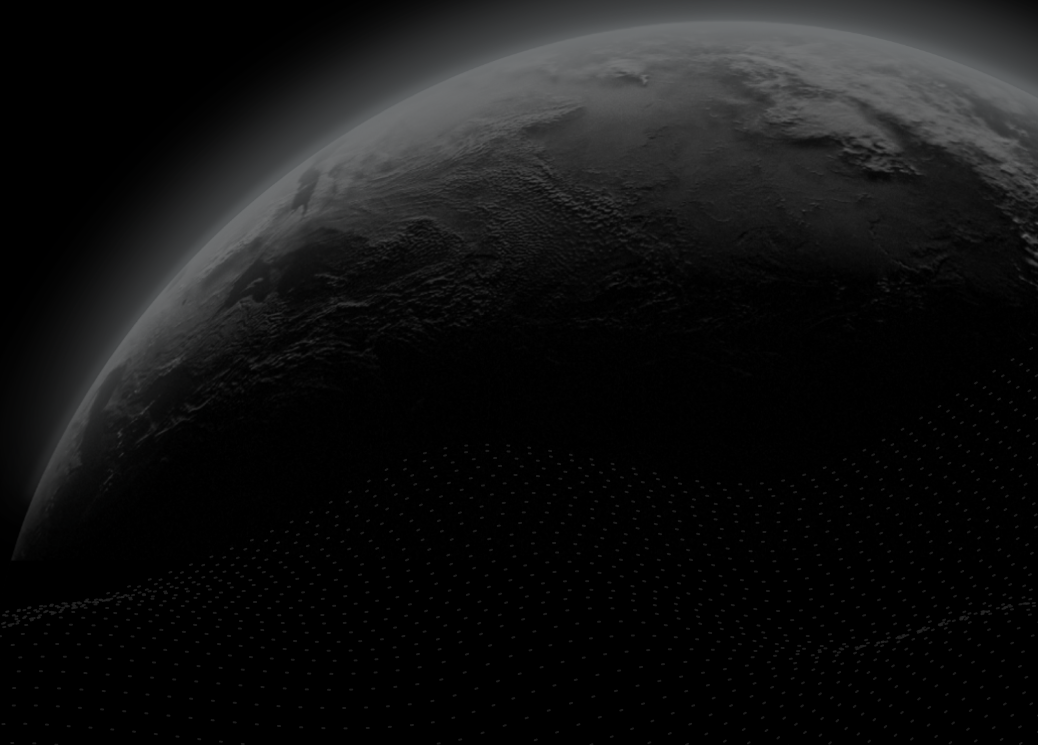
CERTIK

Security Assessment

SwiftOTC - Audit

CertiK Assessed on Mar 5th, 2024



CertiK Assessed on Mar 5th, 2024

SwiftOTC - Audit

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Binance Smart Chain (BSC)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 03/05/2024	N/A

CODEBASE	COMMITS
contracts	af3d359fd88fa06795de042186e04a935d4b556b
View All in Codebase Page	View All in Codebase Page

Highlighted Centralization Risks

Initial owner token share is 100%

Vulnerability Summary

10	3	1	0	6	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
2	Major	1 Mitigated, 1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
2	Medium	2 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
6	Minor	3 Resolved, 3 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
0	Informational		Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | SWIFTOTC - AUDIT

CODEBASE | SWIFTOTC - AUDIT

Repository

contracts

Commit

af3d359fd88fa06795de042186e04a935d4b556b

AUDIT SCOPE | SWIFTOTC - AUDIT

2 files audited ● 2 files with Acknowledged findings

ID	Repo	File	SHA256 Checksum
● SOT	SwiftOTC/contracts	swiftotc.sol	719b6ea99a000c3c44d3d38692a63beecdfa8335722243ab8cf0e10451082ebf
● SOC	SwiftOTC/contracts	swiftotctoken.sol	70b04b16dc81e9030622f2bfee9d59f45e46918cccc5aaa88691130329a6f667

APPROACH & METHODS | SWIFTOTC - AUDIT

This report has been prepared for SwiftOTC to discover issues and vulnerabilities in the source code of the SwiftOTC - Audit project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- 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.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS | SWIFTOTC - AUDIT

