TravelBlock (TRVL) Smart Contract Audit

Author: Magnus Dufwa, Blockalize.com
July 26, 2018

"Blockchain experts specializing in smart contract security audits, and smart contract development."

Introduction

TravelBlock intends to deploy a `TRVL` token contract in August 2018. This report analyzes the smart contracts provided by the team.

The primary smart contract 'TRVLToken.sol' (Token: TRVL) will be the actual token used, and is a ERC20 compatible token. The ERC20 standard has been extended with features to support the management of Reward TRVL tokens, and the extended logic is defined in the smart contract 'RewardToken.sol' (Reward TRVL).

This report is a description of the audit performed by Magnus Dufwa starting on July 24th, 2018. Magnus is a blockchain expert specializing in smart contract security audits, and development.

Executive summary

No potential vulnerabilities have been identified in the token contract, when used together with a centralized service to ensure the security of the system.

TRVL is supporting the ERC20 standard (and using the Open zeppelin standard)

The TRVL Token is partly based on a known open standard - the OpenZeppelin contracts, that are a set of contracts intended to be a safe building block for a variety of uses (reference: https://github.com/OpenZeppelin/openzeppelin-solidity). Overall the codebase is of high quality. It is clean, modular and follows best practices.

The TRVL token is a pausable token contract. The token contract owner can pause and un-pause transfers at any time in the future. The intention for this functionality is to disable this token contract when and if an upgrade of the token contract is required in the future.

The TRVL token is a Claimable Token. The feature gives the owner the ability to transfer ownership of the contract to a new address and it requires the owner of the new address to accept the transfer.

The TRVL token is supporting the HasNoTokens protection feature. This blocks incoming ERC223 tokens to prevent accidental loss of tokens. Should tokens (any ERC20Basic compatible) end up in the contract, it allows the owner to reclaim the tokens,

The ERC20 token features can be used by any token holder of the TRVL tokens.

The smart contracts are well written, using the open zeppelin standard SafeMath to protect against overflow attacks, and in all places deducting amounts first to protect against re-entry attacks.

Features for payments using TRVL tokens and Reward TRVL Tokens by whitelisted addresses

In addition to the standard features the TRVL token supports additional features for using the TRVL Tokens and Reward Tokens as a method of payment on the TravelBlock platform. The non ERC20 standard features are locked so that only whitelisted addresses are allowed to use the features. The following features can only be used by whitelisted addresses:

- Paying with TRVL tokens (smart contract function: paymentRegularTokens)
- Paying with Reward TRVL tokens (smart contract function: paymentRewardTokens)

The payment features can be used by any whitelisted address. The TravelBlock team intention is to only whitelist addresses of their

centralized platform so that the features can only be used by the addresses owned and controlled by the central system. This assumes the careful use by the TravelBlock administrator when the whitelisted addresses are assigned, used, and protected. The intention of this functionality is that the user would not have to facilitate anything on the blockchain and would use the system as a centralized service without any direct control of the underlying blockchain transactions. Instead, TravelBlock's centralized service is handling this, and the keys of the centralized service is owned by TravelBlock.

Features for determining the reward bonus

The TRVL token is holding a list of configured reward percentages to be used. An admin can edit the percentages available for selection.

The function to pay out Reward TRVL is allowing a whitelisted address to select any reward percentage configured, meaning that if more than one reward level is available, the smart contract does not prevent the user to use to the best rate available. As TravelBlock is intending to use the reward features only from the wallets owned by the centralized system (wallets on the whitelist), the actual reward level to be used is controlled centrally.

Deflationary properties of the token

The TRVL token is providing functions for payment using TRVL tokens, Reward TRVL, or a combination of both. Each payment using TRVL tokens:

- Deducts the amount in TRVL tokens from the user wallet address (centralized whitelisted address for that user)
- Generates Reward TRVL tokens and sends them to the user. Reward is based on the configured reward percent selected
- Gives the amount of TRVL tokens minus the Reward TRVL Tokens to the owner (TravelBlock central admin)

This will effectively reduce the supply of TRVL tokens and slowly shift tokens from being TRVL tokens to tokens being Reward TRVL Tokens (locked down version that is not supporting ERC20 standard features). When the Reward TRVL tokens are used as method of payment, they are sent to the owner account. The owner account will collect all used Reward TRVL tokens and cannot send them to any other wallet. As the supply of TRVL tokens is finite, and when used is effectively reduced, and cannot be minted, this creates a deflationary effect.

