Version 1.0

Orbs ORBS
Smart Contract Audit

Author:
Ryan Zarick
ryan@80trill.com

Co-Author:
Caleb Banister
caleb@80trill.com

May 25, 2018

80Trill is a blockchain consulting company specializing in smart contract security audits, design and development

Contents

1 Introduction

This report analyzes the smart contracts provided by the Orbs team. The primary smart contract OrbsToken.sol (ORBS) will be the actual token used by network participants. ORBS aims to be a compatible standard ERC20 token. The following technical due diligence and test information show the code complies with known open standards. This report is a description of the audit performed by the 80Trill team starting on May 21, 2018.

2 Contracts Tested

2.1 Smart Contract Hashes

File Name	SHA256 Hash of File
BasicToken.sol	5c930c1b0b5a1e689c95c2bc0701afcc68473918c5a7727bd14478ddef057df2
CanReclaimToken.sol	1f2d3b6e2e606d978e74923e7fac2752313f736e4494ef472960e8dde5669354
ERC20.sol	53c6af71322f1e0d7cb8b52d2f46005ef105e39a6e2151718dd7c690517bd12b
ERC20Basic.sol	a9cf1d9073a8a58ca6044a1720a93a69020ea80fab3f5169192630590707d593
HasNoContracts.sol	d32926eefb6c7d2166f75c81b63eec90210b825b601e85a4e0cc743ac2b38f86
HasNoTokens.sol	4f1f2ed6122d7d4cbeeb6ceb4cd7e74c8254d6ae2927bffd2e45b11bf0fa07d3
OrbsToken.sol	2ef6faad6c79a22a8e42627dd25507cef4d7c02c96eed5863ca16628872e4fda
Ownable.sol	babaa6f0611e340344d684696f668294d8835467da172f82bf28e0b22bc1b26f
SafeMath.sol	d4d0a9aafa36cd3c3b06a6d2959f252ed07977d51e7ff60dcdf904b05d2e5361
StandardToken.sol	66b71fd90d53eb78d951cee6d3259cde72ece31c0bede645e7bda74068f3d288

3 Testing

The 80Trill team identifies risks contained within smart contracts and provides clients with high quality solutions and security information. This is accomplished via a manual review process, an in-house set of automated test suites and application of common best practices. 80Trill audits are investigations intended to augment the technical expertise of clients by sharing knowledge, thus empowering them to write bulletproof smart contracts.

Most of the Orbs smart contracts are taken from Open Zeppelin's open sourced zepplin-solidity. The Open Zeppelin community has built Truffle test cases to achieve greater than 90 percent coverage of their entire library. The 80Trill team used the zepplin-solidity's Truffle test cases as well as created additional tests for testing both the OpenZeppelin and Orbs smart contracts. In addition to testing the individual OpenZeppelin smart contracts, the same tests were also ran against OrbsToken.sol to verify proper inheritance from the parent smart contracts.

3.1 Code Coverage

The following table shows the test code coverage for v1 of the Orbs smart contracts.

File	% Stmts	% Branch	% Funcs	% Lines
BasicToken.sol	100	100	100	100
CanReclaimToken.sol	100	100	100	100
ERC20Basic.sol	100	100	100	100
HasNoContracts.sol	100	100	100	100
HasNoTokens.sol	100	100	100	100
OrbsToken.sol	100	100	100	100
Ownable.sol	100	100	100	100
SafeMath.sol	100	75	100	100
StandardToken.sol	100	100	100	100

3.2 Tests Run

Contract: ERC20 Properties
✓has a name
✓has a symbol
✓has an amount of decimals (38ms)

Contract: CanReclaimToken
✓should allow owner to reclaim tokens (106ms)
✓should allow only owner to reclaim tokens