10	0	2	2	6	0
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for SwiftOTC - Audit. Through this audit, we have uncovered 10 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
GLOBAL-01	**Centralization Related Risks**	**Centralization**	**Major**	**Mitigated**
SOC-01	**Initial Token Distribution**	**Centralization**	**Major**	**Acknowledged**
SOC-02	Delegation Not Moved Along With Tokens	Logical Issue	Medium	Acknowledged
SOT-03	Incompatibility With Deflationary Tokens	Volatile Code	Medium	Acknowledged
OTC-01	Pull-Over-Push Pattern	Logical Issue	Minor	Resolved
SOC-03	Signature Malleability Of `ecrecover`	Volatile Code	Minor	Acknowledged
SOT-01	Missing Zero Address Validation	Volatile Code	Minor	Resolved
SOT-02	Unchecked ERC-20 `transfer()` / `transferFrom()` Call	Volatile Code	Minor	Acknowledged
SOT-04	Inadequate Validation In `bulkBuyOrders` Function For Order Quantity	Volatile Code	Minor	Resolved
SOT-05	No Restriction On Token Listing In `createOrder`	Logical Issue	Minor	Acknowledged

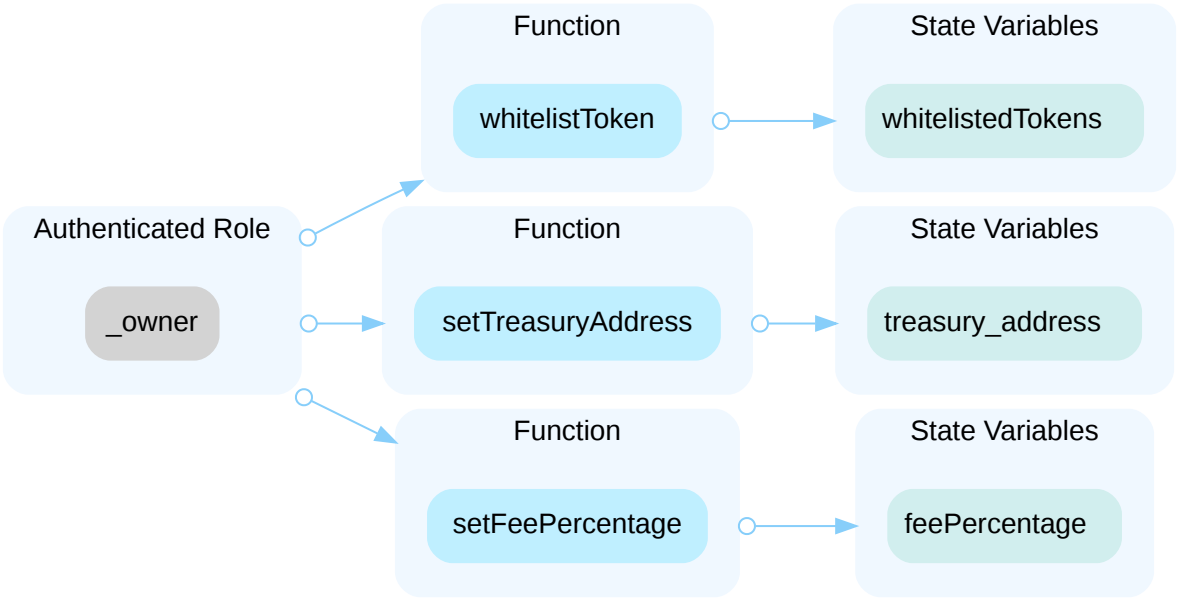
GLOBAL-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	**Major**		**Mitigated**

