Smart Contract Audit

Folks Finance

coinspect

FOLKS FINANCE

Folks Finance

Smart Contract Audit

V220314

Prepared for Blockchain Italia • March 2022

1. Executive Summary

In **February 2022, Folks** engaged Coinspect to perform a source code review of **Folks Finance**. The objective of the project was to evaluate the security of the smart contracts.

The assessment was conducted on the contracts from the Git repository at https://github.com/blockchain-italia/ff-coinspect-contracts. The audit started on commit `bb0fa214d5b6dc51a2d32f49ccff2560cb3f83cc` as of February 14th. Additionally, some last minute changes were introduced on February 24th in commit `c7c8942f2a40f19bae757dc8a3ea9f80b4c521fb`.

Coinspect found the smart contracts to be properly designed. The extensive documentation provided contributed to the audit process and the test suite quality was found to be above average.

The following issues were identified during the assessment:

High Risk	Medium Risk	Low Risk
2	**1**	**0**
Fixed 2	Fixed 1	Fixed 0

2. Assessment and Scope

The audit started on February 14, 2022 and was conducted over the files located at the https://github.com/blockchain-italia/ff-coinspect-contracts git repository on the main branch, as of commit `bb0fa214d5b6dc51a2d32f49ccff2560cb3f83cc`. The files have the following sha256sum hash:

```
4522f448fe19059c04e8f9abdceda122c2f992c4ef094fb23140402e0b312ea8  staking_clear_program.py
b9e4c7465a039c6643e7695dc418d2ebc4270bb8a3047fa73d557b5f86b27238  rewards_aggregator_approval_program.py
a85b24d255e3bf3b2fd913ea581d5ccaabf2d99bb7f53f95caeb71a21eeb5b49  liquidity_approval_program.py
be3adb83133f632d587fb400e6391efe906d2e93df995cd4978e58c08afef727  token_pair_approval_program.py
b93815d82c9d407bfce212cbbe977b29023025c8b52efe9954e85d3db2ee11c3  oracle_adapter_approval_program.py
3c264ddd3f2ff276b7994787e5d498a205cfaa7c30b330312d4a0f0eba558e72  clear_program.py
e60ecda4cad08185cbb1e5feede89a139f3a08b05d7b16fe572d7ff18a3fa832  oracle_adapter/shared.py
4864e373808bd32f7cb0651751c6783a6567997eccde24266b1ef1ef6c682558  oracle_adapter/state.py
32a746fb68c97a3c0fc84f3ec138c23dcc349cfa7bb2020a76f2093aefb63d9a  dispenser_approval_program.py
17240d5f645f74cc7b1042395405a929116ad5d959d16caa0d5b0dcb85cfde4d  oracle_approval_program.py
fb198b5401d49ba362d439b078b98d1412d235411caa21669690414ef42f9a04  common/math.py
b2421636d967e4c14b7362f33ffbddb84c4403f7ba0db024ba378c23dfc71a69  common/formulae.py
5546a40a43777cd7a48fb06729c4add2533e1cacb27d588bcbf0d20407d83088  common/inner_txn.py
62f6c51f111428ce2a5c5852c83595f05e14900e5a356e295580d9c3e2181ae5  common/transactions.py
e88d80032be64013f18fd7d05af080a5f48272efc89e4e56f445e615aee085ed  staking_approval_program.py
b9cd0a188e24d4983db9b8df9b4f4c8bdbf1c2dca89105db918a8c4dd388fe2b  dispenser/shared.py
52ec53b79af1e8ec9852895d1f032fd6bd887e5b60ef813dd6bdf602a3d5664b  dispenser/state.py
```

Folks Finance implements a money market protocol on the Algorand blockchain that allows users to accrue interests on their deposits and borrow assets against provided collateral.

Coinspect encountered no issues with the overall protocol design but found some issues with the implementation. Constants are repeated all over the source code, making it more error-prone and harder to read and comprehend. This bad practice led to issues FF-1 and FF-3.
Issue FF-2 allows attackers to bypass staking periods limits and exchange all their rewards tokens.

On February 24th support for the Algorand Foundation Governance was introduced in commit `c7c8942f2a40f19bae757dc8a3ea9f80b4c521fb`. Coinspect reviewed these changes but focused on the contracts related to the new feature. The files have the following sha256sum hash:

```
9ba08ea76c943190fd78bfed0749e0cb5f778c599850bc73853183781e07b40a  algo_governance/state.py
a501ee998dae9927ce812f5e1f2fa587911d6654e9657a09f517af3ad2db7e99  algo_governance/transactions.py
1d5a8cde50b38efa6fb9f318ae0fbdfb6c44f17ce3b54a56801158025c955103  algo_governance_approval_program.py
6dddc6199a7173976036cff910969dbfe0816e2eba1c0a77d44ac4e06e280186  algo_governance_contract_account.py
80938f968589b40fe8a1a4291c8a80eefe62d94dc3fb1705ea42cd3617c0e2dc  algo_governance_clear_program.py
```

Coinspect did not find any issues with these changes except for some informational ones (FF-4 and FF-5).

