Aragon ANTv2
Smart Contract Audit

coinspect

Smart Contract Audit ANTv2

Prepared for Aragon • October 2020

V201016

1. Executive Summary

2. Introduction

3. Assessment

4. Post-Deployment Verification

5. Disclaimer

1. Executive Summary

In October 2020, Aragon engaged Coinspect to perform a source code review of `ANTv2`, the second version of the Aragon Network Token contract. The objective of the audit was to evaluate the security of the smart contract and verify that the deployed contract matches the reviewed source code.

The assessment covered the packages/v2 directory of the repository at https://github.com/aragon/aragon-network-token as of commit `cbeacf115734b4203dcc4fb6de3bca41620e7162` (tag `audit-v2`) of **October 15th**.

No issues were identified during the assessment.

The `ANTv2` and `ANTv2Migrator` instances deployed in `mainnet` were verified to match the reviewed source code and to be correctly setup and ready for the migration.

2. Introduction

`ANTv2` is a new lightweight ERC-20 token supporting ERC-2612, ERC-3009, minting and burning. It is modeled after Uniswap's `UNI-LP` with minimal changes.

The audit started on October 12th and was conducted on the packages/v2 directory of the repository at https://github.com/aragon/aragon-network-token as of commit `cbeacf115734b4203dcc4fb6de3bca41620e7162` (tag `audit-v2`) of **October 15th**.

The scope of the audit was limited to the following Solidity source files, shown here with their sha256sum hash:

```
c102cd659ef322495c8207313f678f8ed0f044380c4c7c9ca6c010531d8323d5  ANTv2Migrator.sol
b03f3675ae0e1f930499ec2acb47097278515d01c9cc0f5c2d4a092d27560ee4  ANTv2.sol
268c012a5d8eb3786a394bcefbd617837e0286e601fae9d98d5934dca57c9be6  EscrowANTv2Migrator.sol
04249f3a58a3d9f8f6996e9174024c11bbecea866e7e75c41c134846ae20e992  ApproveAndCallReceiver.sol
75885bc9250f9d82aa7fad457488090da798ad5f628b252f2cf786954ecb7132  SafeMath.sol
```

3. Assessment

The ANTv2 contract is a new lightweight token intended to replace ANT. In addition to ERC-20, the ANTv2 token supports ERC-2612, ERC-3009, minting (with a minting role) and burning. It is modeled after Uniswap's UNI-LP with minimal changes. The repository includes also two contracts for migration of ANTv1 to ANTv2: ANTv2Migrator and EscrowANTv2Migrator.

The contracts are compiled with Solidity 0.5.17. This is the latest maintenance release of the 0.5.x series.

The repository contains 60 unit tests for the smart contracts, and all pass without problems. Besides unit tests, the repository also includes 9 E2E tests that are run against a fork of mainnet in pre-deployment and post-deployment scenarios.

The ANTv2 contract implements both ERC-3009 and ERC-2612 to enable interaction with the contract via signed messages instead of direct Ethereum transactions. In order to support ERC-3009 the contract implements the transferWithAuthorization function, and for ERC-2612 it implements the permit function.

The transferWithAuthorization function can be called by anyone with a signed message authorizing a *transfer* (for a given amount, from a given account to another account, within a specified validity period). The signed message also contains a random nonce to prevent replay attacks.

The function permit can be called by anyone with a signed message authorizing an *allowance* (for a given amount, from a given account to another account, and before a specified deadline). The message contains a sequential nonce that prevents replay attacks.

There are advantages of ERC-3009 over ERC-2612:

- ERC-2612 uses sequential nonces, and this has the problem that the function permit can revert if it is not called in order, while ERC-3009 uses random 32-bit nonces and order is not a problem.
- ERC-2612 relies on the ERC-20 allowance mechanism (approve/transferFrom), and it is susceptible to front running and multiple-withdrawal attacks, while ERC-3009 does not have this problem.

It is worth mentioning that ANTv2 has two other improvements over the Uniswap UNI-LP contract it's based on:

- ANTv2 solves a possible replay attack on hardforks (see issue #23) by retrieving the chain id each time it is needed instead of precomputing the ERC-712 domain in the constructor,
- And ANTv2 disallows transfers to the 0 address in order to disambiguate transfers from burning (see issue #25).