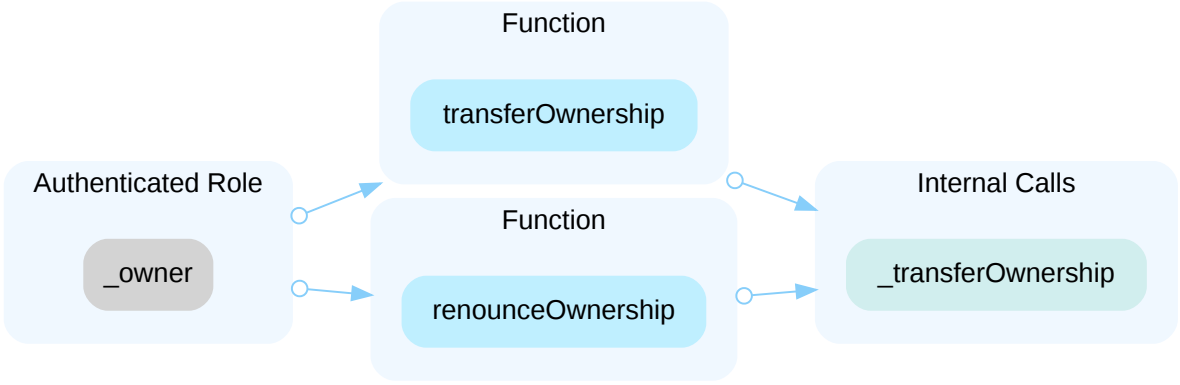
Description

In the contract `OTCSmartContract` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.

- set the treasury address through `setTreasuryAddress`
- add the token to the whitelist
- set the fee percentage through `setFeePercentage`



In the contract `Ownable` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and transfer, renounce the ownership.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[SwiftOTC Team, 03/01/2024]:

The contract deployment address:

https://bscscan.com/address/0xdcb8dd658cb1962f56e82aab544bd47fcd46e23d#code

Contract owner : 0x5C063aF81663a14d055dc56F3B331480a745d2Cf(Timelock)

Multi-sign proxy address:

https://bscscan.com/address/0x0429A038C5603d50601550C98Cd0B60226daD9cE

Signer 1 : 0x0D1fD28a6b435222335F595733834B11701aDaa5

Signer 2: 0x99158f249d0973996b6738Aa390e3ff9EC41D00f

Signer 3: 0x95724Ee61c2b39103C2982e2cfDbCcF968995A84

It requires 2 out of 3 signers to sign the transaction to execute.

Time lock contract address:

https://bscscan.com/address/0x5C063aF81663a14d055dc56F3B331480a745d2Cf

Timelock minDelay: 172800

The Medium Link:

https://medium.com/@swiftotc/the-majority-of-smart-contracts-contain-special-functions-that-are-exclusively-accessible-to-the-0f18d85670a1

[CertiK, 03/01/2024]: While this strategy has indeed reduced the risk, it's crucial to note that it has not completely eliminated it. CertiK strongly encourages the project team to periodically revisit the private key security management of all above-listed addresses.

SOC-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization	**Major**	**swiftotctoken.sol: 802**	**Acknowledged**

Description

All of the `SwiftOTC` tokens are sent to the contract deployer. This is a centralization risk because the deployer can distribute tokens without obtaining the consensus of the community. Any compromise to the deployer address may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution plan should be published in a public location that the community can access. The team should make efforts to restrict access to the private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

Alleviation

[SwiftOTC Team, 02/23/2024]: We will share the status of the published token distribution plan and the multi-sig wallet address that holds the undistributed tokens and the token contract on the mainnet when we will take these steps. Currently, in the near future there is no plan yet for a token.

[SwiftOTC Team, 03/01/2024]: We will not have a token yet, when we will have a token we will mitigate this issue.

[CertiK, 02/23/2024]: It is suggested to implement the aforementioned methods to avoid centralized failure. Also, CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

SOC-02 | DELEGATION NOT MOVED ALONG WITH TOKENS

Category	Severity	Location	Status
Logical Issue	Medium	swiftotctoken.sol: 802	Acknowledged

Description

The voting power of delegation is not moved from one account to another account or `address(0)` along with the `transfer()`、`transferFrom()` and `mint()`. Current `transfer()`、`transferFrom()` and `mint()` are from BEP20 protocol and don't invoke `_moveDelegates()`.

Recommendation

We recommend moving delegation along with these functions.

