VANTAGEPOINT
security at the speed of development

Smart Contract Audit of Folks Finance Cross-Chain Liquid Governance

v1.0
Final Report

Prepared for:

Folks Global LTD

19 Jun 2023

FOR GENERAL DISTRIBUTION

Table Of Content

1. Executive Summary

1.1. Overview

Vantage Point Security Pte Ltd was engaged by Folks Finance to conduct an Algorand smart contract audit of Cross-Chain Liquid Governance. Folks Finance incorporated xAlgo to overcome the issue with committed algos becoming illiquid. To allow non-Algorand users to participate and enjoy the governance rewards, Folks Finance introduced cross-chain governance protocol which allows minted xAlgo tokens to be bridged to the Binance Smart Chain.

Supporting Document

- Cross-Chain Liquid Gov Technical Design

PyTeal Code Repository

- https://github.com/blockchain-italia/ff-vp-contracts

Vantage Point performed this audit by first understanding the Folks Finance Cross-Chain Liquid Governance protocol's business logic based on the document provided. We sought clarifications on potential issues, discrepancies, flaws and plans on how manual operations involved would be carried out through discussions with the Folks Finance team. The smart contract revision audit was conducted on the provided PyTeal code to identify any weaknesses, vulnerabilities, and non-compliance to Algorand best practices. As part of the audit process, Vantage Point executed a set of baseline test cases tailored to the smart contracts in scope. During the course of the project, Vantage Point Security identified two observational findings that require attention:

- Insufficient Validation of Transaction Fields
- Insufficient / Lack of Validation for Time-sensitive Operations

These failed test cases were documented as findings.

Insufficient / Lack of Validation for Time-sensitive Operations

- In the governance.py smart contract, insufficient validations were observed for `update_commit_end` and `update_fee` operations as they either did not enforce validation for defined business logic within the smart contract or the lacked consideration for potential abuse by the admin within the business logic.

Insufficient Validation of Transaction Fields

- In the governance.py smart contract, the subroutine `mint_x_algo` sends xAlgo to `Txn.sender()` using the subroutine `get_transfer_inner_txn` from inner_txn.py. `get_transfer_inner_txn` switches between algo transfer and asset transfer based on the value of asset_id supplied. Without any validation for the asset_id - `App.globalGet(x_algo_id_key)`, a zero value could be supplied into asset_id which may result in transfer of algos instead of xAlgos.

The outcome of this Algorand smart contract audit is provided as a detailed technical report that provides the project owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendation that will resolve the identified technical issue.

2. Project Details

2.1. Scope

App Name	Smart Contract Audit of Folks Finance Cross-Chain Liquid Governance
Testing Window	June 2023
Svn / Git Revision Number	c4c9cae9c0a8a77b4d17ee4a3eda3d82a5b7606b
Project Type	Smart Contract Audit
App Type	Algorand Smart Contract
Smart Contract Administration Structure	In scope smart contract had an administration structure where the administrator address is stored as a global state and privileged operations require Txn.sender() to match the global state value. **Admin Account** • Stored as a global state with `App.globalPut(admin_key, admin.get())` during `create()` operation • Updated through `update_admin()` operation **Privileged Operations** • `setup()` • `update_admin()` • `check_admin_call()` • `schedule_update_sc()` • `update_sc()` • `update_commit_end()` • `update_fee()` • `claim_fee()` • `pause_minting()` • `governance()` As noted above, the admin address has access to privileged operations of xGovernance smart contract. In response to the posed risk, Folks.Finance team has implemented following compensating measures. • Admin account is a multisignature • Privileged operations that directly affect the value of xAlgo or logic of xGovernance smart contract is delayed by at least 24 hours, to provide sufficient window for xAlgo holders to make an informed decision Vantage Point Blockchain also has confirmed Folks.Finance team's plan to shift to a DAO or a similar structure where such controls can be owned by a decentralized entity.

Items Completed	algo_governance\governance.py algo_governance\governance_state.py
Issue Opening Date	**12 Jun 2023** ● Insufficient / Lack of Validation for Time-sensitive Operations [Low] **10 Jun 2023** ● Insufficient Validation of Transaction Fields [Observational]
Issue Closing Date	**14 Jun 2023** ● Insufficient / Lack of Validation for Time-sensitive Operations [Low] **13 Jun 2023** ● Insufficient Validation of Transaction Fields [Observational]

2.1. Status

Component	Review Type	Status
Algorand Smart Contract	Smart Contract Audit	Completed
Algorand Smart Contract	Retest 0	Completed
Algorand Smart Contract	Retest 1	Completed
Algorand Smart Contract	Retest 2	Completed
Algorand Smart Contract	Retest 3	Completed

3. Risk Assessment

This chapter contains an overview of the vulnerabilities discovered during the project. The vulnerabilities are sorted based on the scoring categories CRITICAL, HIGH, MEDIUM and LOW. The category OBSERVATIONAL refers to vulnerabilities that have no risk score and therefore have no immediate impact on the system.

