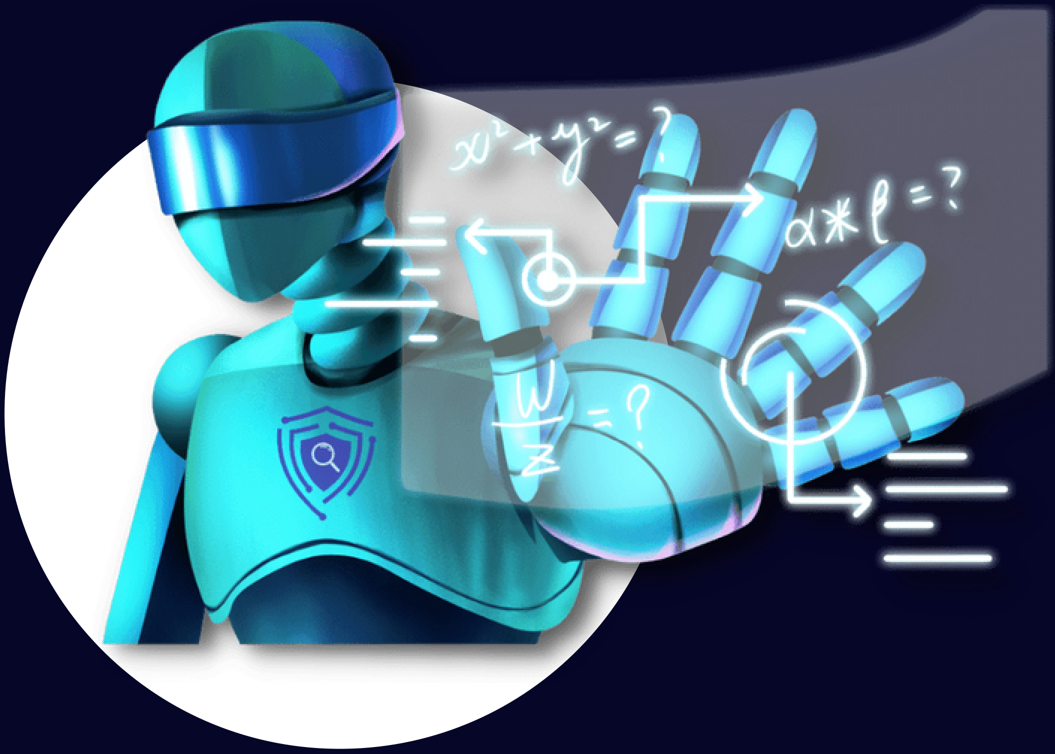
SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY

SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



PROJECT: **SEXN - $SOT TOKEN**

DATE: May 24, 2022

www.safuaudit.com

INTRODUCTION

Client	Sexn - $SOT
Language	Solidity
Contract address	0x9a01AAe88e7e438E4F9E972865fABBB702f0CdA6
Owner	0x161302D3D0b59de878e2331d5CF4d36311DC5A14
Deployer	0x161302D3D0b59de878e2331d5CF4d36311DC5A14
SHA1-Hash	47d9048caf6ad71a1993054e30d3e96670c091f2
Decimals	9
Supply	10,000,000
Platform	Binance Smart Chain
Compiler	v0.8.7+commit.e28d00a7
Optimization	Yes with 200 runs
Website	https://www.sexn.finance/
Telegram	https://t.me/SEXN_Official
Twitter	https://twitter.com/sexnweb3

This audit is for $SOT - the Application Token of SEXN

TABLE OF CONTENTS

APPROACH

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch, that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- Remix IDE
- Mythril
- Open Zeppelin Code Analyzer
- Solidity Code Complier
- Hardhat

RISK CLASSIFICATION

CRITICAL

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

MEDIUM

Issues on this level could potentially bring problems and should eventually be fixed.

MINOR

Issues on this level are minor details and warning that can remain unfixed but would be better fixed at some point in the future

INFORMATIONAL

Information level is to offer suggestions for improvement of efficacity or security for features with a risk free factor.