Alleviation

[SwiftOTC Team, 02/23/2024]: We will not use the Token Governance.

[SwiftOTC Team, 03/01/2024]: We will not have a token yet, when we will have a token we will mitigate this issue.

SOT-03 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS

Category	Severity	Location	Status
Volatile Code	Medium	swiftotc.sol: 1009	Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, an inconsistency in the amount will occur and the transaction may fail due to the validation checks. For example, if a user sends 100 deflationary tokens (with a 10% transaction fee) to the target contract, only 90 tokens actually arrive to the contract.

In the `swiftotc` contract, the `createOrder()` function calls `IERC20(_tokenToSell).transferFrom` to transfer `_tokenToSell` from the `msg.sender` to the contract.

Recommendation

We advise the client to regulate the set of tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[SwiftOTC Team, 02/23/2024]: We will issue several warnings on the interface that we do not support deflationary tokens, and if you do want to trade them then you must send more tokens based on the transaction fees of the token manually.

OTC-01 | PULL-OVER-PUSH PATTERN

Category	Severity	Location	Status
Logical Issue	Minor	swiftotc.sol: 830; swiftotctoken.sol: 786~790	Resolved

Description

The change of `owner` by function `transferOwnership()` overrides the previously set `owner` with the new one without guaranteeing the new `owner` has the ability to actuate transactions on-chain.

Recommendation

We advise refactoring the linked codes as below:

```
69  address public pendingOwner;
70
71  function renounceOwnership() public onlyOwner {
72      _owner = address(0);
73      pendingOwner = address(0);
74      emit OwnershipTransferred(_owner, address(0));
75  }
76
77  function transferOwnership(address newOwner) public onlyOwner {
78      require(address(0) != newOwner, "pendingOwner set to the zero address.");
79      pendingOwner = newOwner;
80  }
81
82  function claimOwnership() public {
83      require(msg.sender == pendingOwner, "caller != pending owner");
84
85      _owner = pendingOwner;
86      pendingOwner = address(0);
87      emit OwnershipTransferred(_owner, pendingOwner);
88  }
```

Alleviation

[SwiftOTC Team, 02/23/2024]: It will be taken care when transferring the owner that the address is the correct one, we wish to stick with the OpenZeppelin Standard for Ownable.

[SwiftOTC Team, 03/01/2024]: This can be mitigated through the timelock & multi-sig since there will be no transfer of ownership after, there would be no reason for it.

SOC-03 | SIGNATURE MALLEABILITY OF `ecrecover`

Category	Severity	Location	Status
Volatile Code	Minor	swiftotctoken.sol: 900	Acknowledged

Description

The `ecrecover()` function is subject to signature malleability. The signature malleability is possible within the Elliptic Curve cryptographic system. An Elliptic Curve is symmetric on the `X`-axis, meaning two points can exist with the same `X` value. In the `r`, `s` and `v` representation this permits us to carefully adjust `s` to produce a second valid signature for the same `r`, thus breaking the assumption that a signature cannot be replayed in what is known as a replay-attack.

```
900           address signatory = ecrecover(digest, v, r, s);
```

- `ecrecover` called without proper checks.

Recommendation

We recommend adding the following checks or to consider the example in `ECDSA.sol` from the OpenZeppelin library.

```
require(uint256(s) <=
0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0, "ECDSA: invalid
signature 's' value");
require(uint8(v) == 27 || uint8(v) == 28, "ECDSA: invalid signature 'v' value");
```

Alleviation

[SwiftOTC Team, 02/23/2024]: Issue acknowledged.

[SwiftOTC Team, 03/01/2024]: We will not have a token yet, when we will have a token we will mitigate this issue.

SOT-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	swiftotc.sol: 954, 974	Resolved

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent loss of those tokens.

```
954         treasury_address = _treasuryAddress;
```

- `_treasuryAddress` is not zero-checked before being used.

```
974         treasury_address = treasuryAddress;
```

- `treasuryAddress` is not zero-checked before being used.

Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[CertiK, 03/05/2024]: `treasury_address` is set to `0x28b7A1818a6BB08Ed817d4D03eDB7457b4C4a71F`.

