SOLIDIFIED

Audit Report for BrainTrust - June 07, 2021

Summary

Audit Report prepared by Solidified covering the BrainTrust smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief on 7 June 2021.

Audited Files

The source code has been supplied in the form of a GitHub repository:

https://github.com/Snowfork/BTRUST-Contracts

Commit number: `f57e34b48c2b662698ef9120d2815f07109c563f`

The scope of the audit was limited to the following files:

```
contracts
├── BTRUST.sol
├── GovernanceDecisions.sol
├── GovernorAlpha.sol
├── SafeMath.sol
└── Timelock.sol
```

Intended Behavior

The smart contracts implement an ERC-20 token and a governance voting system.

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	Medium	-
Test Coverage	Medium	-

Coverage Report:

```
--------------------------|----------|----------|----------|----------|----------------|
File                      |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
--------------------------|----------|----------|----------|----------|----------------|
 contracts/               |    63.21 |    48.75 |    67.65 |    63.47 |                |
  BTRUST.sol              |    78.26 |    73.81 |    78.95 |    78.26 |... 178,179,266 |
  GovernanceDecisions.sol |      100 |      100 |      100 |      100 |                |
  GovernorAlpha.sol       |    58.65 |    36.67 |    66.67 |    58.95 |... 306,310,311 |
  SafeMath.sol            |    11.11 |     5.56 |       10 |    11.11 |... 168,183,184 |
  Timelock.sol            |    60.98 |       50 |    88.89 |    60.98 |... 101,103,105 |
 contracts/tests/         |    33.33 |        0 |       50 |    33.33 |                |
  TimelockHarness.sol     |    33.33 |        0 |       50 |    33.33 |    25,29,30,34 |
--------------------------|----------|----------|----------|----------|----------------|
All files                 |    62.59 |    48.15 |    66.22 |    62.82 |                |
--------------------------|----------|----------|----------|----------|----------------|
```

Issues Found

Solidified found that the BrainTrust contracts contain 1 warning, no critical issues, 1 major issue, 3 minor issues, in addition to 3 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	Centralized Admin Privileges	Warning	Pending
2	GovernorAlpha.sol: Lack of validation msg.value	Major	Pending
3	BTrust.sol: Token susceptible to approve attack	Minor	Pending
4	GovernanceDecisions.sol: Allows duplicate values	Minor	Pending
5	GovernorAlpha.sol: List of proposals might not be executable within a single transaction	Minor	Pending
6	GovernorAlpha.sol: Unused import	Note	-
7	Inconsistent Solidity versions	Note	-
8	Duplicate SafeMath implementations	Note	-

Warnings

1. Centralized Admin Privileges

The full supply of tokens is allocated to one specific address during the contract deployment. This inherently allows the same address to hold complete voting power at the beginning. The contract does not include any token distribution logic or on-demand minting based on usage. Since the tokens (and the voting power) are distributed manually, there is no way to validate the fair distribution from the smart contract.

Furthermore, the `guardian` account has the privilege to change the admin of `Timelock.sol` at will. This allows the address to execute any proposal without going through the whole voting process.

Recommendation
Consider reducing centralized admin privileges.

Critical Issues

No critical issues have been found.

Major Issues

2. `GovernorAlpha.sol`: Lack of validation `msg.value`

The proposal execution method in `GovernorAlpha.sol` does not validate the `msg.value`. This can open up several issues, including loss of funds.

The contract allows flexible fund sources - either from calling address balance or from `GovernorAlpha` or `Timelock` contract balances. Since the execute method is a public function, anyone can create a transaction that uses the contract balance but was intended to be funded by the calling address.

In rare cases, if there is a proposal that allows draining the whole contract was approved (either during early stages with less number of voters or less technically educated voters) it can be executed any time to withdraw the whole funds.

Recommendation
Add a `require()` statement to ensure that the passed `msg.value` is enough to execute the entire proposed transaction array. Validating the target address with trusted addresses is also recommended to avoid any potential issues that can arise from malicious proposals.

Minor Issues

3. `BTrust.sol`: Token susceptible to approve attack

Changing an allowance through the `approve()` method brings the risk that someone may use both the old and the new allowance by unfortunate transaction ordering. A detailed description of this vulnerability can be found here:
https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM

Recommendation
One possible solution to mitigate this race condition is to implement `increaseAllowance()` and `decreaseAllowance()` functions.

4. `GovernanceDecisions.sol`: Allows duplicate values

The functions `addMarketplaceCategory()` and `addFoundationMember()` do not validate the input and this allows duplicate values to be present in both `marketplaceCategories` and `foundationMemberships` array.

Recommendation
Consider checking for duplicates before adding new items to the array if this is not an intended feature.

5. `GovernorAlpha.sol`: List of proposals might not be executable within a single transaction

The function `execute()` attempts to execute all the submitted proposal calls within the same transaction. While a limit of 10 calls is placed on the proposal, 10 calls are more than enough to fill a block (depending on the calls). This would result in an always failing and un-executable proposal.

Recommendation
Allow for partial proposal executions (i.e. resume execution) or run transaction simulations to ensure that no proposal will be un-executable.

Informational Notes

6. `GovernorAlpha.sol`: Unused import

The following debug import has been left in the code:

```
import "hardhat/console.sol";
```

Recommendation
Remove the unused import.

7. Inconsistent Solidity versions

The contracts use different compiler versions defined by pragmas. It is considered best practice to stick to a single compiler version throughout the codebase.

Recommendation
Choose a single compiler version.

8. Duplicate SafeMath implementations

The contracts all use their implementations of secure arithmetic operations.

Recommendation
Consider using a single safeMath library for all contracts.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of BrainTrust or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.