CONTRACT INSPECTION 🔎

Imported contracts or frameworks used:

```
| **IERC20** | Interface |  |||
| **Token** | Interface |  |||
| **IUniswapV2Factory** | Interface |  |||
| **IUniswapV2Router02** | Interface |  |||
| **Context** | Implementation |  |||
| **SafeMath** | Library |  |||
| **Ownable** | Implementation | Context |||
```

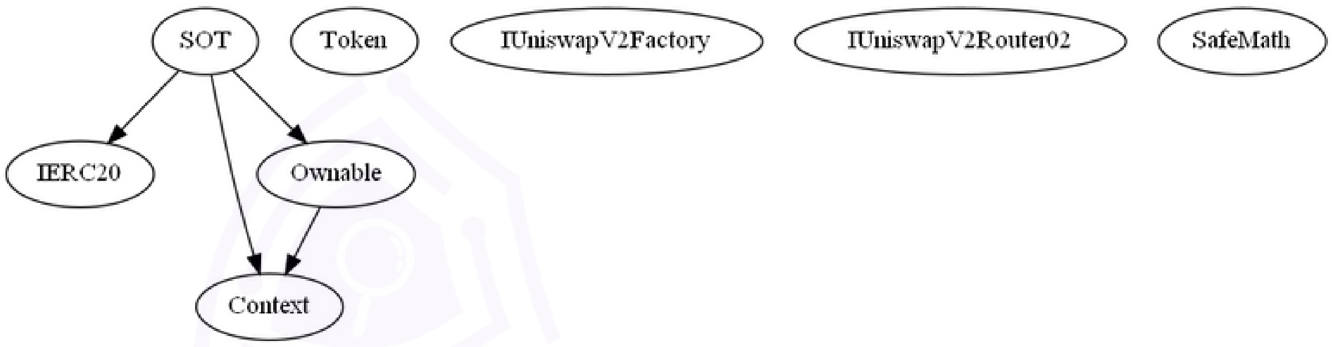
Tested Contract File:

```
|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| SOT.sol | 47d9048caf6ad71a1993054e30d3e96670c091f2 |
```

```
| **SOT** | Implementation | Context, IERC20, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO❗️ |
| └ | name | Public ❗️ |   |NO❗️ |
| └ | symbol | Public ❗️ |   |NO❗️ |
| └ | decimals | Public ❗️ |   |NO❗️ |
| └ | totalSupply | Public ❗️ |   |NO❗️ |
| └ | balanceOf | Public ❗️ |   |NO❗️ |
| └ | transfer | Public ❗️ | 🛑  |NO❗️ |
| └ | allowance | Public ❗️ |   |NO❗️ |
| └ | approve | Public ❗️ | 🛑  |NO❗️ |
| └ | transferFrom | Public ❗️ | 🛑  |NO❗️ |
| └ | tokenFromReflection | Private 🔐 |   | |
| └ | _approve | Private 🔐 | 🛑  | |
| └ | _transfer | Private 🔐 | 🛑  | |
| └ | swapTokensForEth | Private 🔐 | 🛑  | lockTheSwap |
| └ | sendETHToFee | Private 🔐 | 🛑  | |
| └ | _tokenTransfer | Private 🔐 | 🛑  | |
| └ | rescueForeignTokens | Public ❗️ | 🛑  | onlyDev |
| └ | setFee | Public ❗️ | 🛑  | onlyDev |
| └ | setNewMarketingAddress | Public ❗️ | 🛑  | onlyDev |
| └ | _transferStandard | Private 🔐 | 🛑  | |
| └ | _takeTeam | Private 🔐 | 🛑  | |
| └ | _reflectFee | Private 🔐 | 🛑  | |
| └ | <Receive Ether> | External ❗️ | 💵 |NO❗️ |
| └ | _getValues | Private 🔐 |   | |
| └ | _getTValues | Private 🔐 |   | |
| └ | _getRValues | Private 🔐 |   | |
| └ | _getRate | Private 🔐 |   | |
| └ | _getCurrentSupply | Private 🔐 |   | |
| └ | manualswap | External ❗ | 🛑  |NO❗ |
| └ | manualsend | External ❗ | 🛑  |NO❗ |
| └ | toggleSwap | Public ❗ | 🛑  | onlyDev |
| └ | excludeMultipleAccountsFromFees | Public ❗ | 🛑  | onlyOwner |
```

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable
🔐	Private function
🔒	Internal function
NO❗	Function has no modifier

INHERITANCE TREE



Inheritance is a feature of the object-oriented programming language. It is a way of extending the functionality of a program, used to separate the code, reduces the dependency, and increases the re-usability of the existing code. Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract.

