Jorge Rodriguez

CYBERSEC CONTRACT AUDIT REPORT
Uniris - CTDSEC.com

Introduction

During February of 2021, Uniris engaged CTDSec to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. Uniris provided CTDSec with access to their code repository and whitepaper.

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bugfree status. The audit documentation is for discussion purposes only.

I always recommend having a bug bounty program opened to detect future bugs.

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of the Uniris contract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:

- UnirisToken.sol - MD5: 6D1AB78C84EDA7A9F0280B182A758757

Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

Correctness of the protocol implementation [Result OK]

User funds are secure on the blockchain and cannot be transferred without user permission [Result OK]

Vulnerabilities within each component as well as secure interaction between the network components [Result OK]

Correctly passing requests to the network core [Result OK]

Data privacy, data leaking, and information integrity [Result OK]

Susceptible to reentrancy attack [Result OK]

Key management implementation: secure private key storage and proper management of encryption and signing keys [Result OK]

Handling large volumes of network traffic [Result OK]

Resistance to DDoS and similar attacks [Result OK]

Aligning incentives with the rest of the network [Result OK]

Any attack that impacts funds, such as draining or manipulating of funds [Result OK]

Mismanagement of funds via transactions [Result OK]

Inappropriate permissions and excess authority [Result OK]

Special token issuance model [Result OK]

Vulnerabilities

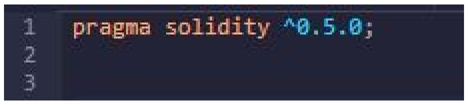
ISSUES

LOW

Compiler versions are not fixed/old.

Solidity versions in source files are not fixed; we recommend fixing them to avoid unwanted behaviors of other versions which have not been developed for it.

We also recommend upgrading the contract as it has some compiler bugs such as EmptyByteArrayCopy and DynamicArrayCleanUp.



```
pragma solidity ^0.5.0;
```

Informational

Lack of natspec comments.

Contract code is missing natspec comments, which help to make code clear and document all function’s parameters.

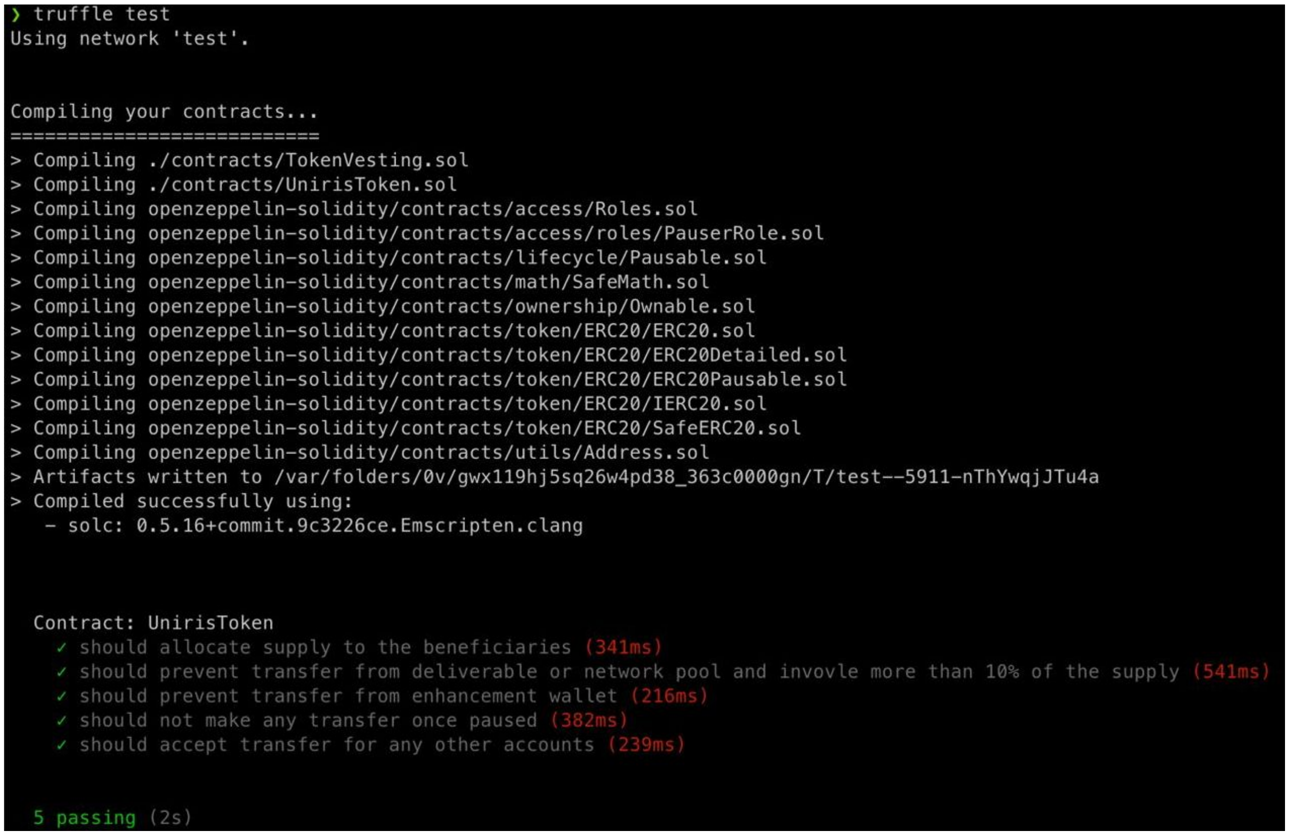
Please follow natspec solidity guide.

Contract transparency

Mint functions can't be used for irregular purposes.

Testing

All the functions described in the contract fulfill their purpose in a satisfactory way.



```
❯ truffle test
Using network 'test'.


Compiling your contracts...
===========================
> Compiling ./contracts/TokenVesting.sol
> Compiling ./contracts/UnirisToken.sol
> Compiling openzeppelin-solidity/contracts/access/Roles.sol
> Compiling openzeppelin-solidity/contracts/access/roles/PauserRole.sol
> Compiling openzeppelin-solidity/contracts/lifecycle/Pausable.sol
> Compiling openzeppelin-solidity/contracts/math/SafeMath.sol
> Compiling openzeppelin-solidity/contracts/ownership/Ownable.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC20.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC20Pausable.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/IERC20.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/SafeERC20.sol
> Compiling openzeppelin-solidity/contracts/utils/Address.sol
> Artifacts written to /var/folders/0v/gwx119hj5sq26w4pd38_363c0000gn/T/test--5911-nThYwqjJTu4a
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



  Contract: UnirisToken
    ✓ should allocate supply to the beneficiaries (341ms)
    ✓ should prevent transfer from deliverable or network pool and invovle more than 10% of the supply (541ms)
    ✓ should prevent transfer from enhancement wallet (216ms)
    ✓ should not make any transfer once paused (382ms)
    ✓ should accept transfer for any other accounts (239ms)


  5 passing (2s)
```

Architecture

Since the images are too large and cannot be attached to the document due to loss of quality, we attach hyperlinks to be able to view them correctly.

Uniris Architecture: https://pasteboard.co/JNpVZGp.png

Summary of the Audit

The contract is safe and correctly applied according to token economics.

After reviewing the contract we came to the conclusion that is safe to deploy.