VANTAGEPOINT
security at the speed of development

Smart Contract Revision Audit of Folks Finance Liquid Governance V3

v1.0
Final Report

Prepared for:
Folks Global LTD

30 Mar 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 revision audit of Folks Finance Liquid Governance V3. Folks Finance had changes to the liquid governance because the current design was not compatible with xGov. Folks Finance proposed a solution where each liquid governance user will have their own escrow containing their own commitment. This lets a user decide whether to be an xGov and which address will receive rewards. The Algorand smart contract revision audit was conducted based on the following materials provided.

Supporting Document

- Liquid Governance V3 Documentation

PyTeal Code Repository

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

Vantage Point performed this audit by first understanding the Folks Finance Liquid Governance V3 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. Multiple test cases failed namely,

- Incorrect Implementation of Smart Contract Logic
- Insufficient Validation for Transfer of Privilege through State Change
- Lack of Transaction Fee Validation for Logic Signatures Allows Algo Balance to be Drained
- Lack of pragma or Pragma for Smart Contracts with Router Class
- Inaccurate Comments

The failed test cases were documented as findings.

Incorrect Smart Contract Logic Implementation

- Liquid Governance V3 allows users to claim the pre-minted gAlgos only after the `PREMINT_END` period. The `claim_premint` operation however did not validate the condition of `Global.latest_timestamp() > PREMINT_END` and allowed users to perform `claim_premint` operation when the burning of gAlgo is active from the previous governance.

Insufficient Validation for Transfer of Privilege through State Change

- Liquid Governance V3 allows the `ADMIN` to update the timestamp for `PREMINT_END`, `COMMIT_END`, `IS_BURNING_PAUSED`, and update the address for `ADMIN` role. While these operations have validation if the `Txn.sender()` is the `ADMIN`, the administrator address entered was not verified if it's not a global zero address.

Other findings with risk rating of observational are discussed thoroughly in the detailed description of vulnerabilities section. The outcome of this Algorand smart contract revision 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 Revision Audit of Folks Finance Liquid Governance V3
Testing Window	21 Mar 2023 to 28 Mar 2023
Target URL	https://github.com/blockchain-italia/ff-vp-contracts
Svn / Git Revision Number	fdf98fe509e5853f7f3ae4aae852c5fb1082e40e
Project Type	Smart Contract Revision Audit
App Type	Algorand Smart Contract
Items Completed	algo_governance\distributor.py algo_governance\distributor_state.py algo_governance\distributor_escrow.py
Issue Opening Date	27 Mar 2023 ● Incorrect Implementation of Smart Contract Logic [Critical] ● Insufficient Validation for Transfer of Privilege through State Change [Observational] 24 Mar 2023 ● ARC-04 Smart Contract with Router Class without Pragma or pragma [Low] ● Inaccurate Comments [Observational] 28 Mar 2023 ● Lack of Transaction Fee Validation [Observational]
Issue Closing Date	28 Mar 2023 ● Incorrect Implementation of Smart Contract Logic [Critical] ● ARC-04 Smart Contract with Router Class without Pragma or pragma [Low] ● Inaccurate Comments [Observational] ● Insufficient Validation for Transfer of Privilege through State Change [Observational] 29 Mar 2023 ● Lack of Transaction Fee Validation [Observational]

2.2. Status

Component	Review Type	Status
Smart Contract Revision Audit	Smart Contract Revision Audit	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
Incorrect Implementation of Smart Contract Logic	Critical	☑
ARC-04 Smart Contract with Router Class without Pragma or pragma	Low	☑
Inaccurate Comments	Observational	☑
Lack of Transaction Fee Validation	Observational	☑
Insufficient Validation for Transfer of Privilege through State Change	Observational	☑

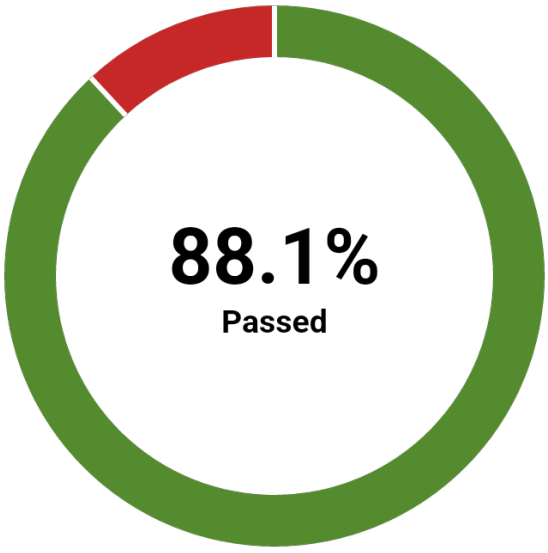
3.2. Vulnerabilities Statistics

Findings Overview

Total	Open	Closed
5	0	5

1 Critical	0 Open, 1 Closed.
0 High	No findings
0 Medium	No findings
1 Low	0 Open, 1 Closed.
3 Observational	0 Open, 3 Closed.

