The risk can be prevented by temporarily locking the contract or renouncing ownership.

CONCLUSION AND ANALYSIS

Smart Contracts within the scope were manually reviewed and analyzed with static tools.

Audit report overview contains all found security vulnerabilities and other issues in the reviewed code.

Found no issue during the first review.

TOKEN DETAILS

Details

Buy fees:	10%
Sell fees:	10%
Max TX:	N/A
Max Sell:	N/A

Honeypot Risk

Ownership:	Owned
Blacklist:	Not detected
Modify Max TX:	Not detected
Modify Max Sell:	Not detected
Disable Trading:	Not detected

Rug Pull Risk

Liquidity:	N/A
Holders:	Clean

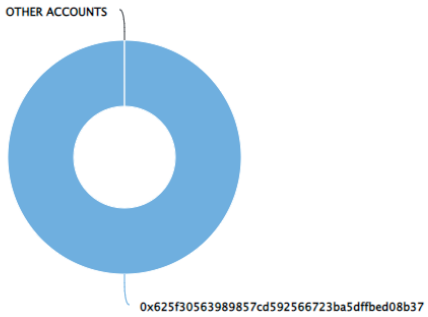
FIDES TOKEN ANALYTICS & TOP 10 TOKEN HOLDERS

The top 10 holders collectively own 100.00% (1,000,000,000,000.00 Tokens) of Fides Token

Token Total Supply: 1,000,000,000,000.00 Token | Total Token Holders: 1

Fides Token Top 10 Token Holders

Source: BscScan.com



(A total of 1,000,000,000,000.00 tokens held by the top 10 accounts from the total supply of 1,000,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x625f30563989857cd592566723ba5dffbed08b37	1,000,000,000,000	100.0000%

TECHNICAL DISCLAIMER

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. The audit can't guarantee the explicit security of the audited project / smart contract.