



1inch

FixedRateSwap

SMART CONTRACT AUDIT

05.08.2021

Made in Germany by Chainsulting.de



Table of contents

1. Disclaimer	4
2. About the Project and Company	5
2.1 Project Overview	6
3. Vulnerability & Risk Level.....	7
4. Auditing Strategy and Techniques Applied	8
4.1 Methodology.....	8
4.2 Used Code from other Frameworks/Smart Contracts	9
4.3 Tested Contract Files.....	10
4.4 Metrics / CallGraph	11
4.5 Metrics / Source Lines & Risk	12
4.6 Metrics / Capabilities.....	13
4.7 Metrics / Source Unites in Scope.....	14
5. Scope of Work.....	15
5.1 Manual and Automated Vulnerability Test	16
5.2. SWC Attacks	17
5.3. Verify Claims	21
5.3.1. Deploy TokenA (TKNA)	21
5.3.2. Deploy TokenB (TKNB)	21
5.3.3. Deploy FixedRateSwap Token (STKNAB)	21
5.3.4. Depositing 10,000 TKN1 in FixRateSwap from not owner address	22
5.3.5. Depositing 10,000 of each token in FixRateSwap from owner address	22
5.3.6. Swap TKNA to TKNB	23



5.3.7. Swap TKNB to TKNA	23
5.3.8. Swap TKNA to TKNB	23
5.3.9. Withdraw 10,000 STKNAB	23
5.3.10. Send token to contract.....	24
5.3.11. Withdraw remaining tokens.....	24
5.3.12. Transfer ownership.....	25
6. Executive Summary	26
7. Deployed Smart Contract.....	26



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of 1Inch Exchange. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (04.08.2021)	Layout
0.2 (05.08.2021)	Test Deployment
0.5 (05.08.2021)	Manual & Automated Security Testing
0.6 (05.08.2021)	Testing SWC Checks
0.7 (05.08.2021)	Verify Claims
0.9 (05.08.2021)	Summary and Recommendation
1.0 (05.08.2021)	Final document
1.1 (TBA)	Added deployed contract addresses

2. About the Project and Company

Company address:

1Inch Limited
Quijano Chambers, P.O. Box 3159, Road Town
Tortola, British Virgin Islands

Sergej Kunz Co-Founder & Chief Executive Officer
Anton Bukov Co-Founder & Chief Technology Officer

Discord: <https://discord.gg/FZADkCZ>

Blog: <https://blog.1inch.io>

Medium: <https://medium.com/@1inch.exchange>

Website: <https://app.1inch.io>

Twitter: <https://twitter.com/1inchExchange>

Reddit: https://www.reddit.com/r/1inch_exchange

Telegram: <https://t.me/OneInchExchange>

Forum: <https://gov.1inch.io>



2.1 Project Overview

The 1inch Network unites decentralized protocols whose synergy enables the most lucrative, fastest and protected operations in the DeFi space. The initial protocol of the 1inch Network is a DEX aggregator solution that searches deals across multiple liquidity sources, offering users better rates than any individual exchange.

This protocol incorporates the Pathfinder algorithm which finds the best paths among different markets over 50+ liquidity sources on Ethereum, 20+ liquidity sources on Binance Smart Chain and 8+ liquidity sources on Polygon. In just two years the 1inch DEX aggregator surpassed \$50B in overall volume on the Ethereum network alone. The 1inch Aggregation Protocol facilitates cost-efficient and secure swap transactions across multiple liquidity sources.

The 1inch Liquidity Protocol is a next-generation automated market maker that protects users from front-running attacks and offers attractive opportunities to liquidity providers. The 1inch Limit Order Protocol facilitates the most innovative and flexible limit order swap opportunities in DeFi. The protocol's features, such as dynamic pricing, conditional orders and extra RFQ support, power various implementations, including stop-loss and trailing stop orders, as well as auctions.

FixedRateSwap is a simple AMM that allows swapping tokens with 1:1 rate and variable fee. Deposit method is limited to onlyOwner by design. Fee is calculated depending on the ratio of balances of the tokens in the contract.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