OWNER PRIVILEGES

Fees

- Buy Fees: 3%
- Sell Fees: 3%

Fees privileges

- Can set buy and sell fees up to 7%
- Can exclude multiple accounts from fees

Ownership

- Owned

Minting

- No mint function

Max Tx Amount

- Can't set max Tx amount

Pause function

- Can't pause trading

Blacklist

- Can't blacklist

MANUAL FUNCTIONS ANALYSIS

The contract is verified to check if functions do and work as they should and malicious code is not inserted.

	Tested	**Result**
Transfer	Yes	Passed
Total Supply	Yes	Passed
Buy Back	Yes	N/A
Burn	Yes	N/A
Mint	Yes	N/A
Rebase	Yes	N/A
Pause	Yes	N/A
Blacklist	Yes	N/A
Lock	Yes	N/A
Max Transaction	Yes	N/A
Transfer Ownership	Yes	Passed
Renounce Ownership	Yes	Passed

VULNERABILITIES TEST

ID	Description	
V-01	Function Default Visibility	Passed
V-02	Integer Overflow and Underflow	Passed
V-03	Outdated Compiler Version	Passed
V-04	Floating Pragma	Minor
V-05	Unchecked Call Return Value	Passed
V-06	Unprotected Ether Withdrawal	Passed
V-07	Unprotected SELF-DESTRUCT Instruction	Passed
V-08	Re-entrancy	Passed
V-09	State Variable Default Visibility	Passed
V-10	Uninitialized Storage Pointer	Passed
V-11	Assert Violation	Passed
V-12	Use of Deprecated Solidity Functions	Passed
V-13	Delegate Call to Untrusted Callee	Passed
V-14	DoS with Failed Call	Passed
V-15	Transaction Order Dependence	Passed
V-16	Authorization through tx.origin	Passed
V-17	Block values as a proxy for time	Passed

V-18	Signature Malleability	**Passed**
V-19	Incorrect Constructor Name	**Passed**
V-20	Shadowing State Variables	**Passed**
V-21	Weak Sources of Randomness from Chain Attributes	**Passed**
V-22	Missing Protection against Signature Replay Attacks	**Passed**
V-23	Lack of Proper Signature Verification	**Passed**
V-24	Requirement Violation	**Passed**
V-25	Write to Arbitrary Storage Location	**Passed**
V-26	Incorrect Inheritance Order	**Passed**
V-27	Insufficient Gas Griefing	**Passed**
V-28	Arbitrary Jump with Function Type Variable	**Passed**
V-29	DoS With Block Gas Limit	**Passed**
V-30	Typographical Error	**Passed**
V-31	Right-To-Left-Override control character (U+202E)	**Passed**
V-32	Presence of unused variables	**Passed**
V-33	Unexpected Ether balance	**Passed**
V-34	Hash Collisions With Multiple Variable Length Arguments	**Passed**
V-35	Message call with the hardcoded gas amount	**Passed**
V-36	Code With No Effects (Irrelevant/Dead Code)	**Passed**
V-37	Unencrypted Private Data On-Chain	**Passed**

FINDINGS

ID	Category	Issue	Severity
V-03	Vulnerabilities	Floating Pragma	Minor
CE-07	Centralization	Withdraw tokens	Medium
GO-01	Gas Optimization	Public functions that should be declared external	Informational
CS-01	Coding Standards	Meaningless State Variables	Informational

V-03: Floating Pragma

Line #29

```
pragma solidity ^0.8.4;
```

Description

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommandation

We advise locking at the lowest pragma version that the contract can be compiled at. For example: pragma solidity 0.8.4;

CE-07: Centralization Risks

CE-07: Owner privileges general risks

The owner of SOT contract has the permission through **onlyDev** modifier to the following:

1. rescueForeignTokens()
2. setFee()
3. setNewMarketingAddress()
4. toggleSwap()

The owner of SOT contract has the permission through **onlyOwner** modifier to the following:

1. excludeMultipleAccountsFromFees()

Description

It should be noted that owner() has the right to convert the SOT in the contract to BNB and withdraw all the BNB in the contract. We recommend that users understand the above information when deciding to hold SOT.

