Silo Defi

Smart Contract Audit

DATE: 13 October 2022 | **PREPARED BY:** Entersoft Pte Ltd

Contents

Revision History & Version Control

Version	Date	Author(s)	Description
2.0	13-Oct-2022	Rony K	Smart Contract Audit Report for Silo Defi

Entersoft was commissioned by Silo Defi to perform a smart contract audit on their Algorand based contract written in Python. The review was conducted from 3rd August 2022 to 26th September 2022. The validation review was conducted from 11th October 2022 to 12th October 2022.

The report is organized into the following sections.

- Executive Summary: A high-level overview of the validation review conducted on the preliminary security audit findings.
- Technical analysis: Our detailed analysis of the Smart Contract Audit.

The information in this report should be used to understand overall code quality, security, correctness, and the meaning that the code will work as Silo Defi described in the smart contract. The analysis is static and entirely limited to the Smart contract code.

1.0 Disclaimer

This is a limited audit report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to: (i) smart contract best coding practices and issues in the framework and algorithms based on white paper, code, the details of which are set out in this report, (Smart Contract audit). To get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us based on what it says or does not say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

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 copies of this report downloaded 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. No one shall have any right to rely on the report or its contents, and Entersoft Australia and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Entersoft) owe no duty of care towards you or any other person, nor does Entersoft make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and Entersoft hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Entersoft hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Entersoft, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the Smart contract is purely based on the smart contract code shared with us alone.

2.0 Overview

2.1 Project Overview

During the period 3rd August 2022 to 26th September 2022, Entersoft performed a smart contract code audit on Silo Defi protocol contract written in Python, and identified a low-severity vulnerability.

During the period **11th October 2022 - 13th October 2022,** Entersoft performed a validation review on the updated smart contract code of the **Silo protocol** and observed that the vulnerability has been remediated.

2.2 Scope

The scope of the audit was to perform a validation review of the smart contract code available from:

- Github Repository: https://github.com/headline-design/algoptions
- Commit ID: a5456706136ea12f665ab471ac195772bced1950

OUT-OF-SCOPE: External contracts, front end web UI and other imported smart contracts.

2.3 Project Summary

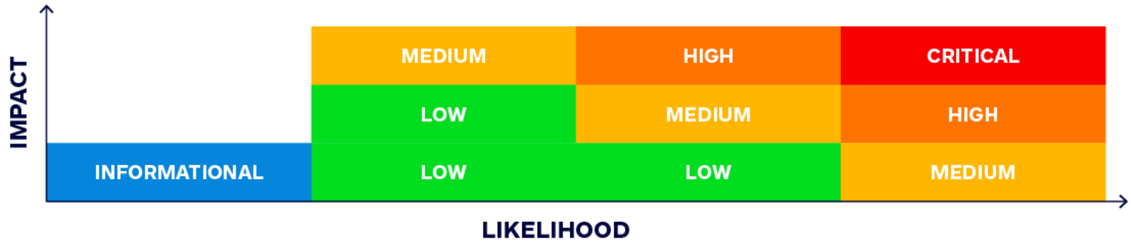
Name	Verified	Audited	Vulnerabilities / Issues
Silo Defi	Yes	Yes	As per the report. Section 4

2.4 Audit Summary

Delivery Date	Method of Audit	Consultants Engaged
13-Oct-2022	Manual review and logic exploitation	2

2.5 Security Level References

Every issue in this report was assigned a severity level from the following classification table:



	MEDIUM	HIGH	CRITICAL
	LOW	MEDIUM	HIGH
INFORMATIONAL	LOW	LOW	MEDIUM

2.6 Vulnerability Summary

● Total Critical	● Total High	● Total Medium	● Total Low	● Total Informational
0	0	0	0	0

3.0 Executive Summary

Entersoft has provided an independent technical security audit to remove smart contract vulnerabilities and to keep it safe from any potential hacks. Smart contract audits assist in protecting blockchain organisations from any fraudulent activity.

The audit performed on the smart contract had some initial security concerns identified during the manual review. The manual review uncovered issues which could impact Silo Defi. We have analyzed the smart contract code line by line and reported the issues identified.

On retest of the smart contract there were no security risks identified out of 11 manual checks performed. The details about these issues can be observed in the technical analysis section.

Result

An audit was conducted on the provided code. The following table provides an overall picture of the security posture.