The admin function '*convertRegularToRewardTokens*' enables the admin to convert TRVL tokens owned by the admin address into Reward TRVL Tokens of any whitelisted address. The intention is to be able to handle special cases manually and convert Reward Tokens to a user directly. This also has the effect of manually lowering the supply of TRVL Tokens publicly tradeable and transferable.

Verification of Smart contract files

Code files reviewed

'TRVLToken.sol'
- TRVL and Reward TRVL token main code file encapsulating the properties of the token: name, symbol, decimals and total capped supply
- The contract also includes non-standard features for payment using TRVL tokens, Reward TRVL tokens, or a combination of both TRVL tokens and Reward TRVL tokens

'RewardToken.sol'
- Code file that manages the properties and configurations of the Reward TRVL token. The contract enables the administrator can configure the reward bonus percentages currently selectable by any whitelisted address.

'Whitelist.sol'
- Provides the administrator only ability to add/update the addresses that are on the whitelist. Only the addresses on the whitelist are enabled to use the payment functions of the TRVL and the Reward token. The administrator can add any addresses desired, however currently the goal of the company is to only allow addresses controlled by TravelBlock on behalf of users.

'Admin.sol'
- Manages the list of administrators of the contracts. All active admins have the same rights and can act as an admin on their own (a multi-signature feature is not part of the contract).

'zeppelin-solidity/contracts/token/ERC20/PausableToken.sol'
- Standard ERC20 token with the ability to freeze and unfreeze token transfer

'zeppelin-solidity/contracts/ownership/HasNoTokens.sol'
- Blocks ERC223 tokens and allows the smart contract to transfer ownership of ERC20 tokens that are sent to the contract address.

'zeppelin-solidity/contracts/ownership/Claimable.sol'
- Gives the owner the ability to transfer ownership of the contract to a new address and it requires the owner of the new address to accept the transfer.

File hashes of code files

The hashes were generated based using MD5 hashing of the file contents.

- TRVLToken.sol: cf92e3ebaaf74c50f451dc5bca2f6f63
- RewardToken.sol: a37cef15b811c6fc9bd4e6ab874f054b
- Whitelist.sol: 0d1ef1a11d46d0ef27e1f95b786777e9
- Admin.sol: 52e73e5b962d627a16063b7d4103ffe9
- StandardToken.sol: 1474ffcb71730bc98f2a60a247d81369
- PausableToken.sol: 4a40de9b83482464a27fc8644688a3b2
- Pausable.sol: 58a9290812ac9141ebe237cda9782ea7
- Claimable.sol: 67f128c2228e879c7d136a2a2a1b5d18

- CanReclaimToken.sol: 0d1c2a3105d7f8562073178a9c173397
- HasNoTokens.sol: 7e84147fe62d5406cb1a534d721767ac
- Ownable.sol: 0dcca46270f0dad0f7a753c8758bbe6a
- BasicToken.sol: c88255702228b1f882fea4898ef45850
- ERC20Basic.sol: 7cdcedb6621d5e344cc434262d004660
- ERC20.sol: 0cdf76866b6c90b8715242fb3372e586
- SafeERC20.sol: 84341259db71f810b40a37f603c2f4d3
- SafeMath.sol: 74b680004a2b882e6fc95b0386355ad2

Scope of audit

This audit is reviewing that the smart contract is executed as expected according to the given requirements, and that the coded algorithms work as expected. It does not guarantee that the code is bugfree but intends to highlight any areas of weaknesses.

This audit makes no statements or warranties about the viability of the TravelBlock's business proposition, the individuals involved in this business or the regulatory regime for the business model.

Potential Vulnerabilities

No potential vulnerabilities have been identified in the token contract, for the intended use cases.

Recommendations

VERY LOW IMPORTANCE

- Consider using HasNoEther contract from the open zeppelin code base (https://github.com/OpenZeppelin/openzeppelin-solidity/blob/master/contracts/ownership/HasNoEther.sol). The HasNoEther code reduces the risk of ETH being sent to the token contract.
    - **Required Action**: Action not required

Code review

The code has been reviewed manually line by line to ensure the code is working as intended.

There were no anomality's found in the review.

Automated testcases

The code has been tested using automated test scripts using truffle.

All tests passed.