Recommandation

- We advise the client to carefully manage the privilege accounts' private key to avoid any potential risks of being hacked.

GO-01: Public functions that should be declared external

Description

The following functions are declared as public and are not invoked in any of the contracts contained within the project's scope.

- rescueForeignTokens() - Line #328
- setFee() - Line #333
- setNewMarketingAddress() - Line #345
- toggleSwap() - Line #421
- excludeMultipleAccountsFromFees() - Line #425

Recommandation

- We advise that the functions' visibility specifiers are set to external to save gas

CS-01: Coding Standards

Line # 119, 154 : Meaningless State Variables

```
address private _previousOwner;
mapping (address => uint256) private _tOwned;
```

Description

_previousOwner and **_tOwned** are never used

Line # 170, 173, and 176: Variables declared 0 and not used

Description

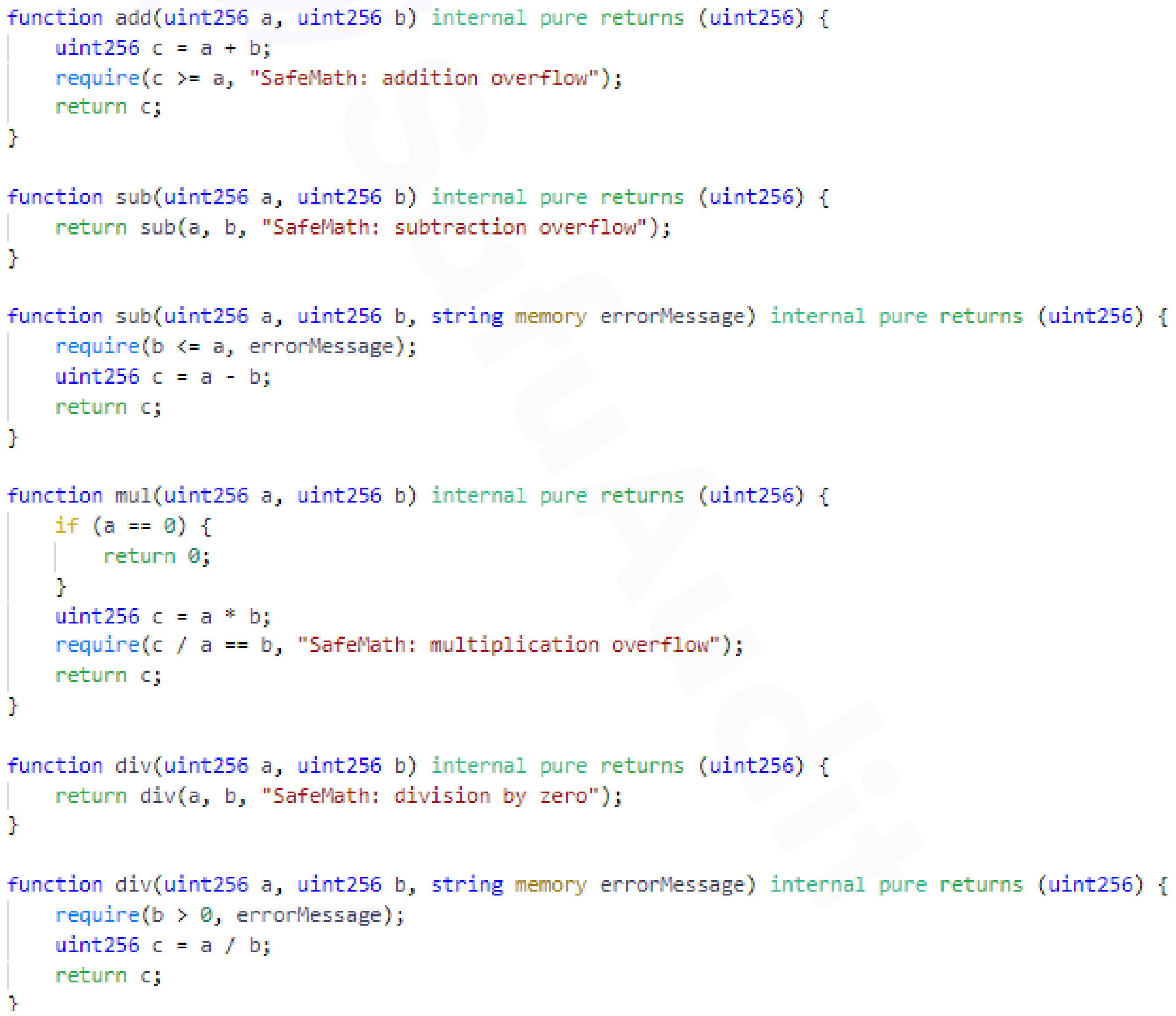
_redisFeeOnSell and **_redisFeeOnBuy** are set to 0 and can be changed in setFee() function but have no effect on the transfers.
_redisFee is used in transfer function, but declared 0 for all buy/sell/transfer cases