Contract: Claimable
✓should have an owner
✓should allow the setting of pending-owner multiple times (108ms)
✓changes pendingOwner after transfer (43ms)
✓should prevent to claimOwnership from no pendingOwner
✓should prevent non-owners from transfering

```
    after initiating a transfer
        ✓changes allow pending owner to claim ownership (39ms)
        ✓should prevent non-pending-owners from claiming pending transfer
        ✓should prevent owner from claiming pending transfer for pending-owner (41ms)
    after initiating a transfer of ownership to self
        ✓should allow owner to claim pending transfer for pending-owner (42ms)

Contract: HasNoContracts
    ✓should allow owner to reclaim contracts
    ✓should allow only owner to reclaim contracts

Contract: HasNoTokens
    ✓should not accept ERC223 tokens

Contract: OrbsToken
    construction
        invalid arguments
            ✓should not allow to initialize with a 0 distributor
        success
            ✓should return correct name after construction
            ✓should return correct symbol after construction
            ✓should return correct decimal points after construction
            ✓should return correct initial totalSupply after construction
            ✓should transfer the total supply to the distributor

Contract: Ownable
    ✓should have an owner
    ✓doesn not change owner after transfer ownership due to claimable override
    ✓should prevent non-owners from transfering (41ms)

Contract: BasicToken
    total supply
        ✓returns the total amount of tokens
```

```
  balanceOf
      when the requested account has no tokens
          ✓returns zero
      when the requested account has some tokens
          ✓returns the total amount of tokens
  transfer
      when the recipient is not the zero address
          when the sender does not have enough balance
              ✓reverts
          when the sender has enough balance
              ✓transfers the requested amount (81ms)
              ✓emits a transfer event
      when the recipient is the zero address
          ✓reverts


Contract: StandardToken
  total supply
      ✓returns the total amount of tokens
  balanceOf
      when the requested account has no tokens
          ✓returns zero
      when the requested account has some tokens
          ✓returns the total amount of tokens
  transfer
      when the recipient is not the zero address
          when the sender does not have enough balance
              ✓reverts
          when the sender has enough balance
              ✓transfers the requested amount (55ms)
              ✓emits a transfer event
      when the recipient is the zero address
          ✓reverts
  approve
      when the spender is not the zero address
          when the sender has enough balance
              ✓emits an approval event
              when there was no approved amount before
          ✓approves the requested amount
        when the spender had an approved amount
          ✓approves the requested amount and replaces the previous one (38ms)
      when the sender does not have enough balance
        ✓emits an approval event
        when there was no approved amount before
          ✓approves the requested amount
        when the spender had an approved amount
          ✓approves the requested amount and replaces the previous one
    when the spender is the zero address
      ✓approves the requested amount (40ms)
      ✓emits an approval event
  transfer from
    when the recipient is not the zero address
      when the spender has enough approved balance
        when the owner has enough balance
          ✓transfers the requested amount (57ms)
          ✓decreases the spender allowance (45ms)
          ✓emits a transfer event
        when the owner does not have enough balance
          ✓reverts
      when the spender does not have enough approved balance
        when the owner has enough balance
          ✓reverts
        when the owner does not have enough balance
          ✓reverts (43ms)
    when the recipient is the zero address
      ✓reverts
  decrease approval
    when the spender is not the zero address
      when the sender has enough balance
        ✓emits an approval event (84ms)
        when there was no approved amount before
          ✓keeps the allowance to zero (41ms)
        when the spender had an approved amount
          ✓decreases the spender allowance subtracting the
```

```
requested amount (44ms)
      when the sender does not have enough balance
        ✓emits an approval event
        when there was no approved amount before
          ✓keeps the allowance to zero (39ms)
        when the spender had an approved amount
          ✓decreases the spender allowance subtracting the requested amount (66ms)
    when the spender is the zero address
      ✓decreases the requested amount (38ms)
      ✓emits an approval event
  increase approval
    when the spender is not the zero address
      when the sender has enough balance
        ✓emits an approval event
        when there was no approved amount before
          ✓approves the requested amount (41ms)
        when the spender had an approved amount
          ✓increases the spender allowance adding the requested amount (45ms)
      when the sender does not have enough balance
        ✓emits an approval event
        when there was no approved amount before
          ✓approves the requested amount (39ms)
        when the spender had an approved amount
          ✓increases the spender allowance adding the requested amount (44ms)
    when the spender is the zero address
      ✓approves the requested amount (43ms)
      ✓emits an approval event
```

4 Findings

The followings sub-sections highlight the issues found during testing and code review. They are ordered by risk in descending order.

4.1 Not Returning `false` on Failure of ERC20 Methods

Risk	Status	Party	Code Version
Low	Unresolved	Orbs	v1

The ERC20 standard defines the `transfer` method as returning `false` on failure. The ORBS implementation of ERC20 uses the common technique of calling `revert()` on failure instead of returning `false`. Due to non-conformance to the ERC20 standard ORBS may not successfully interact with other software. This is due to the expectation of a receiving a `false` when ERC20 methods fail, which in this case returns an exception. Although the ERC20 standard specified return values of false upon failure, further improvements (mainly to save gas) have been implemented as exceptions. At this point these exceptions are the common best practice, and as such their use is assumed to be acceptable moving forward.

5 Conclusion

80Trill's findings have shown Orbs's ORBS to conform to the ERC20 token standard with no major security risks. In conclusion the Orbs smart contracts are complete, well tested, well documented and appear to have no vulnerabilities.

6 Qualification

This report reflects 80Trill's current understanding of known security patterns as they relate to the Orbs contract(s). It is not an endorsement of the reliability or effectiveness of the contract(s), merely an assessment of their logic and robustness. 80Trill has not reviewed the related Orbs project.

This report should not be viewed as investment or legal advice. It is provided by 80Trill as is, without warranty of any kind, express or implied. In no event shall 80Trill be liable with respect to any subject matter relating to this report including any decisions based upon it.

Formal security audits are not enough to guarantee secure smart contract(s). 80Trill recommends Orbs establish a bug bounty program, setting a period of time during which security researchers attempt to break the token's invariants, in turn encouraging further and active analysis of the contract(s).