AuditAce

FROM INCEPTION TO SUCCESS

SAFEFTT

Smart Contract Audit Report

ABOUT AUDITACE

Audit Ace is built, to combat financial fraud in the cryptocurrency industry, a growing security firm that provides audits, Smart contract creation, and end-to-end solutions to all crypto-related queries.

Website - https://auditace.tech/

Telegram - https://t.me/Audit_Ace

Twitter - https://twitter.com/auditace_

Github - https://github.com/Audit-Ace

Overview

AUDITACE team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks.

Audit Result: Failed

Audit Date: November 16, 2022

KYC: Not Done

Audit Team: TEAM AUDITACE

Reason for Failure : Contract contains serious high centralisation risk (ability to Mint, set upto 100% Tax, Blacklist any wallet, Etc.) which may severely effect and damage investor funds.

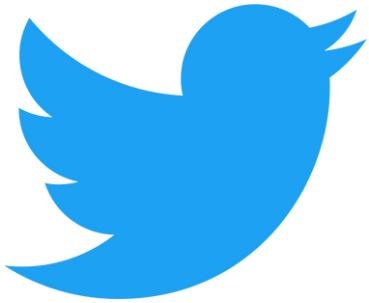
Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

SAFEFTT

Social Media Overview

https://t.me/safefttglobal



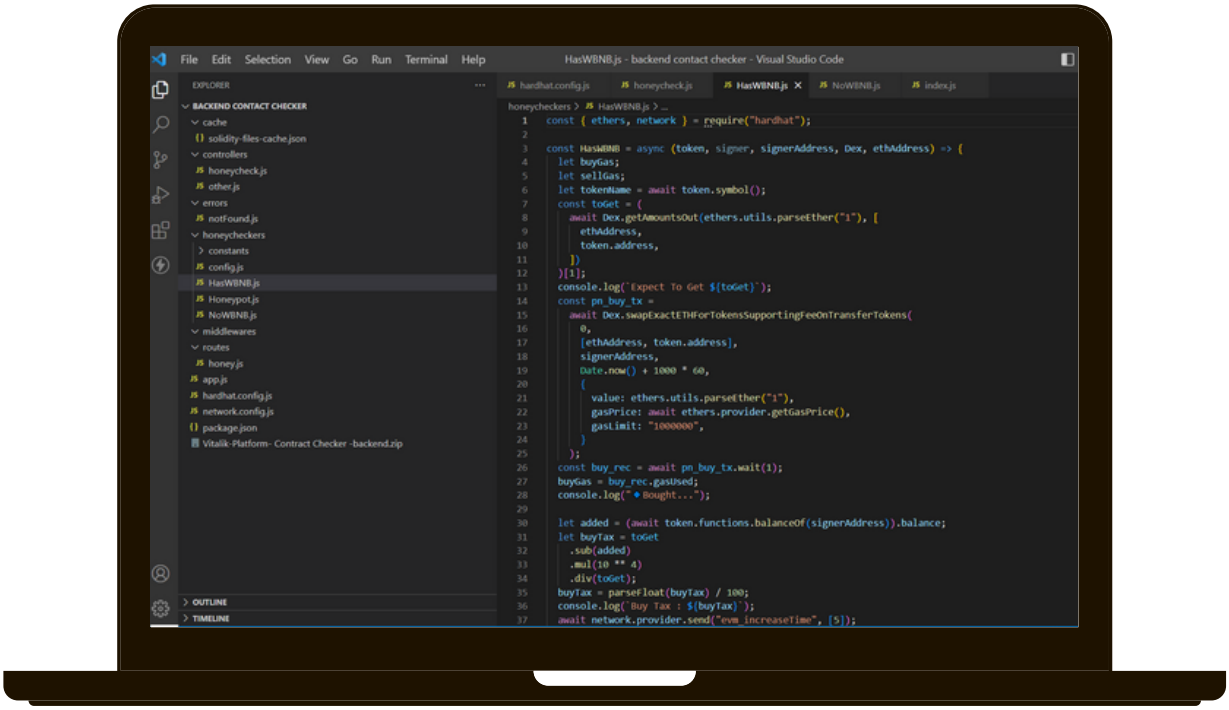
https://twitter.com/SafeFTT

https://www.safeftt.com/

Token Summary

Parameter	Result
Address	0x85b9b3BCE409ba2745791b016894A526621709dA
Token Type	BEP 20
Contract Checksum	adbc2efbbe71619971ee81e3d5e227e469c8acd599cd86c07af9b18541912487
Decimals	18
Supply	350,000,000
Platform	Binance Smart Chain
Compiler	v0.8.14+commit.80d49f37
Token Name	SAFEFTT
Symbol	SFTT