Recommandation

- We recommend removing the variables for coding clearness

GOOD PRACTICES ✓

- The owner cannot mint new tokens after deployment
- The owner cannot stop or pause the contract
- The owner cannot set the fees above 7%
- The owner cannot set a transaction limit
- The smart contract utilizes "SafeMath" to prevent overflows



```
function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a, "SafeMath: addition overflow");
    return c;
}

function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    return sub(a, b, "SafeMath: subtraction overflow");
}

function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
    require(b <= a, errorMessage);
    uint256 c = a - b;
    return c;
}

function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    if (a == 0) {
        return 0;
    }
    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");
    return c;
}

function div(uint256 a, uint256 b) internal pure returns (uint256) {
    return div(a, b, "SafeMath: division by zero");
}

function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
    require(b > 0, errorMessage);
    uint256 c = a / b;
    return c;
}
```

WEBSITE

Website	https://www.sexn.finance/
Domain Registry	http://www.godaddy.com/
Domain Expiry Date	2023-04-24
Response Code	200
SSL Checker and HTTPS Test	Passed
Deprecated HTML tags	Passed
Robots.txt	Informative
Sitemap Test	Informative
SEO Friendly URL	Passed
Responsive Test	Passed
JS Error Test	Passed
Console Errors Test	Minor
Site Loading Speed Test	1.56 seconds - Passed
HTTP2 Test	Passed
Safe Browsing Test	Passed

DISCLAIMER

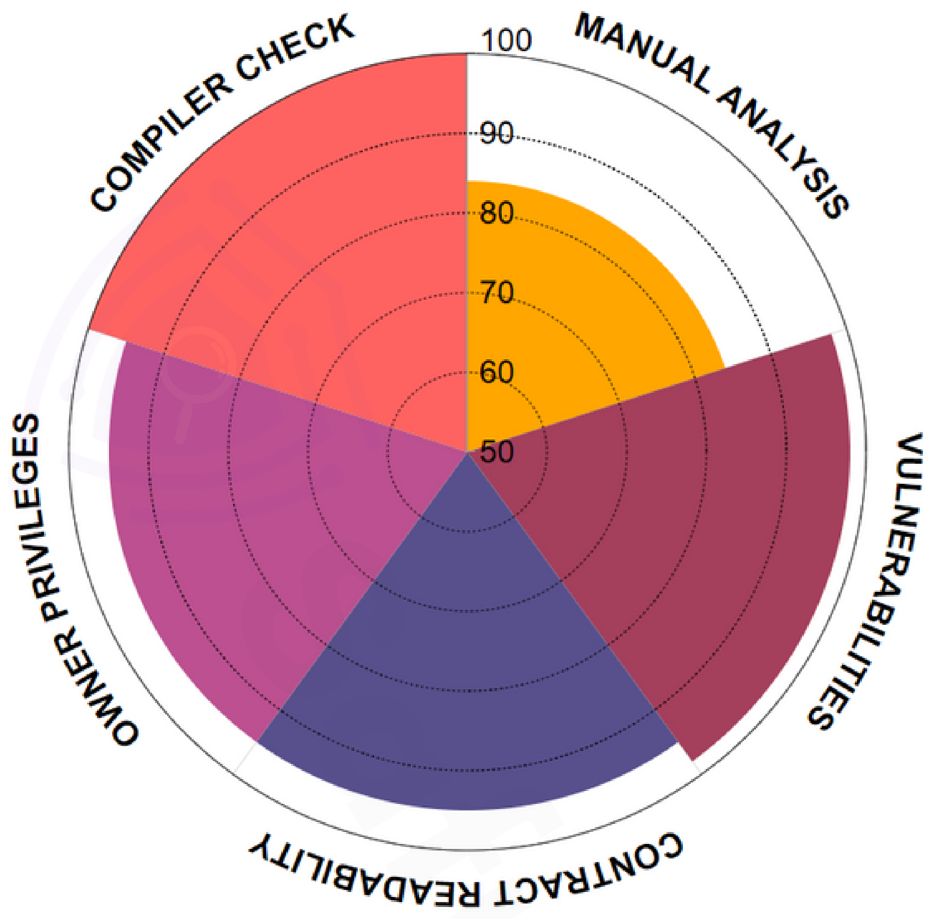
SafuAudit.com is not a financial institution and the information provided on this website does not constitute investment advice, financial advice, trading advice, or any other sort of advice. You should not treat any of the website's content as such. Investing in crypto assets carries a high level of risk and does not hold guarantees for not sustaining financial loss due to their volatility.