SOT-02 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	Minor	swiftotc.sol: 1009~1013, 1065~1069, 1078~1082, 1084, 1120~1124, 1133~1137, 1139, 1161	Acknowledged

Description

The return values of the `transfer()` and `transferFrom()` calls in the smart contract are not checked. Some ERC-20 tokens' transfer functions return no values, while others return a bool value, they should be handled with care. If a function returns `false` instead of reverting upon failure, an unchecked failed transfer could be mistakenly considered successful in the contract.

```
1009        IERC20(_tokenToSell).transferFrom(
1010            msg.sender,
1011            address(this),
1012            _amountToSell
1013        );
```

```
1065            IERC20(order.tokenToBuy).transferFrom(
1066                msg.sender,
1067                treasury_address,
1068                feeAmount
1069            );
```

```
1078        IERC20(order.tokenToBuy).transferFrom(
1079            msg.sender,
1080            order.seller,
1081            transferAmountToSeller
1082        );
```

```
1084        IERC20(order.tokenToSell).transfer(msg.sender, order.amountToSell);
```

```
1120                IERC20(order.tokenToBuy).transferFrom(
1121                    msg.sender,
1122                    treasury_address,
1123                    feeAmount
1124                );
```

```
1133             IERC20(order.tokenToBuy).transferFrom(
1134                 msg.sender,
1135                 order.seller,
1136                 transferAmountToSeller
1137             );
```

```
1139             IERC20(order.tokenToSell).transfer(msg.sender, order.amountToSell
);
```

```
1161         IERC20(order.tokenToSell).transfer(order.seller, order.amountToSell);
```

Recommendation

It is advised to use the OpenZeppelin's `SafeERC20.sol` implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if false is returned, making it compatible with all ERC-20 token implementations.

Alleviation

[SwiftOTC Team, 02/23/2024]: Issue acknowledged.

SOT-04 INADEQUATE VALIDATION IN `bulkBuyOrders` FUNCTION FOR ORDER QUANTITY

Category	Severity	Location	Status
Volatile Code	Minor	swiftotc.sol: 1094	Resolved

Description

The `bulkBuyOrders()` function, designed for bulk order execution, incorporates a validation flaw at line 1094. The validation condition `require(orderIds.length < orders.length, "Invalid orderIds")` is intended to check that the quantity of orders processed in a transaction is strictly less than the total number of orders. We understand that when the number of orders passed in as a parameter is equal to the total number of orders, the transaction should also be executable.

Recommendation

We recommend performing a review of the validation logic in the `bulkBuyOrders` function on line 1094.

Alleviation

[SwiftOTC Team, 02/23/2024]: Issue acknowledged. Changes have been reflected in the commit hash: https://github.com/SwiftOTC/contracts/commit/f6ddfe9626ff599288b780e11655cb4f71a62aca

SOT-05 | NO RESTRICTION ON TOKEN LISTING IN `createOrder`

Category	Severity	Location	Status
Logical Issue	Minor	swiftotc.sol: 990	Acknowledged

Description

The `createOrder()` function, responsible for order creation, lacks restrictions on the sale of tokens. Users can transfer assets to the contract without limitations on selling tokens in the order, potentially leading to malicious token listing in the contract. This facilitate phishing attack.

```
989     function createOrder(
990         address _tokenToSell,
991         uint256 _amountToSell,
992         address _tokenToBuy,
993         uint256 _amountToBuy,
994         address _allowedBuyer
995     ) public nonReentrant noContractsAllowed {
```

Recommendation

Currently, the contract implements a whitelist mechanism that will charge fees when listing tokens that are not in whitelist. We recommend using the whitelist to restrict the tokens that can be listed in an order within the `createOrder()` function.

Alleviation

[SwiftOTC Team, 02/23/2024]: In order to not centralize the contract we will not add the whitelist feature. We will create multiple warnings before buying an order to check the contract address and all the things associated in order to prevent malicious tokens to be traded.

APPENDIX | SWIFTOTC - AUDIT

Finding Categories

Categories	Description
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.