Smart Contract Source Code Audit

Prepared for SWARM • June 2019

v190613

1. Table Of Contents

2. Executive Summary

In June 2019, SWARM engaged Coinspect to perform a source code review of the SWARM fundraising smart contracts. The objective of the audit was to evaluate the security of the smart contracts.

During the assessment, Coinspect found no security issues. Moreover, the code was found to be well documented and very readable, following good coding practices. All contracts compile without warnings (except two inconsequential warnings in Whitelisted.sol because of two unused function parameters). The repository also includes 42 tests that exercise all contracts and verify their expected behaviour and functionality, and all tests pass.

It is worth mentioning that the contracts are not fully autonomous and rely on transactions by accounts controlled by SWARM.

3. Introduction

The SWARM fundraising contracts include the SRC20 token and additional contracts that implement a factory that SWARM uses to create new SRC20 tokens, a registry/manager of SRC20 tokens that handles SRC20 burn/mint in relation to SWM token staking, and auxiliary contracts for features of the SRC20 tokens such as transfer restrictions, freezability and authority and delegate roles.

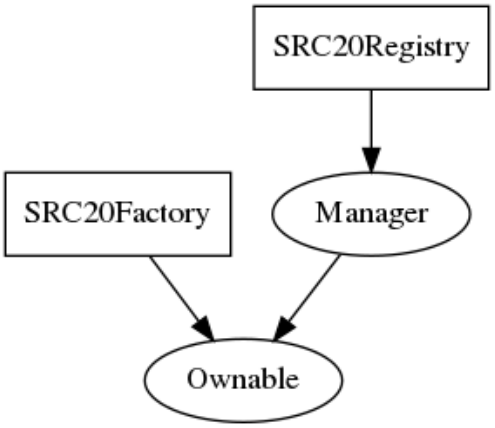
Coinspect was provided with a snapshot of the repository (without Git history) dated June 5. The scope of the audit was limited to the following Solidity source files (shown here with their sha256 hash):

```
c8a9a5561e4b607f57cb1add43f1f0e5b32a096ce22a5389a16222b12eafbc50  factories/Manager.sol
86a11c1a5e41cd2f6687c2a5f03782a38c7cee61e40a6927b35bbcfa59ccf651  factories/SRC20Factory.sol
c2433f7357881c3ec522fb227b697b257e64ed327899c4ea10c8ba19e7620cec  factories/SRC20Registry.sol
1c4e30fd3aa765cb0ee259a29dead71c1c99888dcc7157c25df3405802cf5b09  Migrations.sol
33ed53eb1cecdd9b89a8c4f9fc540e660d5d6663020cc01a1ddc887bfc6ff323  rules/ITransferRestriction.sol
f39fd46c2cd61c9a4838c275f74e335414650e1e98b791fffe2685bcb2b30d09  rules/Whitelisted.sol
b52a8477ee3e157660394d6445574abbc6d4a1892fe4ee318e6173655485b0b5  token/AuthorityRole.sol
a198b509adf6d33068c01644949182de639ea8736bd072e5747fc272b08b3163  token/DelegateRole.sol
9bdb5fd40e141cbb0194357dc292afe6bdd3fcba355dbba0bc7a78a30a0d4c1c  token/Featured.sol
0a5cbcd97179b3c26acb5e4172d61b5264ce44cf21f47bf15124ac09432d3a8b  token/Freezable.sol
fe1e188dc252c8716d78720425f769f1387fe210e128b9fb68b881f3a894d21d  token/ISRC20Managed.sol
ab817357295470c84d38b3f6c038a9d658c8f8077d3fe067066b0577f992dcba  token/ISRC20Owned.sol
718adc1ec80c8b06c3205530cd987ebbf8c54b895c0b46600df1892365d6bbbb  token/ISRC20.sol
2fd5d806c4bf1231c43b6f74714b94204e64b81cc5ff44512139c178df0ba230  token/Managed.sol
0b3768d08e09346380e4a23901f71221a4bffa5cbf977dab02196b5167265d40  token/SRC20Detailed.sol
1e217e2e1f0cdfb846284bda4f6435b1265a429ddbc114741eadcca916505892  token/SRC20.sol
```

4. Contracts

Here we provide a brief analysis of the contracts.