3.1. Summary of Vulnerabilities

Vulnerability Title	Risk Score	Closed
Insufficient / Lack of Validation for Time-sensitive Operations	Low	☑
Insufficient Validation of Transaction Fields	Observational	☑

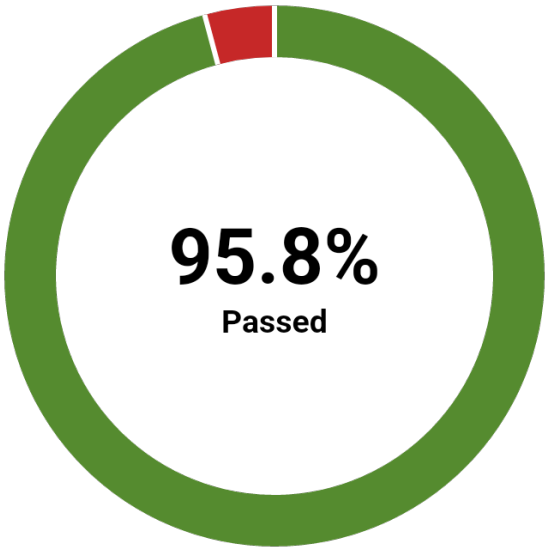
3.2. Vulnerabilities Statistics

Findings Overview

Total	Open	Closed
2	0	2

0	Critical	No findings
0	High	No findings
0	Medium	No findings
1	Low	0 Open, 1 Closed.
1	Observational	0 Open, 1 Closed.

Test Case Status



Open	0.0%
Passed	95.8%
Failed	4.2%

4. Detailed Description of Vulnerabilities

4.1. Insufficient / Lack of Validation for Time-sensitive Operations

Low

Closed

BACKGROUND

Business logic of smart contracts may include operations or functions which can only be approved when specific time-related conditions such as timestamps, rounds and states are met. Validating such conditions prior to approval of application calls are essential in enforcing the business logic.

DESCRIPTION

Affected File/Code

- governance.py [update_commit_end]
- governance.py [update_fee]

Instance 1

Based on provided documentation, the `governance.py` smart contract's `update_commit_end` operation has following requirements.

- `update_commit_end` is a privileged operation which is reserved for the admin address
- `update_commit_end` will be called only after Algorand Governance rewards have been distributed to the smart contract

While the subroutine `check_admin_call()` sufficiently validates if the `Txn.sender()` is an admin address, there is no validation that verifies if the `update_commit_end` is being called after Algorand Foundation's governance rewards have been distributed.

In a situation where the administrator is malicious or the admin's wallet has been compromised, this potentially allows the admin to make abusive use of the smart contract at the expense of xAlgo holders using transaction groups such as below.

Malicious Transaction

- Gtxn[0] - `update_fee` - Increases the fees claimable against the total xAlgo supply (up to 5%)
- Gtxn[1] - `claim_fee` - Mint xAlgo at an increased fee rate based on Gtxn[0] at the expense of existing xAlgo holders. Higher fee rate decreases the value of xAlgo further as more xAlgo is minted as a fee for the admin.
- Gtxn[2] - `update_commit_end` - Update the commit_end_key value to so that the malicious transaction group can be executed again

Instance 2

Based on the provided documentation, the governance.py smart contract's update_fee operation has following requirements.

- `update_fee` is a privileged operation which is reserved for the admin address

However, based on the safeguard measures taken for other admin operations that impacts the value of xAlgo, the `update_fee` operation does not have any time-based enforcement on when it can be used to update the fees.

In a situation where the administrator is malicious or the admin's wallet has been compromised, this potentially allows an admin to make abusive use of the smart contract at the expense of xAlgo holders. Following example transaction group can be used to increase the fees to the level which may be higher than published rates right before executing `claim_fee`.

Malicious Transaction

- Gtxn[0] - `update_fee` - Increases the fees claimable against the total xAlgo supply (up to 5%)
- Gtxn[1] - `claim_fee` - Mint xAlgo at an increased fee rate based on Gtxn[0] at the expense of existing xAlgo holders

IMPACT

Insufficient validation of business logic or flawed business logic may approve a group of operations which could allow the admin to either claim higher fees or claim fees more than once within a single governance period.

RECOMMENDATIONS

It is recommended to carefully review the time-based conditions that needs to be met for each function or operation within the smart contract. Enforce such conditions in scope through an assert, using most appropriate form of state that represents time on blockchain.