License Type	None
Language	Solidity

CONTRACT FUNCTION SUMMARY



Can edit Tax?

DETECTED

Can take back Ownership?

NOT DETECTED

Is Blacklisted?

DETECTED

Is Whitelisted?

DETECTED

Is Mintable?

DETECTED

Can transfer Pausable?

DETECTED

Is Trading with CooldownTime?

DETECTED

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.

- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.

- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.

- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.

- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.

- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

Issues Checking Status

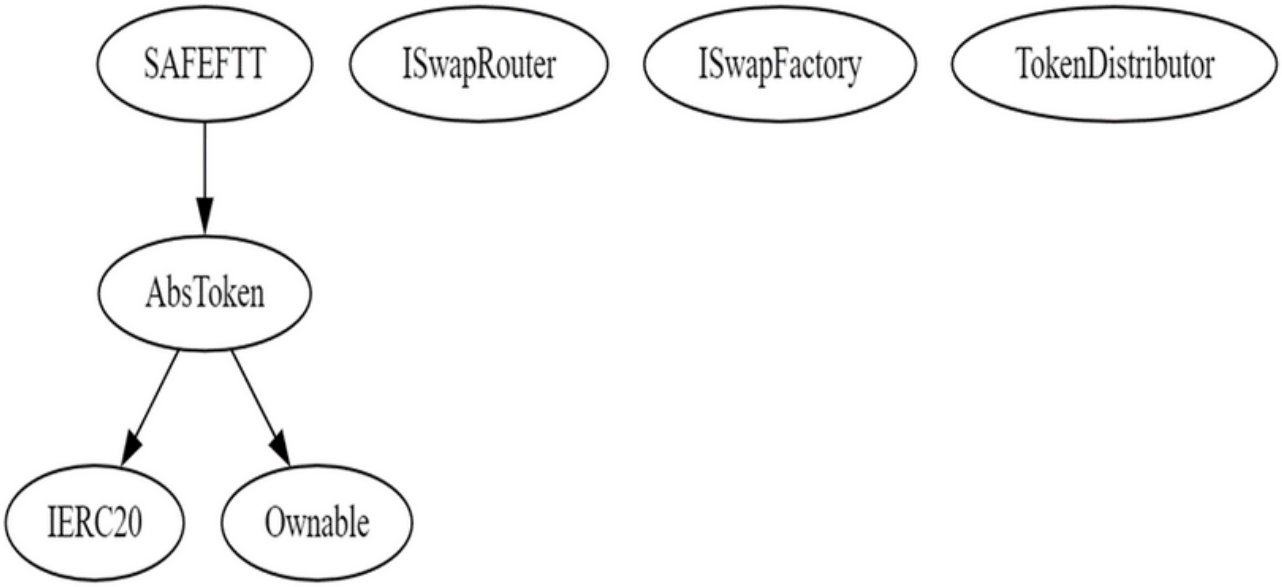
No	Issue Description	Checking Status
1	Compiler warnings.	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Design Logic.	Passed
12	Cross-function race conditions.	Passed
13	Safe Zeppelin module.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Fallback function security.	Passed
17	Arithmetic accuracy.	Passed

SWC ATTACK TEST

SWC ID	Description	Test Result
SWC-100	Function Visibility	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed

SWC ID	Description	Test Result
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Grieving	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions with Multiple Variable Length Arguments	Passed
SWC-134	Unencrypted Private Data On-Chain	Passed