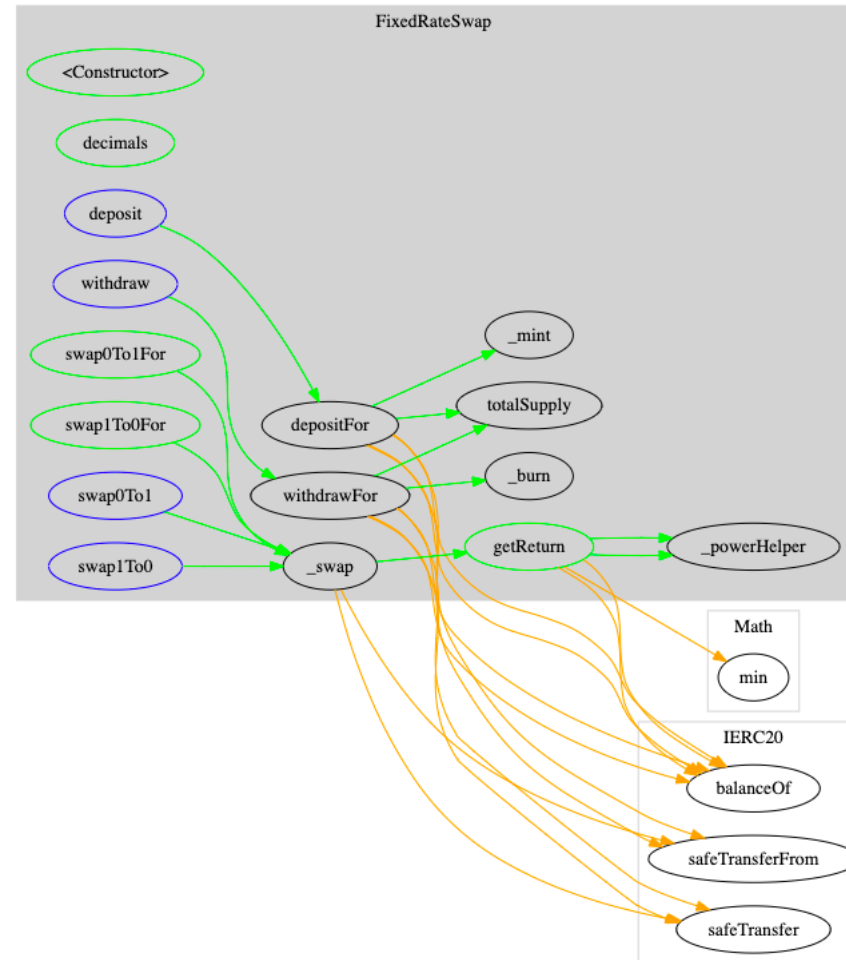
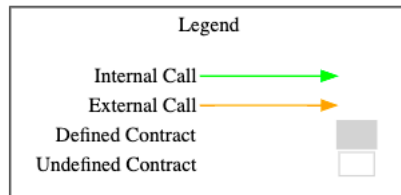
Dependency / Import Path	Source
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.2.0/contracts/access/Ownable.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.2.0/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.2.0/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/utils/math/Math.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v4.2.0/contracts/utils/math/Math.sol

4.3 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

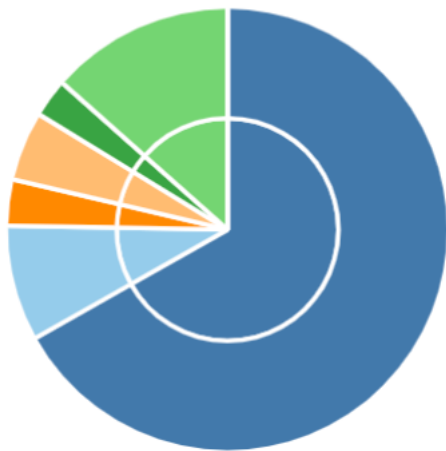
File	Fingerprint (MD5)
FixedRateSwap.sol	8b9bd9e653b7dcc9a6d648c05f703071

4.4 Metrics / CallGraph

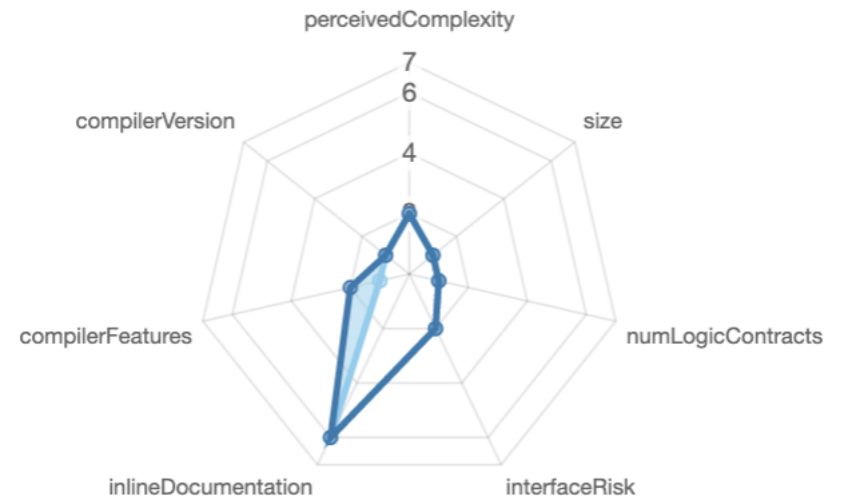


4.5 Metrics / Source Lines & Risk










source comment single block mixed
empty todo blockEmpty



overall average





4.6 Metrics / Capabilities


Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts	
<div><div>^0.8.0</div></div>				<div></div>		**** (0 asm blocks)		<div></div>	
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECTransfer	
<div></div>		<div></div>		<div></div>		<div></div>		<div></div>	

