SOLIDIFIED

Audit Report for Token.io. September 9th, 2018.

Summary

Audit Report prepared by Solidified covering Token.io's Token Smart Money (TSM) System.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the below contracts. The debrief took place on September 10, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

- TokenIOStorage.sol
- TokenIOLib.sol
- TokenIOERC20.sol
- TokenIOAuthority.sol
- TokenIOCurrencyAuthority.sol
- TokenIOFeeContract.sol
- TokenIOFX.sol
- TokenIOMerchant.sol
- TokenIONameSpace.sol
- Ownable.sol
- Safemath.sol

Notes

The audit was performed on commit `f92525bb9b0fe1f648fc8dbe1dd3b8e83b8cc053`
The audit was based on the solidity compiler `0.4.23+commit.e67f0147`

Intended Behavior

The TSM smart contracts provide depository tokens that can be backed 1:1 with fiat currency held by a currency issuer, bank, or other authorized financial institution.

Token fiat currency tokens are ERC20 compliant and have an extensible storage contract to allow for future interoperability with third party systems, and to provide various financial services (e.g. currency exchange, escrow, etc.) through smart contract interfaces.

Issues Found

Critical

No critical issues found.

Major

1. Ownable contract does not implement a function to revoke access

The ownable contract allows for the provision of owner access to several different addresses, but does not contains function for revoking access. Revoking access is necessary both in the case of compromised keys and when upgrading to new contracts.

There is some deprecation logic in contract `TokenIOERC20.sol`, the remaining contracts do not implement a similar feature, while also not allowing a change the Storage address, nor for the pausing or destruction of the contract, meaning an update is impossible without leaving the old contract active. This issue is aggravated by the issue below.

Recommendation
Implement function to revoke owner privileges.

Minor

No minor issues found.

Notes

2. Beware of possible collisions in TokenIOStorage

User provided strings, such as the `currency` symbol or the `issuerFirm` are used as storage keys in `TokenIOStorage.sol`. This can result in collisions in the database that would cause unintended consequences.

The use of variable sized strings is also discouraged because this can result in ambiguous inputs for the keccak hashing function (http://solidity.readthedocs.io/en/v0.4.21/abi-spec.html#abi-packed-mode)

Specifically, in the hashing inputs outlined below, there are two dynamically sized elements `currency` and `issuerFirm` which means there is some ambiguity in the construction of the hash (i.e. two different sets of inputs to `keccak256` could result in the same set of input packed data):

```
bytes32 id_b = keccak256(abi.encodePacked('token.issued', currency, issuerFirm));
```

Recommendation

Currently the storage contract alternates between using addresses and strings for the composition of storage keys. We assumed this is intentional, and data supposed to persist through upgrades is related to currency/issuer names and data directly related to contracts is related to the contract's address. We recommend reviewing the instances above to ensure a) when strings are used integrity measures are taken in order to prevent currencies/issuers with the same name and b) data tied to contract's addresses is not supposed to persist between upgrades and checks are performed to ensure that continuity exists between presently held info.

3. Use of hard coded constants

Usage of hard-coded constants exists in numerous instances, particularly seen within `TOkenIOMerchant.sol`. Hardcoded constants may be impacted by structural changes to the EVM and should be avoided where possible.

4. SafeMath is outdated

An outdated version of Safemath is currently used. The current version switched the checks performed to `require()`, as they are commonly used to perform input validation. `Assert` should be only used to check for invariants, because it consumes all remaining gas from the transaction.

Recommendation
Ensure the use of the latest version of SafeMath for all implemented contracts.

5. Consider locking version pragma

The contracts currently use the latest solidity compiler version, but do not lock pragma.

Recommendation
Lock pragma to the compiler version used during development, as future version might introduce unforeseen changes.

6. Use of experimental features

The contract TokenIOLib make use of `pragma experimental ABIEncoderV2`. ABIEncoderV2 allows for returning dynamic sized types from functions, feature not currently in use in Token.io's smart contracts. Experimental features have not gone through extensive testing and use and can hide vulnerabilities not yet known.

Recommendation
Remove the experimental pragma declaration.

7. Incorrect constructor comments

The comments in the constructors of the following contracts are incorrectly copied from the `TokenIOERC20` source code:

- `TokenIOFeeContract`

- `TokenIOFX`
- `TokenIONameSpace`

In addition, the constructor comments in `TokenIOMerchant` are incorrectly copied from `TokenIOAuthority`.

8. Consider including messages in `require()` statements

Solidity allows for the inclusion of a message in `require()` statements since version 0.4.20. Consider including the messages to provide feedback on why transactions reverted.

SOLIDIFIED

Audit Report for Token.io. September 9th, 2018.

Closing Summary

The contracts provided for this audit are of very good quality. Community audited code has been reused whenever possible. A safe math library is used to prevent overflow and underflow issues, unless it is clearly not required.

No reentrancy attack vectors have been found and precautions have been taken to avoid uninitialized storage pointers that may lead to overwriting storage.

Static analysis revealed numerous instances in which message calls to external contracts were made, particularly in TokenIOLib.sol. In each instance, any external calls were confirmed to be to contracts owned by the user.

There are no transaction order issues present.

The ERC20 token implementation enforces approvals to be set to 0, before assigning another value, in order to protect the Approve / TransferFrom attack vector. Apart from this minor deviation for security reasons, the ERC20 implementation is standard compliant.

One major issue has been found, along with several informational notes.

The gas usage for the contracts is relatively high for their intended functionality. This is due to the storage contract abstraction and upgradable nature of the contracts.

Disclaimer

Audit Report for Token.io. September 9th, 2018.

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.