Render Token Audit

OPENZEPPELIN SECURITY | SEPTEMBER 18, 2017 Security Audits

The Otoy team asked us to review and audit their Render Token (RNDR) and crowdsale contracts. We looked at the code and now publish our results.

The audited contract is in the RenderToken/rendertoken repository. The version used for this report is the commit `2967e106004e26cbad5b9a34e57e7f07bde45256`.

Good job using OpenZeppelin to write minimal extra code!

Here's our assessment and recommendations, in order of importance.

Low Severity

Misuse of FinalizableCrowdsale

(This was written for a previous version that was audited, and it was later fixed.)

As it is documented in the comments, to use `FinalizableCrowdsale` you must inherit from it and define a custom `finalization` function. Instead, `finalize` was redefined. Although this isn't causing any problems in the current state of the code, the misuse of the library hinders maintainability and may cause severe problems in future revisions. Remove this function and move the extra minting to a `finalization` function.

Notes

- Consider adding events to log when an address is added or removed from the whitelist.
- *(This was written for a previous version that was audited.) What is named "minimum cap" (`minCap`) is not really a cap, because the word means an upper bound. In OpenZeppelin we use the term "goal" to refer to this concept.*
- `RenderToken` is an instance of `MintableToken`, which has a public variable `mintingFinished` initially set to `false`. Since this is a public variable that will be shown in interfaces (such as Etherscan's) it might cause some confusion if it remains `false` after the crowdsale ends. Consider calling `token.finishMinting()` at finalization, to set the variable to `true` and avoid this potential confusion.
- The final token allocation will be: 25% of supply will have been on sale in the crowdsale, 65% is assigned to the foundation, and 10% will be held by the founders. Tokens not sold in the crowdsale will be sent to the foundation.

Conclusion

One low severity issue was found and explained, along with recommendations on how to fix it.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the Render Token contract. We have not reviewed the related Render project. The above should not be construed as investment advice. For general information about smart contract security, check out our thoughts here.

Related Posts

Beefy
Zap Audit

BRUSHFAM
OpenBrush Contracts Library Security Review

Linea
Bridge Audit

Beefy Zap Audit

BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

Security Audits

OpenBrush Contracts Library Security Review

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Security Audits

Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits