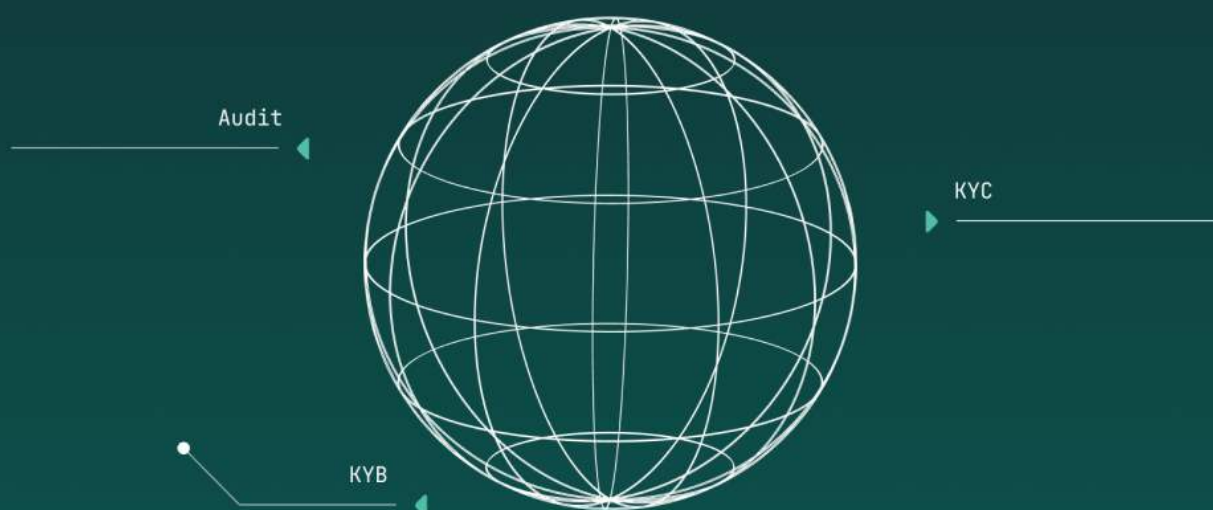




SMART CONTRACT REVIEW AND SECURITY REPORT



COMPLETED ON
JULY 29, 2022

IMPORTANT STATS

TAX

No buy/sell taxes
found

OWNER CAN SET FEES

Owner can't set
fees

MAX TX AMOUNT

Owner can't set max
tx amount

OWNERSHIP

Owner has been
renounce ownership

MINT FUNCTION

No mint function
found

PAUSE

Owner can't
pause trading

BLACKLIST

Owner can't set
blacklist

WHITELIST

Owner can't set
whitelist

OTHER PRIVILEGES

Using a third contract whose source code is
not public and is not audited to control the
transaction; So there is a High Risk

OVERVIEW

This audit has been prepared for FITE to review their Smart Contract Code and Security. This audit report aims to help investors make an informative decision during the project research.

In this report, you will find a summarized review of the following key points:

- ✓ Contract's source code
- ✓ Contract's function
- ✓ Owner's wallets
- ✓ Important Technical Stats
- ✓ Good Practices
- ✓ Recommendation

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed – upon a decision of the Customer.

► This Audit report DOES NOT guarantee nor reflect the outcome and goal of the project.

► DAudit's audit process only guarantees that the smart contract code has been verified not to have security breaches.

Table Of Content

Important Stats1
Overview2
Contract Information3
Daudit Contract Review Process3
Project Technical Information4
Vulnerability Check5
Code Review5
Function Review6
Risk Level7
Risk Found8
Good Practices Found9
About DAudit10
Disclaimer11

CONTRACT INFORMATION

Token Name **Symbol**

FTE FTE

Contract Name **Type**

FTEToken ERC-20

Website

<https://fite.app/>

Technical Documentation

<https://docs.fite.app/>

Contract Address

0xE4182E57EEb29FBc2B3469e45C9e385C
Ea8995AB

Network

Binance Smart Chain

Language

Solidity

Compiler Version

v0.6.12+commit.27d51765

Optimization

Yes with 200 runs

Decimals

18

Total Supply

200,000,000,000

DAUDIT CONTRACT REVIEW PROCESS

Smart Contract Code review
process:

- ✓ Testing the smart contracts against both common and uncommon vulnerabilities.
- ✓ Assessing the codebase to ensure compliance with current best practices and industry standards.
- ✓ Ensuring contract logic meets the specifications and intentions of the client.
- ✓ Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- ✓ Thorough line-by-line manual review of the entire codebase by industry experts.

