quantstamp / Lendroid_Review Private

Branch: master ▾ Lendroid_Review / README.md Find file Copy path

kbak fixed scope 851ae83 21 hours ago

1 contributor

169 lines (118 sloc) 9.58 KB

Overview

This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

This security audit report follows a generic template. Future Quantstamp reports will follow a similar template and they will be fully generated by automated tools.

Specification

Our understanding of the specification was based on the following documentation:

- TGE and TRS specification document, and
- the below specification extracted through email conversations with Lendroid.
    - Accept ether as a well as a vesting decision from contributors
        - Persist amount of eth contributed + vesting decision
        - Provide support for enumeration of all contributors
        - Only allow whitelisted individuals to contribute
        - Only allows contributions within TGE timeframe
        - Only accept contributions up to `totalCap`
        - Individuals may only contribute up to `individualCap`
        - Allow duplicate contributions from individuals (up to individual cap)
    - Support bulk whitelist / blacklist operations
        - Only permissible by owner
    - Support ownership transfer
    - Immediately transfer contributed funds to `fundsWallet` after contribution
    - Support bulk withdraw function of funds
        - Should not be required since funds are immediately transfered on contribution
        - Only permissible by owner
    - Allows users to change their vesting decision
        - Only once TGE has started
        - Only for 5 days after TGE ends
        - Only for users who have contributed (and are whitelisted)
    - If user sends funds to contract without specifying a data section, just accepts fund as a contribution with no vesting
    - Support changing the individual cap at any time
        - Only owner
    - Uses `SafeMath`

  - Has all properties of the Open-Zeppelin `Ownable` contract

However, this report is strictly a review of the `SimpleTGE.sol` contract as specified in email exchanges with the Lendroid team.

Methodology

The review was conducted during 2017-Feb-12 thru 2017-Feb-18 by Richard Artoul and the Quantstamp team, which included senior engineers Kacper Bak and Steven Stewart.

Their procedure can be summarized as follows:

0. Code refactoring
1. Code review
   - Review of the specification
   - Manual review of code
   - Comparison to specification
2. Testing and automated analysis
   - Test coverage analysis
   - Symbolic execution (automated code path evaluation)
3. Best-practices review
4. Itemize recommendations

Source Code

The following source code was reviewed during the audit.

Repository	Commit
tge-contracts	2c3e8ea

Security Audit

Quantstamp's objective was to evaluate the Lendroid TGE code for security-related issues, code quality, and adherence to best-practices.

Possible issues include (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights

Test coverage

We evaluated the test coverage using truffle and solidity-coverage. The below notes outline the setup and steps that were performed.

Setup

Testing setup:

- Truffle v4.0.1
- TestRPC v6.0.7
- solidity-coverage v0.4.9
- Oyente v0.2.7
- Mythril v0.10.7

Steps

Steps taken to run the full test suite:

- Made the following edits in `contracts SimpleTGE.sol`
  - line 142: removed equals in `_publicTGEEndBlockTimeStamp > _publicTGEStartBlockTimeStamp`
  - line 145: added `require(_individualCapInWei <= _totalCapInWei);`
  - line 156: renamed `changeindividualCapInWei` to `changeIndividualCapInWei`
  - line 158: added `require(_individualCapInWei <= totalCapInWei);`
- Ran the coverage tool: `./node_modules/.bin/solidity-coverage`

Evaluation

The coverage result of the `SimpleTGE.sol` file:

```
84.13% Statements 53/63
71.67% Branches 43/60
84.21% Functions 16/19
85.51% Lines 59/69
```

We evaluated the coverage report and identified missing test coverage for `else` paths of `require()` statements. For example, `require(_totalCapInWei > 0);`, does not have any tests covering cases when the underlying expression evaluates to `false`. We recommend adding tests to cover these edge cases.

Symbolic execution (the Oyente tool) did not detect any vulnerabilities of types Parity Multisig Bug 2, Callstack Depth Attack, and Re-Entrancy Vulnerability.

Oyente reported a potential Time Dependency attack as `contributeAndVest()` and `contributeWithoutVesting()` rely on the modifier `whilePublicTGEIsActive()`, however we believe this to be a benign issue. A concern is that the transfers may occur seconds before or after the sale starts or finishes. We do not consider this an issue because the sale is meant to last for days.

Oyente reported a potential Money Concurrency attack between `fundsWallet.transfer(msg.value)` (line 177 of the function `contribute()`) and `beneficiary.transfer(this.balance)` (line 137 of the function `reclaimEther()`), however we believe this to be another benign issue. A concern is that in/out transfers may occur to/from the same wallet concurrently. The function `reclaimEther()`, however, is meant to be used by the owner, if ever.

Oyente reported that EVM code coverage is `99.2%`. The tool explored almost all the possible paths.

Mythril tool reported no issues.

Recommendations

Avoiding Copy and Paste Code

We noted that standard Zeppelin contracts, such as `SafeMath` and `Ownable` were copied and pasted into the `SimpleTGE.sol` file, instead of importing them through the `import` directives.

Code Documentation

We noted that majority of the functions were self-explanatory. Although standard documentation tags (such as `@dev`, `@param`, and `@returns`) were missing, inline comments provided sufficient information to clarify the code.

Appendix

File Signatures

Below are SHA256 file signatures of the relevant files reviewed in the audit.

```
$ shasum -a 256 ./contracts/* ./contracts/*/* ./contracts/*/*/*
98a2f9cf3f6d74d71d23374f6b118f9a4ecc12e0b2b701f3b7ba1fb7d5bc8026  ./contracts/Migrations.sol
647eef1d451afd00d8057bd460e4a4183dfe9298f0d65f13055d085a489c842f  ./contracts/SimpleTGE.sol

$ shasum -a 256 ./test/* ./test/*/*
09b80bc152afcb33cf260bbd9c556b511169dd3129a5ae94a6fa7d043d1c6a23  ./test/simple_tge.js

$ shasum -a 256 ./migrations/*
42c21b4229b39fd1cad164ed6d4c24168620e2f04a66521b2b2f2945e23b867d  ./migrations/1_initial_migration.js
```

Disclosure

Purpose of report

The scope of our review is limited to a review of Solidity code and only the source code we note as being within the scope of our review within this report. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks.

The report is not an endorsement or indictment of any particular project or team, and the report does not guarantee the security of any particular project. This report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset.

No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp Technologies Inc. (QTI). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that QTI are not responsible for the content or operation of such web sites, and that QTI shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that QTI endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. QTI assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by QTI; however, QTI does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.