SOLIDIFIED

Audit Report for TinyTap CourseVault - April 18, 2023

Summary

Audit Report prepared by Solidified covering the TinyTap CourseVault smart contract.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on March 23, 2023, and the results are presented here.

Audited Files

The source code has been supplied in a private source code repository:

https://github.com/tinytap/smart-contracts/blob/master/contracts/CourseVault.sol

Commit number: `e292dccac24290ccf3824754984150c5ed3338f8`

Intended Behavior

TinyTap CourseVault is a contract that allows users to withdraw funds that are associated with a particular token ID from the CourseNFT collection.

Findings

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Low	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

Issues Found

Solidified found that the TinyTap CourseVault contracts contain no critical issues, no major issues, 1 minor issue, 1 informational note and 2 warnings.

We recommend issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	Users can permanently lose access to their funds if CourseVault is paused indefinitely	Minor	Acknowledged
2	Function setVerifiedAccountsBatch() can save gas by declaring accounts and states as calldata	Note	-
3	Function withdrawERC20() allows owner to drain the contract tokens	Warning	-
4	Removing a verified account from the whitelist can be front-run	Warning	-

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. Users can permanently lose access to their funds if `CourseVault` is paused indefinitely

There is a minor risk that `CourseVault` gets paused indefinitely, for instance, if both the `owner` and `founder` lose access to their keys after the contract has been paused. In such a case, users will be permanently unable to withdraw their funds.

Recommendation

Either eliminate the pausing functionality or set a maximum amount of time that the contract could be paused for.

Informational Notes

2. Function `setVerifiedAccountsBatch()` can save gas by declaring `accounts` and `states` as `calldata`

Declaring the parameters `accounts` and `states` as `calldata` instead of `memory` can potentially save on gas, since the array values would be directly read from `calldata` instead of being copied to `memory` first.

Recommendation

Declare the `accounts` and `states` parameters as `calldata` instead of `memory`.

Warnings

3. Function `withdrawERC20()` allows `owner` to drain the contract tokens

The function `withdrawERC20()` allows the contract `owner` to drain the entire `ERC20` contract funds at any time of their choice.

4. Removing a verified account from the whitelist can be front-run

The function `setVerifiedAccount()` allows either the `founder` or the `verifier` to remove a verified account from the whitelist. This however can be circumvented, where the account to be removed can always front-run this transaction with their own `withdraw()` transaction first.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of TinyTap or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Oak Security GmbH