- ✔ **No security risks identified**
- ✘ **Security risk identified**

#	Smart Contract Audit Test Cases	Result
1	Access Control Policies	✔
2	LOGIC-Sig	✔
3	Decimal Calculation	✔
4	Rekey to Property	✔
5	Input Validation	✔
6	Freeze/Clawback Address	✔
7	Proxy Assessment	✔
8	Fee and Amount Check	✔
9	Pragma Version	✔
10	Group Size Validation	✔
11	Alerthub Setup	✔
Overall Security Posture		**Secure**

4.0 Technical Analysis

The following results are the efforts of manual analysis on the smart contract in scope.

Note: Result set to Negative indicates that there is no security risk. Positive indicates that there is a security risk that needs to be addressed. Informational issues are something that should be followed as a best practice, and are not visible from the smart contract.

4.1 Access Control Policies

Result	Negative
Description	Access control policies would prevent any unauthorized users from performing operations/transactions on the blockchain.
Source File	Not applicable / Not available
Observation	There is no risk observed during the review process in terms of access controls. The authentication process would generally happen when the user connects his wallet (Algorand) to sign the transactions used during exercising the SILO protocol options token. Once the connection is established, the transactions would be signed using the private key on the client device (Participating node). An attacker needs to gain access to the victim's private key to sign the transaction for trading purposes to perform an unauthorised transaction. Unless the private key is compromised, this attack will not happen. Validator contracts need manager_address to be passed as a parameter. This value is hardcoded into the contract code, so this contract would be executed by the manager account.
Remediation	Not required
Reference	Not applicable / Not available

4.2 LogicSig

Result	Negative
Description	The transactions need to be authorized by the sender by signing them before they are sent to the network. The signed transaction object includes the transaction and a type of signature. There are three types of signatures used for the transactions which are as follows: 1. Single Signatures 2. Multi Signatures 3. Logic Signatures In Silo protocol smart contract, the Logic signature is being used to sign the transaction.
Source File	Not applicable / Not available

Observation	On line 121, there is a validation to make sure the address that is being used for authorising the transactions belongs to the sender.
	
	algoptions-main/algoptions/util.py from lines 118 to 122
	
	def signWithLogicSig(self, logicsig):
	    address = logicsig.address()
	    for i, txn in enumerate(self.transactions):
	        if txn.sender == address:
	            self.signed_transactions[i] = LogicSigTransaction(txn, logicsig)

Remediation	Not applicable / Not available
Reference	https://developer.algorand.org/docs/get-details/transactions/signatures/#logic-signatures

4.3 Decimal Calculation

Result	Negative
Description	Decimal calculations are critical in financial industries and other scientific areas as miscalculations or assumptions could be costly and could have devastating effects. This could go wrong, especially in the scenarios where there are conversions in the source code that convert the values from one type

	to another type. During the conversion process, there will be a marginal financial loss for the customer or for the business. These marginal financial differences could add up to huge losses when the transactions are scaled.
Source File	algoptions-main/algoptions/operation_transactions/execute.py Line 16, 17, 81, 82
Observation	No issues were observed in the exercise_fee and platform_fee calculations.
Remediation	Not applicable / Not available
Reference	https://developer.algorand.org/docs/get-details/transactions/signatures/

4.4 ReKeyTo

Result	Negative
Description	Rekeying is a powerful protocol feature that enables an Algorand account holder to maintain a static public address while dynamically rotating the authoritative private spending key(s).
Source File	algoptions/algoptions/contracts/escrow_logicsig.py
Observation	ReKeyTo is observed in the following location: algoptions/algoptions/contracts/escrow_logicsig.py Txn.rekey_to() == Global.zero_address(),--line 38
Remediation	Not applicable / Not available
Reference	https://developer.algorand.org/docs/get-details/dapps/smart-contracts/apps/#allowed-transaction-properties https://developer.algorand.org/docs/get-details/accounts/rekey/

4.5 Input Validations

Result	Negative
Description	Input validations are required to make sure that the smart contract code always runs as expected. Insufficient input validations could lead to unexpected results or could be risky behavior.
Source File	Not applicable / Not available
Observation	Sufficient input validations are observed across the code base. No risk is observed in terms of input validations in the smart contracts.
Remediation	Not required
Reference	Not applicable / Not available

4.6 Freeze/Clawback Address

Result	Negative
Description	The Freeze and Clawback addresses are required to prevent any financial loss. Freeze would allow your smart contract to freeze a particular account from performing any transaction in case of malicious activities. And Clawback address would allow the smart contract to revoke the rewards, incentives, and tokens from the user back to a clawback address in case of malicious activity.
Source File	algoptions/algoptions/contracts/call_validator_smart_contract.py algoptions/algoptions/contracts/put_validator_smart_contract.py
Observation	The freeze and clawback address code is observed in the following locations: algoptions/algoptions/contracts/call_validator_smart_contract.py Gtxn[2].config_asset_clawback() == Global.zero_address(),--line 85 Gtxn[3].config_asset_clawback() == Global.zero_address(),--line 92 algoptions/algoptions/contracts/put_validator_smart_contract.py Gtxn[2].config_asset_clawback() == Global.zero_address(),--line 84 Gtxn[3].config_asset_clawback() == Global.zero_address(),--line 91 algoptions/algoptions/testing/resources.py freeze=account.getAddress(),--line 105 clawback=account.getAddress()--line 106