Following scripts were executed:

- BasicToken.test.travelblock.js
- CanReclaimToken.test.travelblock.js
- Claimable.test.travelblock.js
- HasNoTokens.test.travelblock.js
- PausableToken.test.travelblock.js
- SafeERC20.test.travelblock.js
- SafeMath.test.js
- StandardToken.test.travelblock.js
- TestConstants.travelblock.js
- tokenTests.travelblock.js

Test execution result printout:

```
$ truffle test
Using network 'development'.


  Contract: StandardToken (BasicToken.test.travelblock.js)
    balanceOf
      when the requested account has no tokens
        √ returns zero
      when the requested account has some tokens
        √ returns the total amount of tokens (48ms)
    transfer
      when the recipient is not the zero address
        when the sender does not have enough balance
          √ reverts (59ms)
        when the sender has enough balance
          √ transfers the requested amount (89ms)
          √ emits a transfer event (61ms)
      when the recipient is the zero address
        √ reverts (38ms)


  6 passing (988ms)
```

```
Contract: CanReclaimToken - TravelBlock
    √ should allow owner to reclaim tokens (126ms)
    √ should allow only owner to reclaim tokens


  2 passing (692ms)


  Contract: Claimable - TravelBlock
    √ should have an owner
    √ changes pendingOwner after transfer (59ms)
    √ should prevent to claimOwnership from no pendingOwner
    √ should prevent non-owners from transfering (46ms)
    √ loses owner after renouncement (53ms)
    after initiating a transfer
      √ changes allow pending owner to claim ownership (75ms)


  6 passing (959ms)


  Contract: HasNoTokens - TravelBlock
    √ should not accept ERC223 tokens (48ms)
    √ should allow owner to reclaim tokens (123ms)
    √ should allow only owner to reclaim tokens


  3 passing (900ms)




  Contract: PausableToken - TravelBlock
    pause
      when the sender is the token owner
        when the token is unpaused
          √ pauses the token (53ms)
          √ emits a Pause event (42ms)
        when the token is paused
          √ reverts
      when the sender is not the token owner
        √ reverts
    unpause
      when the sender is the token owner
        when the token is paused
          √ unpauses the token (50ms)
          √ emits an Unpause event (38ms)
        when the token is unpaused
          √ reverts
      when the sender is not the token owner
        √ reverts
    pausable token
      paused
        √ is not paused by default
        √ is paused after being paused (56ms)
        √ is not paused after being paused and then unpaused (94ms)
      transfer
        √ allows to transfer when unpaused (77ms)
        √ allows to transfer when paused and then unpaused (158ms)
        √ reverts when trying to transfer when paused (59ms)
      approve
        √ allows to approve when unpaused (50ms)
        √ allows to transfer when paused and then unpaused (134ms)
        √ reverts when trying to transfer when paused (56ms)
      transfer from
        √ allows to transfer from when unpaused (79ms)
        √ allows to transfer when paused and then unpaused (150ms)
        √ reverts when trying to transfer from when paused (65ms)
      decrease approval
        √ allows to decrease approval when unpaused (59ms)
        √ allows to decrease approval when paused and then unpaused (132ms)
        √ reverts when trying to transfer when paused (58ms)
      increase approval
        √ allows to increase approval when unpaused (51ms)
        √ allows to increase approval when paused and then unpaused (132ms)
```

```
      ✓ reverts when trying to increase approval when paused (79ms)


  26 passing (5s)


  Contract: SafeERC20
    ✓ should throw on failed transfer (46ms)
    ✓ should throw on failed transferFrom (51ms)
    ✓ should throw on failed approve (40ms)
    ✓ should not throw on succeeding transfer (45ms)
    ✓ should not throw on succeeding transferFrom
    ✓ should not throw on succeeding approve (39ms)


  6 passing (900ms)


  Contract: SafeMath
    add
      ✓ adds correctly
      ✓ throws an error on addition overflow
    sub
      ✓ subtracts correctly
      ✓ throws an error if subtraction result would be negative
    mul
      ✓ multiplies correctly
      ✓ handles a zero product correctly
      ✓ throws an error on multiplication overflow
    div
      ✓ divides correctly
      ✓ throws an error on zero division


  9 passing (398ms)




  Contract: StandardToken
    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 (98ms)
            ✓ decreases the spender allowance (63ms)
            ✓ emits a transfer event (47ms)
          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
      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
          when there was no approved amount before
            ✓ keeps the allowance to zero (57ms)
          when the spender had an approved amount
            ✓ decreases the spender allowance subtracting the requested amount
(60ms)