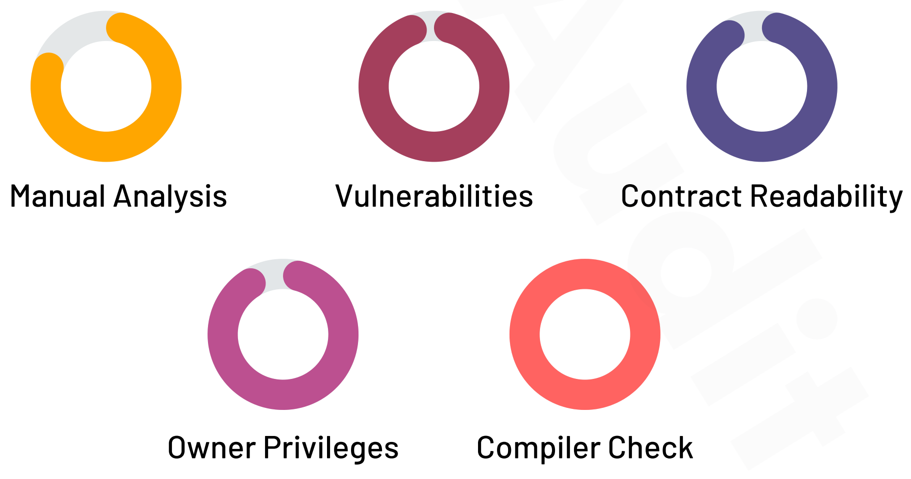
Accuracy of Information
SafuAudit will strive to ensure the accuracy of the information listed on this website although it will not hold any responsibility for any missing or wrong information. SafuAudit provides all information as is. You understand that you are using any and all information available here at your own risk. Any use or reliance on our content and services is solely at your own risk and discretion.

The purpose of the audit is to analyze the on-chain smart contract source code and to provide a basic overview of the project.

While we have used all the information available to us for this straightforward investigation, you should not rely on this report only — we recommend proceeding with several independent audits Be aware that smart contracts deployed on a blockchain aren't secured enough against external vulnerability or a hack. Be aware that active smart contract owner privileges constitute an elevated impact on the smart contract safety and security. Therefore, SafuAudit does not guarantee the explicit security of the audited smart contract. 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.