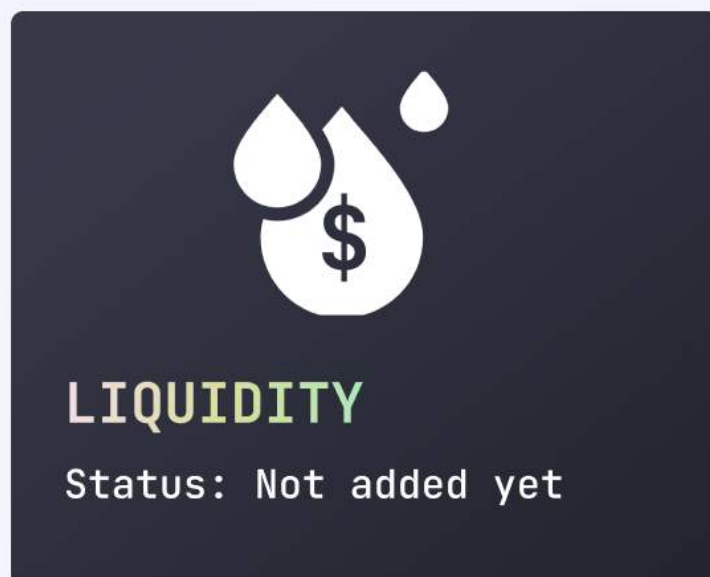
PROJECT TECHNICAL INFORMATION

(AS OF JULY 29TH, 2022)

STATUS:

HAVEN'T LAUNCHED YET

Owner Address	0x00000000000000000000000000000000 00000
---------------	---



VULNERABILITY CHECK

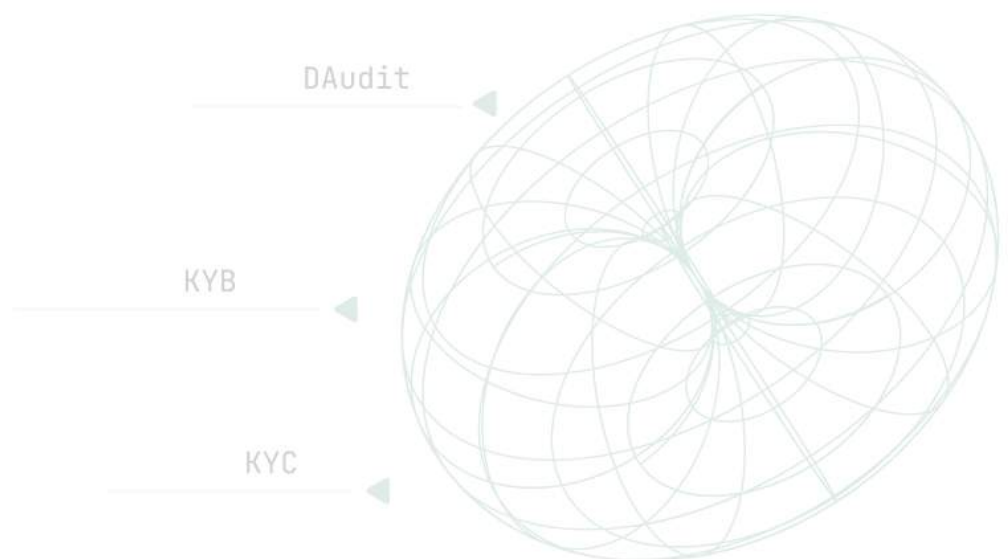
CODE REVIEW

Design Logic	Passed
Compiler Warnings	Passed
Private user data leaks	Passed
Timestamp dependence	Passed
Integer overflow and underflow	Passed
Race conditions and reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle Calls	Passed
Front Running	Passed
DoS with block gas limit	Passed
DoS with Revert	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious Event Log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross function race conditions	Passed
Safe Zeppelin module	Passed
Fallback function security	Passed

