SOLIDIFIED

Audit Report for Open Campus - June 16, 2023

DRAFT - DO NOT PUBLISH

Summary

Audit Report prepared by Solidified covering the Open Campus smart contracts.

Process and Delivery

Two (2) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on June 16, 2023, and the results are presented here.

Audited Files

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

https://github.com/NFTStudios/animoca-open-campus-contract

Commit number: `348f21f50245492e90d446d3b4bf7baa30d133b4`

Intended Behavior

Open Campus is a set of NFT minter contracts.

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	Low	-
Test Coverage	High	-

SOLIDIFIED

Audit Report for Open Campus - June 16, 2023

DRAFT - DO NOT PUBLISH

Issues Found

Solidified found that the Open Campus contracts contain no critical issues, no major issues, 3 minor issues, and 3 informational notes.

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	OpenCampusMinter.sol: A minter can be permanently locked out of the mint() function if they provide a large enough nonce value	Minor	Pending
2	OpenCampusMinter.sol: Function setOpenCampusWalletAddress() does not validate the provided wallet address	Minor	Pending
3	Lockable.sol: Once locked, minting can no longer be unlocked	Minor	Pending
4	OpenCampusMinter.sol: Functions pause(), unpause() and mint() do not emit events	Note	-
5	OpenCampusGenesis.sol: Hardcoding the IPFS URI limits contract reusability	Note	-
6	OpenCampusMinter.sol: Redundant import of the String library	Note	-

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. OpenCampusMinter.sol: A minter can be permanently locked out of the mint() function if they provide a large enough nonce value

If the minter provides a nonce value that is equal to the maximum value allowed by `uint256` in function `mint()`, their `_data.to` address will become permanently locked out of minting any new NFTs. This is because `mint()` always requires that `_data.nonce > nonces[_data.to]`, which in that case would no longer be possible.

Recommendation
Use incremental nonces, where the check `_data.nonce > nonces[_data.to]` is replaced by `_data.nonce == nonces[_data.to]+1`.

2. OpenCampusMinter.sol: Function setOpenCampusWalletAddress() does not validate the provided wallet address

---

The function `setOpenCampusWalletAddress()` does not validate the provided `_openCampusWalletAddress`, which can potentially lead to the `mint()` function unexpectedly reverting.

Recommendation
Validate that `_openCampusWalletAddress != address(0)`.

3. Lockable.sol: Once locked, minting can no longer be unlocked

---

Once minting is locked using the function `lockMint()`, there is no way to reverse this behavior by re-enabling minting once again.

Recommendation
Either provide an owner-only `unlockMint()` function, or in case the above is the intended behavior, provide documentation that supports that.

Informational Notes

4. OpenCampusMinter.sol: Functions pause(), unpause() and mint() do not emit events

Recommendation
Consider having the aforementioned functions emit the appropriate events so that protocol participants can more conveniently detect when the contracts have been paused/unpaused, or an NFT has been minted.

Note
The same issue applies to functions: `Lockable.lockMint()`, `ProtectedMintBurn.addMinter()` and `ProtectedMintBurn.removeMinter()`.

5. OpenCampusGenesis.sol: Hardcoding the IPFS URI limits contract reusability

Hardcoding the IPFS URI passed to the `ERC1155()` constructor can potentially limit future reusability of the `OpenCampusGenesis` contract.

Recommendation
Consider passing the IPFS URI as an `OpenCampusGenesis` constructor parameter.

6. OpenCampusMinter.sol: Redundant import of the String library

Recommendation

Consider removing the unused library import.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of TinyTap LTD 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