<<codecontext.io

STOX Token Sale Code Review

Version 1.0

1. Introduction

This document summarizes our best effort to review of the provided source code of the STOX Token & Token Sale solidity smart contracts. We were given commit `b4702686748bbd2e00c6f85c375e0f8164d448bf` in the `stx-technologies/stox-token` github repository and reviewed the following contracts:

- `Ownable.sol`
- `SaferMath.sol`
- `StoxSmartToken.sol (Bancor's SmartToken.sol specialization)`
- `StoxSmartTokenSale.sol`
- `Trustee.sol`

The following contracts have also been also checked, but only briefly as they have been reviewed by other parties:

From Consensys `https://github.com/ConsenSys/MultiSigWallet` reviewed commit `b9405cc30de4615e325b1d46c71cdef670bdeadc`

- `MultiSigWallet.sol` (updating only the solidity version to `^0.4.11`)

From Bancor `https://github.com/bancorprotocol/contracts` commit `290e63175c34d377a87bf3543000b69c1c64125b` (used via npm `bancor-contracts^0.2.0`)

- `SmartToken.sol`
- `ERC20Token.sol`
- `Owned.sol`
- `TokenHolder.sol`
- `SafeMath.sol`
- `IOwned.sol`
- `ISmartToken.sol`
- `IERC20Token.sol`
- `ITokenHolder.sol`

2. About this review

This is a source code security audit, only security considerations have been annotated in this report. We did not review:

- Contracts functional design
- Design, deployment, operation and contingency plan of the ICO
- EVM code generated
- Credibility or Profitability of the team, idea, or any investment in general.

Critically analyse the provided information: code reviews are hard, and some of the analysis provided in this code review could be erroneous or coming from a misinterpretation of the operations done in this smart contracts.

This review is an opportunity to re-read the code: Instead of checking only the points proposed, consider re-reading all the code to find additional issues.

Bug bounties and multiple security reviews are a must: Code reviews are not enough to ensure the safety of a smart contract, a public bug bounty is essential, as it gives the community at large a chance to participate in reviewing the overall security of the code. A great example is the Aragon Bug Bounty Program.

3. Review results

3.1. General considerations

In general, the code follows all the security measures and best practices expected in when writing solidity smart contracts that will hold user funds:

- Skilled developers with strong knowledge of the Ethereum security domain
- Safe coding patterns
- Easy to read, well commented source code
- Up-to date tools (truffle's latest version)
- Usage/wrapping of well known and well reviewed smart contracts from trusted companies without modification[1]
- A large set of tests, with really good test coverage (100% LoC, including checks from composed assertions)

3.2. Critical problems

No critical problems has been found

3.3. Potential problems

No potential problems has been found

3.4. Minor issues / comments

1. Tests do not cover event logging and their parameters. It's recommended to include unit tests to cover the events to aid in future troubleshooting.
2. Compilation of Bancor's smart contracts with solidity versions 0.4.12 or 0.4.13, will fail since these versions do not allow the use of the same modifier in one function (see https://github.com/ethereum/solidity/blob/develop/Changelog.md "Type Checker: Disallow invoking the same modifier multiple times."). This will be corrected in 0.4.14 (still in development; see https://github.com/ethereum/solidity/pull/2652.). The only bug in the compiler that is covered in the upgrade from 0.4.11 to 0.4.12/0.4.13 is `SkipEmptyStringLiteral` issue (http://solidity.readthedocs.io/en/develop/bugs.html). We verified that this code and the Bancor code does not have any instances of empty string literal, \"\", so no problems are expected to appear using the 0.4.11 compiler.

[1] Even in this cases some bugs could be found later (like the recent parity multisig vulnerability) so a few quick checks through commits has been done to check if any catastrophic bugs has been introduced.

3.5. Already fixed issues

The following issues found in initial reviewed version have been fixed in the lastest reviewed commit (`b4702686748bbd2e00c6f85c375e0f8164d448bf`):

- Necessary post deployment checks are harder if the construction object properties are private.
- Always check for zero-addresses in parameters.
- Some throws in the tests occur for different reasons than expected.
- Error in cap checking
- Methods cannot reuse names from standard contract properties (also could cause a non-fatal bug in solidity 0.4.13)

4. Conclusion

The reviewed smart contracts are well crafted. They contain simple logic to avoid complex scenarios and future problems and follow the expected common security practices for safe smart contract development. No critical or severe problems have been found. We recommend proceeding to a public bug bounty program, as usual in public token offerings.