        when the sender does not have enough balance
          ✓ emits an approval event (45ms)
          when there was no approved amount before
            ✓ keeps the allowance to zero (71ms)
          when the spender had an approved amount
            ✓ decreases the spender allowance subtracting the requested amount
(58ms)
      when the spender is the zero address
        ✓ decreases the requested amount (53ms)
```

```
      ✓ emits an approval event
    increase approval
      when the spender is not the zero address
        when the sender has enough balance
          ✓ emits an approval event (70ms)
          when there was no approved amount before
            ✓ approves the requested amount (61ms)
          when the spender had an approved amount
            ✓ increases the spender allowance adding the requested amount (53ms)
        when the sender does not have enough balance
          ✓ emits an approval event (42ms)
          when there was no approved amount before
            ✓ approves the requested amount (52ms)
          when the spender had an approved amount
            ✓ increases the spender allowance adding the requested amount (54ms)
      when the spender is the zero address
        ✓ approves the requested amount (54ms)
        ✓ emits an approval event (41ms)


  23 passing (5s)



  Contract: Create Travel Token Contract - TravelBlock
    Contract contains the correct amount of currently minted tokens
      ✓ Whitelist values are empty
      ✓ Reverts when add addresses to whitelist is called from a BadGuy (60ms)
      ✓ Reverts when whitelisted values are removed by a BadGuy (40ms)
      ✓ Allows Owner to call an adminOnly function, even if they are not explicitly
on the admins list (97ms)
      ✓ Reverts when address(0) is added to whitelist (42ms)
      ✓ Reverts when address(0) is added to admins (48ms)
      ✓ Reverts when empty list is added to admins
      ✓ Reverts when empty list is removed to admins
      ✓ Adding admins who are already on admins, will not emit an event (39ms)
      ✓ Adding whitelisters who are already on whitelist, will not emit an event
      ✓ Removing whitelisters who are not on whitelist, will not emit an event
      ✓ Removing admins who are not admins already, will not emit an event
      ✓ Values are removed by owner from whitelist (68ms)
      ✓ Values are removed by owner from admins (65ms)
      ✓ Tries to update a reward index that doesntexist (86ms)
      ✓ Reverts when a BadGuy tries to update percentage indexes (94ms)
      ✓ Updates an index value that exists when Owner calls it (99ms)
      ✓ Reverts when we try to add a 100% + 1 reward value
      ✓ Add a 100% reward value (75ms)


  19 passing (11s)



  Contract: Create Travel Token Contract with default values -
    Contract is initialized
      user is removed from whitelist and cant process payments
        ✓ Reverts when an empty list is passed to removeAddressFromWhitelist
function
        ✓ Reverts when an empty list is passed to addAddressToWhitelist function
        ✓ User is removed from the whitelist (44ms)
        ✓ Reverts when the users who was removed tries to process a payment (82ms)
      User calls a paymentRegularTokens method
        ✓ Verifies token balances before we make alterations (110ms)
        ✓ Reverts with zero amount
        ✓ Process payments (167ms)
        ✓ Reverts when the user tries to process payment that is more than they
have
        ✓ Reverts when non whitelisted users tries to process a payment (39ms)
        ✓ Check balances are updated accordingly (282ms)
      User calls a paymentRewardTokens method
        ✓ Reverts when non whitelisted user tries to process a payment
        ✓ Make rewards payment (104ms)
        ✓ Reverts when user is out of reward tokens (80ms)
        ✓ check values are updated properly (272ms)
      User calls a paymentMixed method
        ✓ Reverts when non whitelisted user tries to process a payment
        ✓ Make rewards payment - check values are updated properly (324ms)
```

```
    User calls a paymentRegularTokensPriority method
      √ Check values updated (118ms)
      √ Reverts when non whitelisted user tries to process a payment
      √ Reverts because user does not have enough tokens
      √ Process Payments - Check Values (522ms)


  20 passing (15s)


    User calls a paymentRewardTokensPriority method
      √ Check values updated (137ms)
      √ Reverts when non whitelisted user tries to process a payment (40ms)
      √ Reverts because user does not have enough tokens (42ms)
      √ Make rewards payment - check values stored (463ms)


  4 passing (4s)

    Owner converts tokens to user
      √ Check values updated (263ms)
      √ Reverts when trying to convert tokens for a non-whitelisted user (224ms)
      √ Reverts when owner doesnt have enough tokens to convert (217ms)


  3 passing (2s)
```

References

ERC-20 Token Standard:
https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md