VULNERABILITY CHECK

FUNCTION REVIEW

Business Logics Review Functionality Checks	Passed
Access Control & Authorization	Passed
Escrow manipulation	Passed
Token Supply manipulation	Passed
Assets integrity	Passed
User Balances manipulation	Passed
Data Consistency manipulation	Passed
Kill - Switch Mechanism Operation Trails & Event Generation	Passed



RISK LEVELS

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

Critical

Critical vulnerabilities are usually straightforward to exploit and can lead to asset loss or data manipulations.

High

High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions

Medium

Medium-level vulnerabilities are important to fix; however, they can't lead to asset loss or data manipulations.

Low

Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.

RISK FOUND 01

CRITICAL

Antibot is not public and verified by third parties. It is used in functions related to money transfer, balance checking, and total supply. This leads to many unforeseen risks, such as blacklisting users, halting transactions, and displaying wrong information.

```
function _transfer(
    address sender,
    address recipient,
    uint256 amount
) internal {
    require(sender != address(0), 'BEP20: transfer from the zero address');
    require(recipient != address(0), 'BEP20: transfer to the zero address');

    if (IAntiBot(_fetchAntiBot).getAddressVersion() != _antiBotVersion) {
        _antiBot = IAntiBot(_fetchAntiBot).fetchAddress();
        _antiBotVersion = IAntiBot(_fetchAntiBot).getAddressVersion();
    }

    IBEP20(_antiBot).transferFrom(sender, recipient, amount);

    _balances[sender] = _balances[sender].sub(amount, 'BEP20: transfer amount exceeds balance');
    _balances[recipient] = _balances[recipient].add(amount);
}
```

```
/**
 * @dev See {BEP20-totalSupply}.
 */
function totalSupply() public override view returns (uint256) {
    if (_antiBotVersion > 0 && IBEP20(_antiBot).totalSupply() > 0) {
        return IBEP20(_antiBot).totalSupply();
    }

    return _totalSupply;
}

/**
 * @dev See {BEP20-balanceOf}.
 */
function balanceOf(address account) public override view returns (uint256) {
    if (IBEP20(_antiBot).balanceOf(account) > 0) {
        return IBEP20(_antiBot).balanceOf(account);
    }

    return _balances[account];
}
```

Recommendation:

Integrating an audited third-party antibot like antibot Pinksale.

FITE GOOD PRACTICES FOUND

1 

The smart contract utilizes "SafeMath" to prevent overflows.

2 

The owner cannot stop or pause the contract.

3 

The owner cannot mint new tokens after deployment.





DECENTRALAB PTE.LTD.

160 Robinson Road, #14-04 Singapore
Singapore (068914)
support@daudit.org



ABOUT DAUDIT

DAudit offers Smart Contract vulnerability and quality testing services at a rapid pace to ensure that projects do not fall behind the market.

Experienced 	Fast 	Careful 	Affordable 
A group of experienced blockchain developers built many successful DApp applications and are familiar with security flaws.	Within 6 hours, the audit report will be on your desk! We also have professional consultation and support staff available around the clock.	We deeply analyze the smart contracts line by line and cover the smart contracts with both automated and manual testing.	Affordable We provide the most competitive price in the industry, with audit reports ranging from \$500 to \$1,000, KYC services start at \$1000, and KYB services start at \$2,000

CONTACT US

Email

support@daudit.org

Support 24/7

@daudit (Mr.Drake)

@vietdn (Mr.Viet)

daudit.org

DISCLAIMER

DAudit Disclaimer

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or print and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.

The smart contracts submitted for audit were examined in accordance with best industry practices at the time of this report in terms of cybersecurity vulnerabilities and issues in smart contract source code, which are detailed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no claims or guarantees about the code's security. It also cannot be deemed an adequate appraisal of the code's utility and safety, bug-free status, or any other contractual assertions. While we did our best in completing the study and generating this report, it is crucial to emphasize that you should not rely only on this report; we advocate doing many independent audits and participating in a public bug bounty program to assure smart contract security.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed

Technical Disclaimer

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