As of March 7, commit `b1e0a928df809eea7f632405bc1e9c284cffa94e` was reviewed with the following files with their respective sha256sum:

```
b1e1002acfe57bd344dfa69f0fd085f51b3ebfc30605612f56569e1e909cd14f  staking_clear_program.py
120c6f725960f9da92c8ee0221e4f471b9de6fbc6738d69dbc32d1cfcf4b080e rewards_aggregator_approval_program.py
d8762afbcb12f0152233bf3ea24c6d68d9377832b44dd17d7b60b0a28fe715cb  staking/state.py
a85b24d255e3bf3b2fd913ea581d5ccaabf2d99bb7f53f95caeb71a21eeb5b49  liquidity_approval_program.py
6dddc6199a7173976036cff910969dbfe0816e2eba1c0a77d44ac4e06e280186  algo_governance_contract_account.py
be3adb83133f632d587fb400e6391efe906d2e93df995cd4978e58c08afef727  token_pair_approval_program.py
9ba08ea76c943190fd78bfed0749e0cb5f778c599850bc73853183781e07b40a  algo_governance/state.py
a501ee998dae9927ce812f5e1f2fa587911d6654e9657a09f517af3ad2db7e99  algo_governance/transactions.py
b93815d82c9d407bfce212cbbe977b29023025c8b52efe9954e85d3db2ee11c3  oracle_adapter_approval_program.py
3c264ddd3f2ff276b7994787e5d498a205cfaa7c30b330312d4a0f0eba558e72  clear_program.py
8cc10b06d53937e3dfff899159567f36826d3c5881ccde1acd82860e70e5c9d2  algo_governance_approval_program.py
e60ecda4cad08185cbb1e5feede89a139f3a08b05d7b16fe572d7ff18a3fa832  oracle_adapter/shared.py
4864e373808bd32f7cb0651751c6783a6567997eccde24266b1ef1ef6c682558  oracle_adapter/state.py
deced2e8fa56d1ef3d7d41916acfe187d4371c6920fe97274543c95f9ce8116b  dispenser_approval_program.py
80938f968589b40fe8a1a4291c8a80eefe62d94dc3fb1705ea42cd3617c0e2dc  algo_governance_clear_program.py
6ad7933a1ccd36fc226d1d4ffebd5a6b6d9194a1257bc617320749defb69644a  oracle_approval_program.py
fb198b5401d49ba362d439b078b98d1412d235411caa21669690414ef42f9a04  common/math.py
b2421636d967e4c14b7362f33ffbddb84c4403f7ba0db024ba378c23dfc71a69  common/formulae.py
5546a40a43777cd7a48fb06729c4add2533e1cacb27d588bcbf0d20407d83088  common/inner_txn.py
62f6c51f111428ce2a5c5852c83595f05e14900e5a356e295580d9c3e2181ae5  common/transactions.py
e5f51e5c48dc1c3e5dacf7231ea2a5c8832ed95b30b7f228241d502e86bf55a8  staking_approval_program.py
b9cd0a188e24d4983db9b8df9b4f4c8bdbf1c2dca89105db918a8c4dd388fe2b  dispenser/shared.py
52ec53b79af1e8ec9852895d1f032fd6bd887e5b60ef813dd6bdf602a3d5664b  dispenser/state.py
```

Coinspect verified that it correctly addresses the issues.

3. Summary of Findings

Id	Title	Total Risk	Fixed
FF-1	Zero frAssets awarded on staking claim	High	✔
FF-2	frAsset price dump	High	✔
FF-3	Abandoned stakes are not recoverable	Medium	✔
FF-4	Missing validation on_creation	Info	✔
FF-5	Misleading on_governance documentation	Info	✔

4. Detailed Findings

FF-1	Zero frAssets awarded on staking claim

Total Risk	Impact	Location
High	High	`assets/staking_approval_program.py`
Fixed	Likelihood	
✔	High	

Description

Calling `on_setup_staking` does not correctly set the total rewards amount for the staking resulting in zero rewards.

The variable set is "`rewards`" instead of "`total_rewards`".

Recommendation

Change "`rewards`" to "`total_rewards`" and define constants to avoid future errors.

FF-2	frAsset price dump	
Total Risk **High**	Impact High	Location `assets/rewards_aggregator_approval_program.py`
Fixed ✔	Likelihood High	

Description

Attackers can bypass periods limits and exchange all frAssets immediately.

Each period has associated a global "`limit`" variable, that tracks the rewards that can be claimed and a "`amount_claimed`", that tracks the rewards already claimed. Calling `on_exchange` or `on_immediate_exchange` verifies that "`amount_claimed`" never surpasses "`limit`", but never updates "`amount_claimed`".

An attacker with enough `frAssets` can bypass "`limit`" and claim the whole rewards pool by exchanging a "`limit`" amount of `frAssests` multiple times, since "`amount_claimed`" is always zero.

Recommendation

Update `amount_claimed` in the exchanging functions.

FF-3	Abandoned stakes are not recoverable	
Total Risk Medium	Impact Medium	Location `assets/staking_clear_program.py`
Fixed ✔	Likelihood High	

Description

Stakes abandoned by users `ClearState` transactions are not recovered by the `on_recover` function.

The `staking_clear_program` fails to correctly set the correct abandoned amount to `total_staked_abandoned` due to getting the value from an incorrect key.

Recommendation

Change "`amount_staked`" to "`staked`". We strongly suggest defining constants to avoid future errors

FF-4	Missing validation on_creation

Total Risk	Impact	Location
Info	-	`assets/algo_governance_approval_program.py`
Fixed	**Likelihood**	
✔	-	

Description

Commit and period end timestamps should be validated to be greater than `Global.latest_timestamp()`.

Recommendation

Add the missing validation.

FF-5	Misleading on_governance documentation

Total Risk	Impact	Location
Info	-	`assets/algo_governance_approval_program.py`
Fixed	Likelihood	
✔	-	

Description

The documentation of the `on_governance` function does not represent the actual function behavior.

In the first transaction, the documentation describes sender as “user” where the code checks for an “admin”, and in the second transaction the recipient is described as “user” again but in actuality the function sends it to `Gtxn[0].accounts[1].`

Recommendation

Update source code documentation.

5. Disclaimer

The information presented in this document is provided "as is" and without warranty. The present security audit does not cover any off-chain systems or frontends that communicate with the contracts, nor the general operational security of the organization that developed the code.