Exposed Functions



This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable				
10	0				
External	Internal	Private	Pure	View	
4	10	2	1	2	

StateVariables

Total	 Public
7	2

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complexity Score	Capabilities
	FixedRateSwap.sol	1		153	153	119	15	87	
	Totals	1		153	153	119	15	87	

Legend: []

- **Lines:** total lines of the source unit
- **nLines:** normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC:** normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines:** lines containing single or block comments
- **Complexity Score:** a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

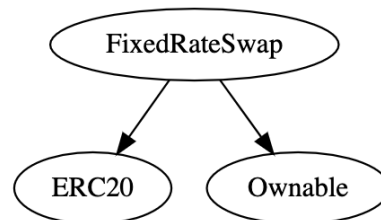
5. Scope of Work

The 1inch Team provided us with the files that needs to be tested. The scope of the audit is the Fixed Rate Swap contract.

The team put forward the following assumptions regarding the security, usage of the contracts:

- The smart contract is coded according to the newest standards and in a secure way
- Swapping tokens with 1:1 rate and variable fee is working (USDC / USDT)
- The fee is calculated depending on the ratio of balances of the tokens in the contract.
- Deposit method is limited to onlyOwner.
- The Ownership is possible to transfer.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract

LOW ISSUES

During the audit, Chainsulting's experts found **no Low issues** in the code of the smart contract

INFORMATIONAL ISSUES

During the audit, Chainsulting's experts found **no Informational issues** in the code of the smart contract

5.2. SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✓
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✗
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

5.3. Verify Claims

5.3.1. Deploy TokenA (TKNA)

Initial Supply: 1,000,000 TKNA

Tx: <https://ropsten.etherscan.io/tx/0xef5efbd367bf4bc11bb2ece2b7c517f57e53250d597c9ae6687a67d6296df59a>

Contract: <https://ropsten.etherscan.io/address/0x11300233F4d723A2EBD5372dC0D09C8981458c00>

5.3.2. Deploy TokenB (TKNB)

Initial Supply: 1,000,000 TKNA

Tx: <https://ropsten.etherscan.io/tx/0xb5ce35672d0678f936de6f6dcb8b7a83aeae112499205369ee2628a1b53d2196>

Contract: <https://ropsten.etherscan.io/address/0x5fafA9e93C4D65b7C7E84772eaB6038bd4D5ee1e>

5.3.3. Deploy FixedRateSwap Token (STKNAB)

DEPLOY

_TOKEN0: 0x11300233F4d723A2EBD5372dC0D09C8981458c00

_TOKEN1: 0x5fafA9e93C4D65b7C7E84772eaB6038bd4D5ee1e

NAME: Swap Token AB

SYMBOL: STKNAB

DECIMALS: 18

transact

Tx: <https://ropsten.etherscan.io/tx/0x0a9744ee3475aad07070f4e464057b81545292949e6deb72d9b0d6c9b3f637f0>

Contract: <https://ropsten.etherscan.io/address/0xa7a5Cd88438FDE6b6c62c8e1F824C68532997939>

Deposit Tokens in FixedRateSwap

5.3.4. Depositing 10,000 TKN1 in FixRateSwap from not owner address

Depositing with not owner address fails as expected. The deposit function can only be called by the contract owner.

```
68 | function deposit(uint256 token0Amount, uint256 token1Amount) external returns(uint256 share) {
69 |     share = depositFor(token0Amount, token1Amount, msg.sender);
70 | }
71 |
    | ftrace | funcSig
72 | function depositFor(uint256 token0Amount, uint256 token1Amount, address to) public onlyOwner returns(uint256 share) {
```

✖ Fail with error 'Ownable: caller is not the owner'

Tx: <https://ropsten.etherscan.io/tx/0x632de725b8c602a5e07283c7553293528b7edeaf28d125a4f5c6f774977b98eb>

5.3.5. Depositing 10,000 of each token in FixRateSwap from owner address

Owner can deposit tokens as expected and receives 20,000 Swap Tokens AB (STKNAB)

- ▶ From 0xf40b44261933d... To 0xa7a5cd88438fd... For 10,000 TokenA (TKNA)
- ▶ From 0xf40b44261933d... To 0xa7a5cd88438fd... For 10,000 TokenB (TKNB)
- ▶ From 0x0000000000000000... To 0xf40b44261933d... For 20,000 Swap Token A... (STKNAB)