Remediation	Not applicable / Not available
Reference	https://developer.algorand.org/docs/clis/goal/asset/freeze/

4.7 Proxy Assessment

Result	Negative
Description	The proxy contract is required to make sure that any future changes in the smart contract code will not affect the existing users or platforms that need to be notified. Making any changes to the existing smart contract needs to be deployed with a new contract address and it heavily impacts the maintenance activities of a developer.
Source File	Not applicable / Not available
Observation	Proxy code is not observed. But as this is not widely observed in the Algorand smart contracts, this is not considered as a security risk.
Remediation	Not applicable
Reference	https://ethereum.stackexchange.com/questions/114809/what-exactly-is-a-proxy-contract-and-why-is-there-a-security-vulnerability-invol

4.8 Fee and Amount Check

Result	Negative
Description	Fee and amount checks are critical for DeFi-based dApps. Missing to do these validations could have a financial impact on the users and the business. Based on the Algorand developer guidelines for TEAL, a fee should be less than some reasonable amount (in micro-Algos). An unchecked fee could burn the entire value of the contract account! Based on the information available on the Internet there is no fee charged for obtaining option tokens, besides transaction fees (0.003 ALGO). A small fee is charged for exercising options - the majority of which is used to reward collateral providers.
Source File	algoptions/algoptions/contracts/call_validator_smart_contract.py algoptions/algoptions/contracts/put_validator_smart_contract.py
Observation	The following fee validations are available in Call_validator_smart_contract.py and put_validator_smart_contract.py contract code algoptions/algoptions/contracts/call_validator_smart_contract.py Line 159: Gtxn[0].amount() == Gtxn[1].fee() + Gtxn[2].fee() + Gtxn[3].fee() Line 210: Gtxn[0].amount() == Gtxn[1].fee() + Gtxn[2].fee() + Gtxn[3].fee() Line 285: Gtxn[0].amount() == Gtxn[1].fee() + Gtxn[2].fee() + Gtxn[3].fee() + Gtxn[4].fee() + Gtxn[5].fee() Line 339: Gtxn[0].amount() == Gtxn[1].fee() + Gtxn[2].fee() + Gtxn[3].fee() algoptions/algoptions/contracts/put_validator_smart_contract.py Line 150: Gtxn[0].amount() == Gtxn[1].fee() + Gtxn[2].fee() + Gtxn[3].fee() Line 187: Gtxn[0].amount() == Gtxn[1].fee() + Gtxn[2].fee() + Gtxn[3].fee() Line 259: Gtxn[0].amount() == Gtxn[1].fee() + Gtxn[2].fee() + Gtxn[3].fee() + Gtxn[4].fee() + Gtxn[5].fee() Line 313: Gtxn[0].amount() == Gtxn[1].fee() + Gtxn[2].fee() + Gtxn[3].fee()
Remediation	Not required
Reference	Not applicable / Not available

4.9 Pragma Version

Issue Result	Negative
Description	Without the pragma version definition, the contract will be interpreted as a version 1 contract.
Source File	algoptions/algoptions/contracts/call_validator_smart_contract.py algoptions/algoptions/contracts/escrow_logicsig.py algoptions/algoptions/util.py algoptions/algoptions/contracts/call_validator_smart_contract.py
Observation	In the contracts, the pragma version 5 is being used. As it is defined in the code, there is no issue observed. algoptions/algoptions/contracts/call_validator_smart_contract.py-line-379 compiled = compileTeal(call_approval_program(10458941, 'UWPTQP34HYROPOJIFGHF3F6NIZRH4HN43H7PHNL3KHRJBN7RE3UY4DB464'), mode=Mode.Application, version=5) algoptions/algoptions/contracts/escrow_logicsig.py-line-84 compiled_contract_program_str = compileTeal(logicsig(), mode=Mode.Signature, version=4) algoptions/algoptions/util.py-line-58 teal = compileTeal(contract, mode=Mode.Application, version=5) algoptions/algoptions/contracts/call_validator_smart_contract.py-line-383 compiled = compileTeal(call_clear_state_program(), mode=Mode.Application, version=5)
Remediation	Not required.
Reference	https://developer.algorand.org/docs/get-details/dapps/avm/teal/guidelines/