4.1. Factory, Registry, Manager

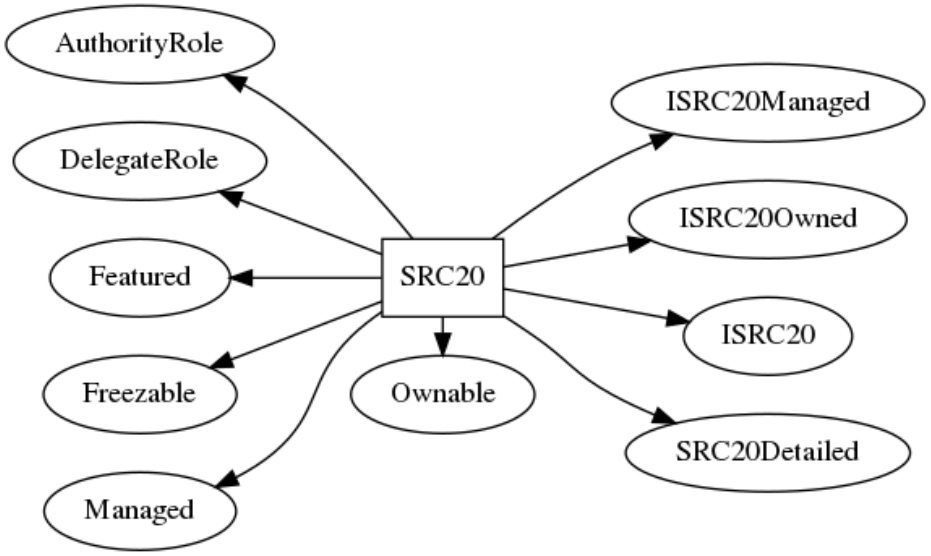


The contract `SRC20Factory` is responsible for creating new SRC20 tokens with the specified token properties and features. It has a reference to a `SRC20Registry`. All newly created tokens are registered in the associated `SRC20Registry`, and their management is also transferred to the registry. Only the owner of `SRC20Factory` can issue new tokens.

The contract `SRC20Registry` is a registry and manager of `SRC20` contracts. Registered factories (`SRC20Factory` contracts) automatically add new SRC20 tokens to the registry, and only the registered factories are allowed to add new tokens. The owner of the `SRC20Registry` contract can also remove a token.

Also, the contract `SRC20Registry` acts as a manager of the SRC20 tokens, handling SRC20 burning and minting in relation to SWM token staking. Only the owner can add or remove factories from the registry.

4.2. The SRC20 tokens



The an `SRC20` contract can be created with the following optional features:

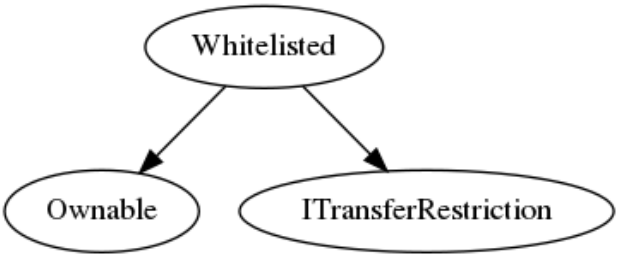
- `ForceTransfer`: allows the token issuer (the owner) to arbitrarily transfer tokens from one account to another;
- `Freezing`: allows the issuer to freeze/unfreeze particular accounts or the whole token; when an account is frozen transfers to/from that account are not allowed; freezing the whole token is equivalent to freezing all accounts;
- `AccountBurning`: allows the issuer to burn a given amount of tokens from a given account.

Minting and burning can only be performed via the `SRC20Registry`, in accordance with SWM token staking.

The `SRC20` contract has a reference to KYA (*Know Your Asset*) information: a URL where the KYA document is stored, and a hash of the document.

The `SRC20` contract can have a list of delegates, that can be added or removed only by the token issuer (the owner). Delegates are the only addresses, besides the owner, allowed to update the token KYA information.

Transfers cannot be performed directly by account addresses as is the case for an ERC20 contract. Instead, a transfer must include the authorization signature obtained via the MAP API, signed by an authority accepted by the token issuer. The token issuer can add or remove authorities to the token.



The `SRC20` contract can optionally have a reference to a contract implementing the `ITransferRestriction` interface. If set, this contract is consulted before allowing a transfer. This allows token customization with transfer restrictions.

Allowances can be *increased* or *decreased*, but not directly set to an absolute value as in a standard ERC20 contract. This makes the token invulnerable to the front running problems of ERC20 contracts.

6. Disclaimer

The present security audit is limited to smart contract code. It does not cover the technologies and designs related to these smart contracts, nor the frameworks and wallets that communicate with the contracts, nor the general operational security of the company whose contracts have been audited. This document should not be read as investment advice or an offering of tokens.