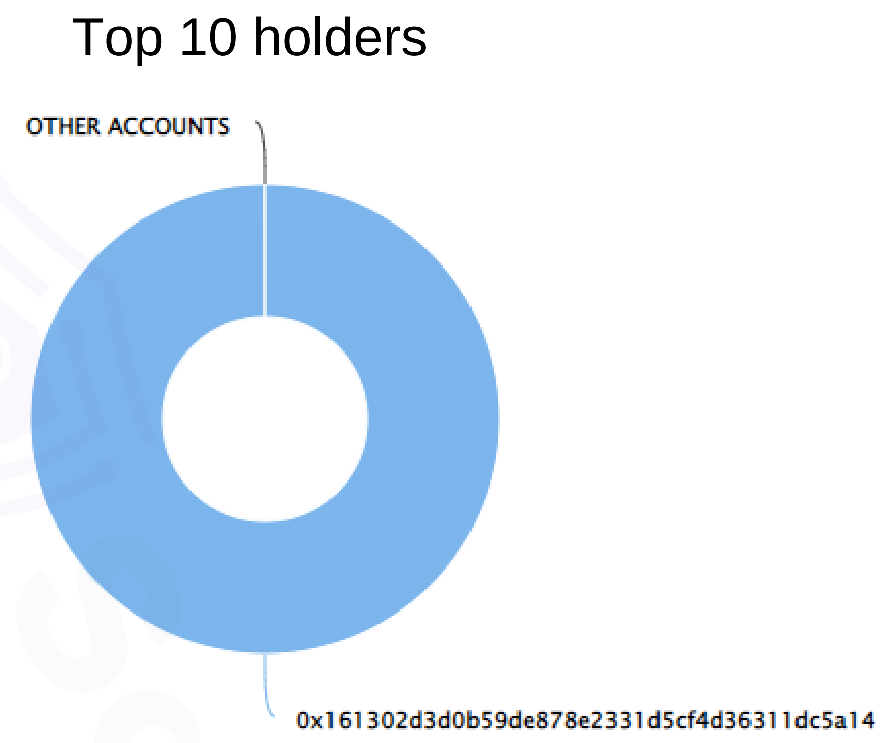
RATING





Final Score:**94.4**

SUMMARY



Rank	Address	Quantity (Token)
1	0x161302d3d0b59de878e2331d5cf4d36311dc5a14	10,000,000

Conclusion

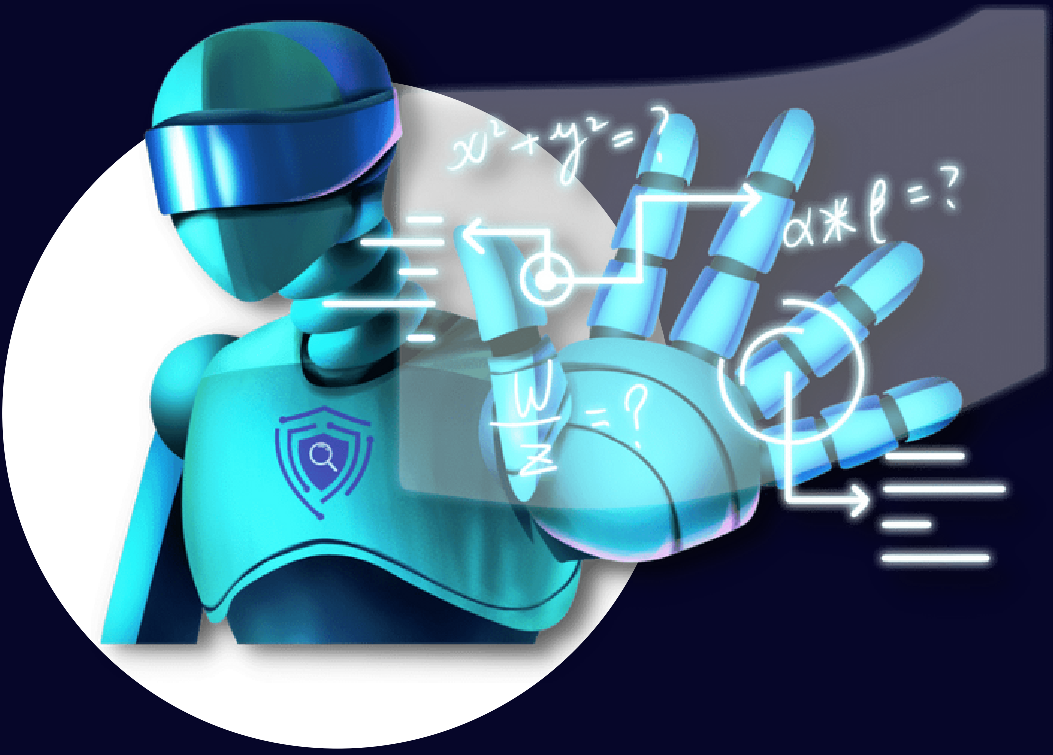
Contract Sexn - $SOT does not contain any severe issues or risk characteristics.

SafuAudit has tested the security based on manual and automated tests. Please note that we don't offer any warranties for the business model.

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY

SAFUAUDIT

SMART CONTRACT AUDITS AND BLOCKCHAIN SECURITY



"Only in growth, reform, and change, paradoxically enough, is true security to be found."

www.safuaudit.com