4.10 Group Size Validation

Result	Negative
Description	Atomic Transfers are irreducible batch transactions that allow groups of transactions to be submitted at one time. If any of the transactions fail, then all the transactions will fail. PyTeal allows programs to access information about the transactions in an atomic transfer group using the Gtxn object. It is a best practice to have group size validation to make sure it is as per the expectation of the number of transactions.
Source File	algoptions-main\algoptions\contracts\call_validator_smart_contract.py Lines 27, 49, 140, 191, 258, 333 algoptions-main\algoptions\contracts\put_validator_smart_contract.py Lines 26, 48, 126, 164, 235, 307

Observation	GroupSize validation is observed in the smart contract code in multiple instances.
	
	Sample observations from one of the above instances are as follows:
	
	compiled_contract_program_str = compileTeal(logicsig(), mode=Mode.Signature, version=4)
	teal = compileTeal(contract, mode=Mode.Application, version=5)

Remediation	Gtxn is zero-indexed and the maximum size of an atomic transfer group is 16. Make sure to validate that the size of the atomic transfer group does not exceed this value.
Reference	https://pyteal.readthedocs.io/en/v0.6.2/accessing_transaction_field.html#atomic-tranfer-groups

4.11 AlertHub Setup

Result	Negative
Description	Alerts would enable real-time monitoring of status and activity on your account. There are currently five types of alerts that when configured could provide visibility and insights to act on any important events.
Source File	Not applicable / Not available
Observation	Silo Defi team has deployed metrika for alerts on their account.
Remediation	Not required
Reference	https://developer.algorand.org/tutorials/monitoring-account-activity-alerthub/#5-respond-to-any-firing-alert

5.0 Auditing Approach and Methodologies applied

Throughout the audit of the smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Fee calculations are as per the intended behavior mentioned on the website.
- Code is safe from vulnerabilities.

The following phases and associated tools were used throughout the term of the audit:

5.1 Structural Analysis

In this step, we have analysed the design patterns and structure of all smart contracts. A thorough check was completed to ensure all Smart contracts are structured in a way that will not result in future problems.

5.2 Code Review / Manual Analysis

Manual Analysis or review is done to identify new vulnerabilities or to verify the vulnerabilities found during the Static Analysis. The contracts were completely manually analysed, and their logic was checked and compared with the one described in the whitepaper. It should also be noted that the results of the automated analysis were verified manually.

5.3 Tools Used for Audit

No tools were used during this audit.

5.4 Recommendations

Aside from undertaking a Smart Contract Audit, Entersoft highly recommends Silo Defi to explore activities such as:

- Penetration testing on the dApp layer including front-end Web UI
- Cloud Security Audits
- Network Penetration Testing
- Other Cyber Security activities such as uplifting controls and processes, cyber awareness and training, etc.

Such activities will help Silo Defi uplift their cyber security posture whilst providing confidence to key stakeholders.

6.0 Limitations on Disclosure and Use of this Report

This report contains information concerning potential details of Silo protocol smart contract and methods for exploiting them. Entersoft recommends that special precautions be taken to protect the confidentiality of both this document and the information contained herein. Security Assessment is an uncertain process, based on past experiences, currently available information, and known threats. All information security systems, which by their nature are dependent on human beings, are vulnerable to some degree. Therefore, while Entersoft considers the major security vulnerabilities of the analyzed systems to have been identified, there can be no assurance that any exercise of this nature will identify all possible vulnerabilities or propose exhaustive and operationally viable recommendations to mitigate those exposures. In addition, the analysis set forth herein is based on the technologies and known threats as of the date of this report. As technologies and risks change over time, the vulnerabilities associated with the operation of the Silo protocol smart contract described in this report, as well as the actions necessary to reduce the exposure to such vulnerabilities will also change. Entersoft makes no undertaking to supplement or update this report based on changed circumstances or facts of which Entersoft becomes aware after the date hereof, absent a specific written agreement to perform the supplemental or updated analysis. This report may recommend that Entersoft use certain software or hardware products manufactured or maintained by other vendors. Entersoft bases these recommendations upon its prior experience with the capabilities of those products. Nonetheless, Entersoft does not and cannot warrant that a particular product will work as advertised by the vendor, nor that it will operate in the manner intended. This report was prepared by Entersoft for the exclusive benefit of Silo Defi and is proprietary information. The Non-Disclosure Agreement (NDA) in effect between Entersoft and Silo Defi governs the disclosure of this report to all other parties including product vendors and suppliers.