BitClave Token Audit

OPENZEPPELIN SECURITY | NOVEMBER 2, 2017

Security Audits

The BitClave team asked us to review and audit their Consumer Activity Token (CAT) contracts. We looked at the code and now publish our results.

The audited code is located in the bitclave/crowdsale repository. The version used for this report is commit `057357fecbcc00c9a6cf96831d71d94fb7a13f03`.

Great work reusing the existing OpenZeppelin libraries! The additional contracts are very thoughtfully designed, and are a good extension of the framework.

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

Update: *The BitClave team has followed most of our recommendations and updated the contracts. The new version is at commit* `d12edef05bcc5e5d24a0e53be224f304e1aa0ea8`.

Critical Severity

No issues of critical severity.

High Severity

No issues of high severity.

Medium Severity

No issues of medium severity.

As documented, to use `FinalizableCrowdsale` you must inherit from it and define a custom `finalization` function. Instead, in `CATCrowdsale` it was `finalize` that was redefined. Although it is not causing any problems in the code as is, the misuse of the library damages reusability. Rename `finalize` to `finalization`, including the `super.finalize()` line.

Update: *This was fixed in the latest version.*

Notes & Additional Information

- Great work reusing the existing OpenZeppelin libraries!
- The code is very modular, and looks thoughtfully designed. It is somewhat hard to navigate the inheritance chain, but this is mostly a consequence of the `Crowdsale` design in OpenZeppelin.
- In `BonusCrowdsale`, there is a comment saying that bonuses are represented as values from 0 to 1000. We think this is wrong, as there could be, for example, a bonus of 200% (represented as 2000). If the comment is wrong, remove it. If it is correct, consider validating the values received in `setBonusesForTimes` and setBonusesForAmounts.

  Update: *This was fixed in the latest version.*
- The value of `decimals` is duplicated in `CATCrowdsale` and `CAToken`. Try to avoid the redundancy so that the contracts can't get out of sync.
- The `TokenMint` event is redundant, given that `MintableToken` already emits both `Mint` and `Transfer` events.

  Update: *This was fixed in the latest version.*
- The calculation of `usdValue` in `BonusCrowdsale` is somewhat obscure, and could use a comment explaining it or intermediate variables. (For example, it is not immediately clear why the division by 100 is needed.)

  Update: *This was fixed in the latest version.*
- Consider removing the special case for when there are no bonuses in `BonusCrowdsale`'s `buyTokens`. It is a small optimization that is not worth the potential for error.

  Update: *This was fixed in the latest version.*

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 Consumer Activity Token contracts. We have not reviewed the related BitClave 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

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

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