SOLIDIFIED

Audit Report for Animoca Open Campus Contracts - Oct 18, 2023

Summary

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

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on Oct 18, 2023, and the results are presented here.

Intended Behavior

Animoca Open Campus Contracts is a set of Solidity contracts for the Open Campus project.

Audited Files

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

https://github.com/animoca/opencampus-ethereum-contracts/
Commit number: `9e8e6d371f38d04298ca05b94e556276faef511c`

Scope:
`/contracts/sale/PublisherNFTMinter.sol`
`/contracts/sale/PublisherNFTSale.sol`

Update: The team provided fixes on October 23, 2023.
Commit number: `3ec9be0e99a9b32620b1b302b4e83d4fcacf44d4`

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 Animoca Core Library V2 contracts contain no critical issues, 1 major issue, 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	PublisherNFTSale.sol: Function currentMintPrice() returns zero for certain combinations of price and discount values	Major	Resolved
2	PublisherNFTSale.sol: Function withdraw() can potentially fail when transferring ETH to a smart contract	Minor	Resolved
3	PublisherNFTSale.sol: Missing constructor validation	Minor	Resolved
4	PublisherNFTSale.sol: Discount percentages are not validated	Minor	Resolved
5	PublisherNFTSale.sol: Discount percentages have low precision	Note	Resolved
6	PublisherNFTSale.sol: Function mint() hardcodes gas calculation values	Note	Resolved
7	PublisherNFTMinter.sol/PublisherNFTSale.sol: Potential value mismatch between PublisherNFTMinter.MINT_SUPPLY_LIMIT and PublisherNFTSale.MINT_SUPPLY_LIMIT	Note	Acknowledged

Critical Issues

No critical issues have been found.

Major Issues

1. `PublisherNFTSale.sol`: Function `currentMintPrice()` returns zero for certain combinations of `price` and discount values

There's a spectrum of value combinations for `price` and `DISCOUNT_PERCENTAGE_x` that will cause the mint price calculations to round down to zero. This could happen if `price * (100 - DISCOUNT_PERCENTAGE_x) < 100`. The following are a couple of examples:

- `price=1`, `discount=1`. This would evaluate to `(1 * (100 - 1)) / 100 = 99/100 = 0`.
- `price=99`, `discount=99`. This would evaluate to `(99 * (100 - 99)) / 100 = 99/100 = 0`.

Recommendation

Consider using a scaling factor to increase the calculation precision.

Status

Resolved

Minor Issues

2. PublisherNFTSale.sol: Function withdraw() can potentially fail when transferring ETH to a smart contract

Function `withdraw()` calls `transfer()` when sending ETH to the `to` address, which only forwards 2300 gas. In cases where the `to` address is a smart contract whose fallback function consumes more than 2300 gas, the call will always fail. This will have the side effect of potentially preventing smart contracts (e.g. DAOs) from receiving transfers.

For a more in-depth discussion of issues with `transfer()` and smart contracts, please refer to `https://diligence.consensys.net/blog/2019/09/stop-using-soliditys-transfer-now/`

Recommendation
Replace instances of `transfer()` with `call()`.

Status
Resolved

3. PublisherNFTSale.sol: Missing constructor validation

The following validations are missing from the `PublisherNFTSale` constructor:

- Array length validation for the `discountThresholds`, `discountPercentages` and `timestamps` arrays.
- `mintPrice`, `mintSupplyLimit` and `mintLimitPerAddress` non-zero validation.

Recommendation

Implement the aforementioned validations.

Note

A similar issue exists in the `PublisherNFTMinter` constructor.

Status

Resolved

4. `PublisherNFTSale.sol`: Discount percentages are not validated

`DISCOUNT_PERCENTAGE_1`, `DISCOUNT_PERCENTAGE_2` and `DISCOUNT_PERCENTAGE_3` can have values greater than `100`, which will result in function `currentMintPrice()` (and subsequently `mint()`) always failing.

Recommendation

Validate the aforementioned variables in the contract's constructor.

Status

Resolved

Informational Notes

5. `PublisherNFTSale.sol`: Discount percentages have low precision

The discount percentages can only be incremented in 100 BPS.

Recommendation

Consider allowing for higher discount percentage precision.

Status

Resolved

6. `PublisherNFTSale.sol`: Function `mint()` hardcodes gas calculation values

Several gas calculation values are hardcoded, which can result in potential issues in case a future EVM fork changes gas prices.

Recommendation

Consider storing the gas values in settable variables.

Status

Resolved

7. PublisherNFTMinter.sol/PublisherNFTSale.sol: Potential value mismatch between PublisherNFTMinter.MINT_SUPPLY_LIMIT and PublisherNFTSale.MINT_SUPPLY_LIMIT

In case of incorrect initialization, there could exist a case where there's a mismatch between the values of `PublisherNFTMinter.MINT_SUPPLY_LIMIT` and `PublisherNFTSale.MINT_SUPPLY_LIMIT`.

Recommendation

Consider creating a separate 'settings' contract that both contracts fetch this value from.

Status

Acknowledged. Team's response: "*the 2 contracts are deployed on different networks, so they cannot fetch the value from a common settings contract. Also, we do not want to hardcode this value as it may be subject to changes before deployment*".

Disclaimer