Inheritance Tree



Summary

- Owner is able to mint unlimited tokens
- Owner is able to set taxes up to 100%
- Owner is able to blacklist an arbitrary address from selling/transferring tokens
- Owner is able to set max buy/transfer amount to 0
- Owner is able to re-initialize anti-sniper at any time with arbitrary dead blocks, buyers in dead blocks will be blacklisted and are not able to sell/transfer their tokens anymore
- Owner is able to disable trading at any time

Classification of Risks

Severity	Description
◆ **High-Risk**	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
◆ **Medium-Risk**	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
◆ **Low-Risk**	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
◆ **Gas Optimization /Suggestion**	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
◆ **High-Risk**	**6**
◆ **Medium-Risk**	**0**
◆ **Low-Risk**	**0**
◆ **Gas Optimization / Suggestions**	**0**

MANUAL AUDIT

High Risk Findings

Centralization - Owner is able to mint unlimited new tokens:

In a normal transfer, sender balance must be reduced and receiver must be increased by transfer amount, but in SAFEFTT contract, sender balance is reduced only if:
1- sender is not whitelisted
2- sender and recipient are not same address
a malicious owner only needs to whitelist an arbitrary wallet using setFeeWhiteList function, and then by sending tokens to itself (same sender and recipient) can increase his balance by unlimited amounts
Code:

```
_transfer:
if (pancakeRouter(from,to)) {
        require(balance >= amount, "balanceNotEnough");}
_tokenTransfer:
if (pancakeRouter(sender,recipient))
        _balances[sender] = _balances[sender] - tAmount;
_takeTransfer function, only increase recipient balance:
_balances[to] = _balances[to] + tAmount;
emit Transfer(sender, to, tAmount);
```

MANUAL AUDIT

Centralization - Owner is able to set taxes up to 100%:

Code:

```
function setBuyLPDividendFee(uint256 dividendFee) external
onlyOwner {
    _buyLPDividendFee = dividendFee;
}
function setBuyFundFee(uint256 fundFee) external onlyOwner {
    _buyFundFee = fundFee;
}
function setSellLPDividendFee(uint256 dividendFee) external
onlyOwner {
    _sellLPDividendFee = dividendFee;
}
function setSellFundFee(uint256 fundFee) external onlyOwner {
    _sellFundFee = fundFee;
}
function setSellLPFee(uint256 lpFee) external onlyOwner {
    _sellLPFee = lpFee;
}
```

MANUAL AUDIT

Centralization - Owner is able to blacklist an arbitrary address from selling/transferring tokens:

Code:

```
function setGirls(address addr, bool enable) external onlyOwner {
    _girls[addr] = enable;
  }
_tokenTransfer:
        require(!_girls[sender]);
```

MANUAL AUDIT

Centralization - Owner is able to set max buy/transfer amount to 0

Code:

```
_tokenTransfer:
if (isSell) {
        swapFee = _sellFundFee + _sellLPDividendFee + _sellLPFee;
} else {
        require(tAmount <= maxTXAmount);
        swapFee = _buyFundFee + _buyLPDividendFee;
}
setMaxTxAmount:
function setMaxTxAmount(uint256 max) public onlyOwner {
     maxTXAmount = max;
}
```

MANUAL AUDIT

Centralization - Owner is able to disable trading at any time (buying/selling/transferring):

Code:

```
return moon:
function returnMoon() external onlyOwner {
     goMoonBlock = 0;
}
_transfer:
if (_swapPairList[from] || _swapPairList[to]) {
    if (!_feeWhiteList[from] && !_feeWhiteList[to]) {
        if (0 == goMoonBlock) {
            require(false);
       }
```

MANUAL AUDIT

Centralization - Owner is able to re-initialize anti-sniper at any time with arbitrary dead blocks, buyers in dead blocks will be blacklisted and are not able to sell/transfer their tokens anymore:

Code:

```
_transfer:
if (block.number < goMoonBlock + kb && !_swapPairList[to]) {
      _girls[to] = true;
}
_transferTokens:
require(!_girls[sender]);
setkb (reseting anti-bot):
function setkb(uint256 a) public onlyOwner{
     kb = a;
}
```