The `ANTv2Migrator` contract is used for migration of `ANTv1` tokens to `ANTv2`. The migration is enabled when the owner (the Community Multisig) calls the function `initiate`, that mints a supply of `ANTv2` equal to the total supply of `ANTv1` and transfers the minting role to the owner (the Community Multisig). After this point, any holder of `ANTv1` can migrate to `ANTv2` by calling `approveAndCall` in the `ANTv1` instance with the migrator address and the desired amount as arguments (with `amount = -1` for migrating all). The migration consists of burning the `ANTv1` tokens (by transferring to `0x000000000000000000000000000000000000dEaD`) and transferring to the user the equal amount of `ANTv2` tokens. This is a 1:1 conversion, so a user holding `ANTv1` would hold the same amount of `ANTv2` post-migration.

As an alternative, the user can choose to use the `EscrowANTv2Migrator` contract for migrating the user's `ANTv1` balance to `ANTv2`. This involves first instantiating an `EscrowANTv2Migrator` for a given *recipient* and *initiator* addresses and transferring the user's `ANTv1` balance to the escrow. Then, at any time, the *initiator* (and only the *initiator*) can finalize the migration by calling the function `migrate` in `EscrowANTv2Migrator`, which in turn uses the deployed `ANTv2Migrator` to perform the migration, the user's `ANTv1` tokens are burned and equal amount of `ANTv2` tokens are transferred to the *recipient*. The mainnet addresses of the deployed `ANTv1`, `ANTv2` and `ANTv2Migrator` are hardcoded in `EscrowANTv2Migrator` source code.

4. Post-Deployment Verification

The contracts' source code is verified in Etherscan. Coinspect independently verified that the deployed bytecode matches the source code reviewed. To do this, Coinspect compiled the contract using exactly the same Solidity version and parameters that were used in the deployed version, and compared the bytecode resulting from the compilation with the deployed contracts.

The deployed contracts are compiled with Solidity version `0.5.17+commit.d19bba13`, optimizer enabled, 999999 runs.

The `ANTv2` contract was deployed at address **`0xa117000000f279D81A1D3cc75430fAA017FA5A2e`** by transaction `0xe14b82231bcc35f3182c73d02e494b79f15df9570307ea1815bb4a2140c44d91` in block #11060110 (Oct-15-2020 11:38:40 AM +UTC), and `ANTv2Migrator` was deployed at address **`0x078BEbC744B819657e1927bF41aB8C74cBBF912D`** by transaction `0xd76222e0ea0b7ee61422829fc86774f93c03ef28f8e25bd59757c1789de8f964` in block #11060132 (Oct-15-2020 11:41:50 AM +UTC).

The `ANTv2` constructor `(address initialMinter)` was called with the following parameter: `0x078BEbC744B819657e1927bF41aB8C74cBBF912D (ANTv2Migrator)`. The minter role was set to the `ANTv2Migrator` address, and the corresponding `ChangeMinter` event was emitted.

The `ANTv2Migrator` constructor `(address _owner, IERC20 _antv1, ANTv2 _antv2)` was called with the following parameters:
`0xbeefbeef03c7e5a1c29e0aa675f8e16aee0a5fad` (the Community Multisig),
`0x960b236a07cf122663c4303350609a66a7b288c0` (`ANTv1`),
`0xa117000000f279D81A1D3cc75430fAA017FA5A2e` (`ANTv2`).

The following properties have been verified on the deployed instances of `ANTv2` and `ANTv2Migrator`:

- The migration can only be initiated by the Community Multisig wallet, and until it is initiated, no one is able to migrate.
- Following the migration's initiation, `ANTv2` minting role will be held by the Community Multisig.
- The migration is a 1:1 `ANTv1` to `ANTv2` conversion, such that a user holding `ANTv1` is able to receive the exact same balance of `ANTv2` post-migration.
- Once the migration has initiated, a transaction to `0x960b236A07cf122663c4303350609A66A7B288C0` (`ANTv1`) with the data `0xcae9ca51000000000000000000000000078bebc744b819657e1927bf41ab8c74cbbf912dffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff000000000000000000000000000000000000000000000000000000000000006000000000000000000000000000000000000000000000000000000000000000000` will

migrate all of the sender's `ANTv1` balance into `ANTv2` (this is an `approveAndCall(0x078BEbC744B819657e1927bF41aB8C74cBBF912D,-1,0x)`).

5. Disclaimer

The information presented in this document is provided as is and without warranty. Vulnerability assessments are a “point in time” analysis and as such it is possible that something in the environment could have changed since the tests reflected in this report were run. This report should not be considered a perfect representation of the risks threatening the analysed system, networks and applications.