Everus Token Audit

OPENZEPPELIN SECURITY | OCTOBER 26, 2017

Security Audits

The Everus team asked us to review and audit their Everus Token (EVR) contract. We looked at the code and now publish our results.

The audited code is located in the EverusWorld/Contracts repository. The version used for this report is commit `41cd2da8382225b5f6e5eebd5eafae9dc9ae1b85`.

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

High Severity

No issues of high severity.

Medium Severity

No issues of medium severity.

Low Severity

Unnecessary low level `call`

The implementation of `approveAndCall` is using a low level `call` to invoke a function on another contract. A comment explains this is *"so one doesn't have to include a contract in here just for this"*. We would actually strongly recommend including the extra contract, and replacing the usage of `call` for a Solidity function call. Manually crafting the function signature is highly error prone, and on top of that, the arguments are not typechecked. Consider adding an abstract contract with the `receiveApproval` function declared, and using it to implement `approveAndCall`.

No Transfer event for minted tokens

It is recommended, in the ERC20 spec, to emit a `Transfer` event with the source (`_from`) set to `0x0` when minting new tokens. This enhances user experience by allowing applications such as Etherscan to learn of the new token holders. In this case this is only relevant for the constructor, where the initial balance is assigned to the contract creator. Nonetheless, consider emitting the corresponding event: `Transfer(0x0, msg.sender, _initialAmount)`.

Notes & Additional Information

- The ERC20 implementation used is the `HumanStandardToken` from ConsenSys.
- It is generally recommended to use a safe math module because otherwise operations may silently overflow and cause bugs. In this case it is not strictly necessary because in this simple contract all integer amounts will be at most the token's initial supply. However, it is something that should be kept in mind for any further lines of code added.

functionality of ERC20 tokens, described here. Consider implementing one of the proposed mitigations, or using the ERC20 implementation from OpenZeppelin which already has one in place.

Conclusion

No critical or high severity issues were found. Some small changes were proposed to follow best practices and reduce potential attack surface.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the Everus Token contract. We have not reviewed the related Everus 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

OpenBrush Contracts Library Security Review

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

Linea Bridge Audit

Security Audits

Security Audits

Security Audits

OpenZeppelin

Defender Platform

Secure Code & Audit
Secure Deploy
Threat Monitoring
Incident Response
Operation and Automation

Services

Smart Contract Security Audit
Incident Response
Zero Knowledge Proof Practice

Learn

Docs
Ethernaut CTF
Blog

Company

About us
Jobs
Blog

Contracts Library

Docs

Privacy | Terms of Use