SOLIDIFIED

Audit Report for Animoca - July 5, 2021

Summary

Audit Report prepared by Solidified covering a subset of the Animoca smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief on 5 July 2021.

Audited Files

The source code has been supplied in the form of specific commits in three GitHub repositories:

https://github.com/wighawag/universal-forwarder/tree/0ac0b2ece2feaee7ce0e5401480eca4016835b9c

Scope limited to the following files:

```
src/
├── ForwarderRegistry.sol
├── Test
│   ├── TestSpecificForwarderReceiver.sol
│   └── TestUniversalForwardingReceiver.sol
├── UniversalForwarder.sol
└── solc_0.7
    └── ERC2771
        ├── IERC2771.sol
        ├── IForwarderRegistry.sol
        ├── UsingAppendedCallData.sol
        ├── UsingSpecificForwarder.sol
        └── UsingUniversalForwarding.sol
```

https://github.com/animoca/ethereum-contracts-assets/tree/c9b3d82bf9cf72a1f726887410b4ce9fe1fd32e2

Socpe limited to the following files:

```
contracts
├── bridging
│   ├── ChildERC20Base.sol
│   ├── ERC20BasePredicate.sol
│   ├── ERC20EscrowPredicate.sol
│   └── ERC20MintBurnPredicate.sol
├── mocks
│   └── token
│       ├── ERC20
│       │   ├── ChildERC20BurnableMock.sol
│       │   ├── ChildERC20Mock.sol
│       │   ├── ERC20BurnableMock.sol
│       │   ├── ERC20Mock.sol
│       │   └── ERC20ReceiverMock.sol
└── token
    └── ERC20
        ├── ChildERC20.sol
        ├── ChildERC20Burnable.sol
        ├── ERC20.sol
        ├── ERC20Burnable.sol
        ├── ERC20Receiver.sol
        ├── IERC20.sol
        ├── IERC20Allowance.sol
        ├── IERC20BatchTransfers.sol
        ├── IERC20Burnable.sol
        ├── IERC20Detailed.sol
        ├── IERC20Metadata.sol
        ├── IERC20Mintable.sol
        ├── IERC20Permit.sol
        ├── IERC20Receiver.sol
        └── IERC20SafeTransfers.sol
```

https://github.com/animoca/ethereum-contracts-core/tree/7db4e33e56f6c691b16891ce5878bdee5a84a481

Scope limited to the following files:

```
/contracts/access/MinterRole.sol
/contracts/access/Ownable.sol
/contracts/algo/EnumMap.sol
/contracts/bridging/IChildToken.sol
/contracts/bridging/ITokenPredicate.sol
/contracts/introspection/IERC165.sol
/contracts/lifecycle/Pausable.sol
/contracts/metatx/ManagedIdentity.sol
/contracts/utils/Recoverable.sol
/contracts/utils/RLPReader.sol
/contracts/utils/types/AddressIsContract.sol
/contracts/utils/types/UInt256ToDecimalString.sol
/contracts/utils/types/UInt256Extract.sol
```

Intended Behavior

The smart contracts implement an ERC-20 token that can be bridged to a L2 chain, the related bridge contracts and contracts for a meta-transaction forwarder.

Code Complexity and Test Coverage

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 equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

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

Issues Found

Solidified found that the Animoca contracts contain no critical issues, 2 major issues, 2 minor issues, in addition to 4 informational notes.

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

Issue #	Description	Severity	Status
1	Multiple contracts: The function onERC20Received() can be called by anyone	Major	Pending
2	ERC20.sol: The function _batchBurnFrom() incorrectly updates _totalSupply	Major	Pending
3	Multiple Contracts: The function recoverERC20s() might fail to recover certain ERC-20 tokens	Minor	Pending
4	Bridging contracts: centralized design, the manager role can perform any actions	Minor	Pending
5	Inconsistent Solidity versions	Note	-
6	/contracts/metatx/ManagedIdentity.sol: Outdated compiler warning suppression	Note	-
7	ERC20EscrowPredicate.sol: The contract expects that ERC-20 token contract transfer() and transferFrom() functions return true on successful transfer	Note	-
8	Ownable.sol: Zero address validation	Note	-

Critical Issues

No critical issues have been found.

Major Issues

1. Multiple contracts: The function onERC20Received() can be called by anyone

The message sender is never checked in any of the function `onERC20Received()` implementation.

Affected contracts:
`ChildERC20.sol` - `Withdrawn` event will be emitted.
`ChildERC20Burnable.sol` - anyone can burn tokens belonging to the contract.
`PolygonREVV.sol` - `escrowed` amount can be arbitrarily increased by anyone.

Furthermore, the mock contracts contain similar implementation.
`ChildERC20Mock.sol` - `_inEscrow` amount can be artificially increased by anyone.
`ERC20ReceiverMock.sol` - `ERC20Received` event will be emitted.

Recommendation
Consider checking that `msg.sender` is a valid (expected) token contract.

2. ERC20.sol: The function _batchBurnFrom() incorrectly updates _totalSupply

The function `_batchBurnFrom()` reduces `_totalSupply` supply multiple times by the amount burned so far while executing the loop.

Recommendation
Consider moving the `_totalSupply` updating code outside of the `for` loop.

Minor Issues

3. Multiple Contracts: The function recoverERC20s() might fail to recover certain ERC-20 tokens

The function `recoverERC20s()` will not transfer `ERC20` tokens which `transfer()` function does not return `true`.

Contracts affected:
`ChildERC20Mock.sol`
`Recoverable.sol`
`PolygonREVV.sol`

Recommendation
Consider using the `SafeERC20` library.

4. Bridging contracts: centralized design, the manager role can perform any actions

The bridging contracts are controlled by one address. This centralization allows the address to withdraw the escrow funds anytime by providing a custom `log` input.

Furthermore, an address controlled by one user or private key comes with the risk of getting stolen or lost.

Recommendation
We recommend explicitly inform the users with associated risks. We also suggest extra care with offline key management for this account and getting a full-stack audit for the off-chain key management code.

Informational Notes

5. Inconsistent Solidity versions

The contracts use different compiler versions defined by pragmas. It is considered best practice to stick to a single compiler version throughout the codebase.

Recommendation
Choose a single compiler version.

6. `/contracts/metatx/ManagedIdentity.sol`: Outdated compiler warning suppression

The function `_msgData()` uses the `this;` statement to suppress a compiler warning. This trick is not necessary anymore with current compiler versions.

Recommendation
Simplify function for code clarity.

7. `ERC20EscrowPredicate.sol`: The contract expects that `ERC-20` token contract `transfer()` and `transferFrom()` functions return `true` on successful transfer

The function `exitTokens()` will fail if ERC20 `transfer()` does not return `true`
The function `lockTokens()` will fail if ERC20 `transferFrom()` does not return `true`
This could result in locked tokens.

Recommendation
Consider using the `SafeERC20` library.

8. Ownable.sol: Zero address validation

The function `transferOwnership()` does not check for `address(0)` value of `newOwner` parameter.

Recommendation
Consider requiring that the `newOwner` parameter is not `address(0)` value if ownerership should not be renoucable.

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.

Solidified Technologies Inc.