Tx: <https://ropsten.etherscan.io/tx/0x55d5419f8910d483643085614ed94366f8fbf49f6dd45be57121d0141fd27cb0>

Swap tokens

5.3.6. Swap TKNA to TKNB

‣ From 0x627da5015e602... To 0xa7a5cd88438fd... For 1,000 ⬇ TokenA (TKNA)

‣ From 0xa7a5cd88438fd... To 0x627da5015e602... For 999.9 ⬇ TokenB (TKNB)

Tx: <https://ropsten.etherscan.io/tx/0xb9e023f9cc76be21867d9cf625bb57f885b022e157b4dbdce557f66c6c5ce751>

5.3.7. Swap TKNB to TKNA

‣ From 0x627da5015e602... To 0xa7a5cd88438fd... For 1,000 ⬇ TokenB (TKNB)

‣ From 0xa7a5cd88438fd... To 0x627da5015e602... For 999.9 ⬇ TokenA (TKNA)

Tx: <https://ropsten.etherscan.io/tx/0x3e0087791b468c4bcd50d9c97fbd6a5e3fb0dd10f0984f40b717739e974eea79>

5.3.8. Swap TKNA to TKNB

‣ From 0x627da5015e602... To 0xa7a5cd88438fd... For 2,000 ⬇ TokenB (TKNB)

‣ From 0xa7a5cd88438fd... To 0xf40b44261933d... For 1,999.79999999999567 ⬇ TokenA (TKNA)

Tx: <https://ropsten.etherscan.io/tx/0x110888448fb3084cc16d33a5d1ef8676c5eeb96644da9ffad4016f4d0726cc04>

Withdraw

5.3.9. Withdraw 10,000 STKNAB

Withdraw successful with collected fees. Shares are calculated correctly.

‣ From 0xf40b44261933d... To 0x0000000000000000... For 10,000 ⬇ Swap Token A... (STKNAB)

‣ From 0xa7a5cd88438fd... To 0xf40b44261933d... For 4,000.15000000002165 ⬇ TokenA (TKNA)

‣ From 0xa7a5cd88438fd... To 0xf40b44261933d... For 6,000.05 ⬇ TokenB (TKNB)

Tx: <https://ropsten.etherscan.io/tx/0xa2fe19f1536ec36a1a0efe417b82ad62cdaad77a593ab1f7d35bf69d610207e1>

5.3.10. Send token to contract

Any user can send token to the contract. The token supply in the contract grows and the shares gain value. The sender gets nothing in return, but the shareholders can withdraw the funds.

► From 0x627da5015e602... To 0xa7a5cd88438fd... For 5,000 TokenA (TKNA)

Tx: <https://ropsten.etherscan.io/tx/0xe2b46f54a84b824bd38915fc0ec1e70940c04ca0a1ffe82e012c22caf166ac2f>

5.3.11. Withdraw remaining tokens

Withdrawing the remaining 10,000 STKNAB token. All tokens from the contract are withdrawn.

► From 0xf40b44261933d... To 0x0000000000000000... For 10,000 Swap Token A... (STKNAB)

► From 0xa7a5cd88438fd... To 0xf40b44261933d... For 9,000.15000000002165 TokenA (TKNA)

► From 0xa7a5cd88438fd... To 0xf40b44261933d... For 6,000.05 TokenB (TKNB)

Tx: <https://ropsten.etherscan.io/tx/0x28a1321d125fb034f20d02eb9ad82f7ffe12f2dd5a4a3db14fdd5c554c45040e>

balanceOf

"0xa7a5Cd88438FDE6b6c62c8e1F824C68532997939"

0: uint256: 0

Token A balance after withdrawal

balanceOf

"0xa7a5Cd88438FDE6b6c62c8e1F824C68532997939"

0: uint256: 0

Token B balance after withdrawal

totalSupply

0: uint256: 0

STKNAB supply after withdrawal

5.3.12. Transfer ownership

Ownership can be transferred by owner as expected.

transferOwnership

newOwner: "0x627dA5015e6027a968E5FE5b195D4293D29104E2"



transact

owner

0: address: 0x627dA5015e6027a968E5FE5b195D4293D29104E2

Tx: <https://ropsten.etherscan.io/tx/0x73c160900609f9504e7a33502c0e4a032080900b65bcc88bbf28fc6de7b46930>

6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debrief took place on the August 05, 2021. The overall code quality of the project is good, and the simplicity of the contract greatly benefit the security and decreased the attack surface significant. It correctly implemented widely-used and reviewed contracts from OpenZeppelin.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the functions. During the audit, no critical issues were found after the manual and automated security testing and the claims been successfully verified.

7. Deployed Smart Contract

VERIFIED

Contract is deployed here:

<https://etherscan.io/address/0x40bbdE0eC6F177C4A67360d0f0969Cfc464b0bB4#code>