Test Case Status



Open	0.0%
Passed	88.1%
Failed	11.9%

4. Detailed Description of Vulnerabilities

4.1. Incorrect Implementation of Smart Contract Logic

Critical

Closed

BACKGROUND

The smart contract logic implements the protocol's business logic in code to ensure expected operations are carried out according to the documented requirements. With incorrect implementation of such logic, the smart contract may approve transactions which are not allowed or behave in an unexpected way.

DESCRIPTION

Affected File/Code

- distributor.py [claim_premint]

The `claim_premint` operation does not validate if `premint_end` has passed before allowing `claim_premint` operation. As such validation is not in place, it could allow attackers to repeatedly perform the following.

1. Commit 100 Algo into the escrow through `mint` operation.
   - The attacker's escrow balance will be equal to 100 Algo.
2. Call `claim_premint` operation to get 100 gAlgo.
3. Call `burn` operation to exchange 100 gAlgo for 100 Algo from the previous governance smart contract.
4. Commit 100 Algo into the escrow through `mint` operation.
   - The attacker's escrow balance will be equal to 200 Algo.

IMPACT

The attacker can commit significantly higher amount of Algo into the escrow for governance and get rewarded. In addition, this also means that the Algo balance kept at the dispenser would be insufficient to meet the burning operations called by legitimate users of Folks.Finance as illegally minted gAlgo could have been burned first and drained the Algo balance of the dispenser smart contract.

RECOMMENDATIONS

Review the smart contract's logic based on the requirements and ensure the smart contract's actual implementation is sufficiently accurate based on what is available on-chain. Ensure there is no discrepancy between published formulae, logic, and the smart contract's behavior. In this case, a validation to verify if the pre-mint period has ended should be implemented.

COMMENT

Reviewed on 28 Mar 2023

Based on the commit efe948df48cc555e7de09ee3cb71b401dca18b09, a validation was added to only allow the claim_premint operation to be called after the pre-mint period. This issue is closed.

4.2. ARC-04 Smart Contract with Router Class without Pragma or pragma

Low

Closed

BACKGROUND

ARC-04 Algorand Smart Contract Transaction Calling Conventions is a standard for encoding smart contract's method calls. ARC-04 allows wallets and dapp frontends to properly encode call transactions based on the interface description and explorers to show details of invoked methods. Smart contracts complying to ARC-04 standards respond to both method calls and bare app calls. To make it easier for an application to route across many base app calls and methods, PyTeal introduced Router class which adheres to the ARC-4 ABI conventions with respect to when methods and base app calls. As Router class is still expecting backwards-incompatible changes, it is recommended use pragma/Pragma expression to pin the version of PyTeal in the source code.

DESCRIPTION

Affected File/Code

- distributor.py

The smart contract in scope is an ARC-04 compliant smart contract which makes use of PyTeal's Router class for routing bare and method application calls. To ensure backward-incompatible changes do not affect the behavior of the smart contract, it is recommended to pin the PyTeal compiler version. No pragma was found in the affected smart contracts to pin the PyTeal compiler versions.

IMPACT

As PyTeal's Router class may have backward-incompatble changes in the future, if the PyTeal compiler version is not pinned, this PyTeal smart contract may compile in an expected way which may result in an unwanted routing of application calls.

RECOMMENDATIONS

It is recommended to make use of Pragma or pragma to pin the version of PyTeal in the source code to cater for potential backward-incompatible changes to the Router class. The correct range of the compiler version should be determined and specified in the code.

COMMENT

Reviewed on 28 Mar 2023

Based on the commit 36097eba7c3e1b5dbc6ae0643b67d70f15901f72, pragma was added. This issue is closed.