COMMENT

Reviewed on 13 Jun 2023

Based on discussion with Folks Finance team and commit 1f813433857544d225a15b2841b5a8ed4084b193, following points were noted. This issue is closed.

Instance 1

There is at least a day of delay from the time when `update_commit_end` is executed and when the `claim_fee` becomes callable again, providing a window of opportunitiy for the xAlgo holders to burn xAlgo and get Algo if they do not agree with the privileged operations executed

Instance 2

New time-related conditions are now required for `update_fee` is called by the admin where either of the following conditions need to be fulfilled.

- Condition 1 - There is more than "time delay" remaining till commit end
- Condition 2 - The fee already has been claimed for the current period

With the condition 1, it provides a window for existing and potential xAlgo holders to make an informed decision on if the updated fee. For condition 2, as the fee would have been claimed already at the agreed rate, fee changes do not affect the xAlgo holders for the current period. xAlgo holders who do not agree with the updated fees can burn the xAlgo once the period ends.

REFERENCES

CWE-367: Time-of-check Time-of-use (TOCTOU) Race Condition
https://cwe.mitre.org/data/definitions/367.html

4.2. Insufficient Validation of Transaction Fields

Observational

Closed

BACKGROUND

Smart contracts need to validation transaction field values to ensure expected arguments are being submitted as part of transaction field values. Insufficient or incorrect validation of such may result in financial loss or unexpected smart contract behaviors.

DESCRIPTION

Affected File/Code

- governance.py [mint_x_algo]

It was noted that the `governance.py` smart contract makes use of subroutine `mint_x_algo()` to send xAlgo to `Txn.sender()`. As the subroutine `get_transfer_inner_txn()` switches between Algo payment and ASA transfer based on the value of `asset_id` supplied, lack of validation for `App.globalGet(x_algo_id_key)` may result in a transfer of Algo instead of xAlgo.

However, upon further validation of the smart contract logic, the only possible path of `App.globalGet(x_algo_id_key).value()` returning 0 would be when the admin has not called the `setup()` method.

IMPACT

Smart contracts with insufficient or incorrect validation logic for their transaction fields may allow participants to have transactions with unexpected transaction field values approved. When transactions with unexpected values are wrongly approved by the smart contract, it may be abused by attackers for malicious purposes.

RECOMMENDATIONS

Based on the defined business logic and expected transaction field values, ensure sufficient validation logic is enforced prior to approving transactions or transaction groups.

For `mint_x_algo` subroutine, following can be considered.

- Plan the deployment in a such way that there is no point in time where users can call `mint()` before `setup()` is called.
- Use of `Assert(Not(App.globalGet(x_algo_id_key)==0)` within subroutine `mint_x_algo`
- Use of `Assert(App.globalGetEx(Int(0), x_algo_id_key).hasValue())` within subroutine `mint_x_algo`

COMMENT

Reviewed on 12 Jun 2023

Based on confimration with Folks Finance team, following points were noted.

- Folks Finance team would deploy the smart contract and call setup() prior to publishing the application

- Within the current context of the smart contract logic, even if non-admin user is able to call mint(), it is not possible to get more Algo than the amount transferred from Txn.sender() to the smart contract.

5. Appendix

5.1. Disclaimer

The material contained in this document is confidential and only for use by the company receiving this information from Vantage Point Security Pte. Ltd. (Vantage Point). The material will be held in the strictest confidence by the recipients and will not be used, in whole or in part, for any purpose other than the purpose for which it is provided without prior written consent by Vantage Point. The recipient assumes responsibility for further distribution of this document. In no event shall Vantage Point be liable to anyone for direct, special, incidental, collateral or consequential damages arising out of the use of this material, to the maximum extent permitted under law.

The security testing team made every effort to cover the systems in the test scope as effectively and completely as possible given the time budget available. There is however no guarantee that all existing vulnerabilities have been discovered. Furthermore, the security assessment applies to a snapshot of the current state at the examination time.

5.2. Risk Rating

All vulnerabilities found by Vantage Point will receive an individual risk rating based on the following four categories.

Critical

A CRITICAL finding requires immediate attention and should be given the highest priority by the business as it will impact business interest critically.

High

A HIGH finding requires immediate attention and should be given higher priority by the business.

Medium

A MEDIUM finding has the potential to present a serious risk to the business.

Low

A LOW finding contradicts security best practices and have minimal impact on the business.

Observational

An OBSERVATIONAL finding relates primarily to non-compliance issues,security best practices or are considered an additional security feature that would increase the security stance of the environment which could be considered in the future version of smart contract.