SOLIDIFIED

Audit Report for Animoca Core Library V2 - June 28, 2023

Summary

Audit Report prepared by Solidified covering the Animoca Core Library V2 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 June 28, 2023, and the results are presented here.

Audited Files

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

https://github.com/animoca/ethereum-contracts/tree/version-2.0.0

Commit number: `6be5516fa5b55ef58940a47c4a19231fc46bf69b`

Scope:
```
/contracts/token/ERC20/preset/ERC20FixedSupply.sol
/contracts/token/ERC20/preset/proxied/ERC20FixedSupplyProxied.sol
/contracts/token/ERC20/preset/ERC20MintBurn.sol
/contracts/token/ERC20/preset/proxied/ERC20MintBurnProxied.sol
/contracts/payment/CumulativeMerkleClaim.sol
/contracts/token/ERC20/libraries/ERC20Storage.sol#L42 (function
initWithAllocations())
```

Update: The team provided fixes on July 3, 2023.
Commit number: `c813045b79473a100e8005c7f1ce6ae340f7d235`

Intended Behavior

Animoca Core Library is a Solidity contracts development library.

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, no major issues, 1 minor issue, 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	ERC20MintBurnProxied.sol / ERC20FixedSupplyProxied.sol: Contracts cannot transfer ownership	Minor	Resolved
2	CumulativeMerkleClaim.sol: The contract remains paused after setMerkleRoot() has been called	Note	Resolved
3	Redundant imports and library usage	Note	Resolved
4	Missing documentation	Note	Resolved

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

1. ERC20MintBurnProxied.sol / ERC20FixedSupplyProxied.sol: Contracts cannot transfer ownership

The contracts `ERC20MintBurnProxied` and `ERC20FixedSupplyProxied` cannot transfer ownership, as they do not inherit from `ContractOwnershipBase`.

Recommendation

Implement ownership transfer functionality in the aforementioned contracts.

Status

Resolved

Informational Notes

2. `CumulativeMerkleClaim.sol`: The contract remains paused after `setMerkleRoot()` has been called

It should be always the case that the contract needs to be unpaused after `setMerkleRoot()` has been called. In its current implementation however, the contract will remain paused, and the owner must send another `unpause()` transaction for it to be fully functional.

Recommendation
Consider automatically unpausing the contract in the same `setMerkleRoot()` function call.

Status
Resolved

3. Redundant imports and library usage

- `ERC20MintBurn.sol`: Redundant import and usage of `ERC20Storage`. Resolved.
- `ERC20MintBurnProxied.sol`: Redundant usage of `ERC20Storage` for `ERC20Storage.Layout`. Resolved.
- `ERC20FixedSupplyProxied.sol`: Redundant import of `ContractOwnershipBase`, and redundant usage of `ERC20MetadataStorage` for `ERC20MetadataStorage.Layout` and `ERC20PermitStorage` for `ERC20PermitStorage.Layout`. Resolved.
- `ERC20MintBurnProxied.sol`: Redundant import of `ContractOwnershipBase`, and redundant usage of `ERC20MetadataStorage` for `ERC20MetadataStorage.Layout` and `ERC20PermitStorage` for `ERC20PermitStorage.Layout`. Resolved.

4. Missing documentation

- `CumulativeMerkleClaim.sol`: Should document that users need to provide implementation for `_distributePayout()`. Resolved.

Disclaimer

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