REFERENCES

PyTeal - ABI Support
https://pyteal.readthedocs.io/en/stable/abi.html?highlight=Router#abi-support

PyTeal - Pragma
https://pyteal.readthedocs.io/en/stable/api.html?highlight=Router#pyteal.Pragma

4.3. Inaccurate Comments

Observational

Closed

BACKGROUND

Code comments play a vital role in providing more human-readable context to the readers of the code so that technical specifications, transaction details and the expectations can be clearly conveyed to the users or future developer team members. Ensuring

Inaccurate comments could cause confusion to the developers and users who rely on the information provided in understanding the logic of smart contracts. Confusion in the business logic and technical specifications may not be critical in the short-term but for the continuity and maintainability of the smart contract projects, it is important to have the most updated details reflected in comments.

DESCRIPTION

Affected File/Code:

- distrbutor.py [update_end]

It was noted that the smart contract code had incorrect comments for the `update_end` operation. The `update_end` operation mentions through a comment that the verifier is an old admin. However, the operation can be carried out by the current admin calling the function.

IMPACT

Inaccurate comments could cause confusion to the developers and users who rely on the information provided in understanding the logic of smart contracts. Confusion in the business logic and technical specifications may not be critical in the short-term but could potentially end up with a wrong implementation which may have rooted from a wrong understanding of the existing smart contract logic.

RECOMMENDATIONS

Update comments according to the latest business logic and technical specification to ensure the understanding of the developers who read the code are coherent.

COMMENT

Reviewed on 28 Mar 2023

Based on the commit 36097eba7c3e1b5dbc6ae0643b67d70f15901f72, the comment was modified to `# verify caller is admin`. This issue is closed.

4.4. Lack of Transaction Fee Validation

Observational

Closed

BACKGROUND

Algorand transactions have a minimum transaction fee of 1000 microAlgos per each transaction and this amount may change if the network gets congested and variable transaction fees per byte are used instead. For both logic signatures and smart contracts, validating transaction fees are crucial to in making sure the balances of logic signatures or smart contracts are not drained and also to ensure transactions can be approved even in cases where there are increased transaction fees due to network congestion.

DESCRIPTION

Affected File/Code:

- distributor_escrow.py [lsig_program]

It was noted that the logic signature above did not have sufficient validation for the transaction fee that may allow draining of the logic signature balance.

IMPACT

Having incorrect or insufficient validation for transaction fee may cause rejection of legitimate transactions during network congestion or allow draining of smart contract or logic signature balance

RECOMMENDATIONS

It is recommended to inform the user of the following key points:

- A payment transaction from the owner prior to the logic signature's transaction is sufficient for the logic signature's approval. This is applicable if the logic signature was to be used outside the context of Folks Finance Algo Governance V3, as a general escrow.
- The transaction fee is not validated as the escrow is expected to be re-keyed to the distributor smart contract throughout its lifecycle. Escrow owners are to review transactions closely before approving when it involves the escrow.

COMMENT

Reviewed on 29 Mar 2023

Details about the escrow was added to the document. This issue is closed.

4.5. Insufficient Validation for Transfer of Privilege through State Change

Observational

Closed

BACKGROUND

Algorand smart contracts may have methods available to initialize, setup, or transfer ownership of privileged roles to another address. As such, privileged addresses play a vital role in managing the smart contract's critical configurations. It is beneficial to have a validation that prevents invalid addresses such as global zero address being used as a new privileged address.

DESCRIPTION

Affected File/Code

- distributor.py [update_admin]

It was noted that the distributor smart contract's `update_admin` operation makes changes to the `new_admin` global state that stores a privileged address critical to the smart contract's operation. However, insufficient validation was noted as the address could be set to a global zero address.

IMPACT

As privileged addresses stored on-chain perform roles which are essential to smart contract or protocol's operation, any transactions sent with wrong account values such as `Global.zero_address()` could have significant impact to proper operation of the smart contract and the protocol resulting in a loss due to inability to update critical configuration parameters in time.

RECOMMENDATIONS

For operations that commit or make changes to existing on-chain states such as local, global or box storage, ensure the address is validated for the following conditions at the minimum.

- Length of 32 bytes
- Not a global zero address

COMMENT

Reviewed on 28 Mar 2023

Based on confirmation from Folks Finance team, the validation for address length exists and administrator addresses are set with multi-signatures which require multiple parties to review and approve the address being submitted as a new admin address. Existing processes of the Folks Finance team sufficiently addresses the issue. This issue is closed.

REFERENCES

PyTeal Documentation - Global Zero Address
https://pyteal.readthedocs.io/en/stable/api.html#pyteal.Global.zero_address

PyTeal Documentation - Registering Methods

https://pyteal.readthedocs.io/en/stable/abi.html?highlight=abi.address#registering-methods

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.