Security Audit Report

OpenQ

3/21/2022

PREPARED FOR:
OpenQ, openq.devs

ARCADIA CONTACT INFO
Email: audits@arcadiamgroup.com
Telegram: https://t.me/thearcadiagroup

Table of Contents

Executive Summary

A Representative Party of OpenQ ("**CLIENT**") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the following OpenQ smart contracts on the **OpenQDev/OpenQ-Contracts** github repository at Commit **#46d09e18c5819f031d7fc75bc45d51495bed3ac5**

The scope of this audit included the following files:

1. **BountyV0**.sol
2. **Bounty**.sol
3. **BountyFactory.**sol
4. **OpenQV0**.sol
5. **IOpenQ**.sol
6. **OpenQStorable**.sol
7. **Oraclize**.sol
8. **OpenQStorage**.sol

Arcadia completed this security review using various methods primarily consisting of dynamic and static analysis. This process included a line-by-line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

There were **5** issues found, **0** of which were deemed to be 'critical', and **1** of which were rated as 'high'.

Severity Rating	Number of Original Occurrences	Number of Remaining Occurrences
CRITICAL	0	0
HIGH	1	1
MEDIUM	1	1
LOW	2	2
INFORMATIONAL	1	0

Findings

1. Use SafeMathUpgradeable

Issue: **OPENQ-1**
Severity: **INFORMATIONAL**

Target:**OpenQV0.sol**
Finding Type: **DYNAMIC**

https://github.com/OpenQDev/OpenQ-Contracts/blob/audit/contracts/OpenQ/Implementations/OpenQV0.sol#L25

Using SafeMathUpgradeable instead of SafeMath would be good for an upgradeable contract.

Action Recommended:

Import SafeMathUpgradeable library from OpenZeppelin.

2. Additional checks for expiration time

Issue: **OPENQ-2**
Severity: **LOW**

Target:**BountyV0.sol**
Finding Type: **DYNAMIC**

Functions `receiveFunds` and `receiveNft` should check `_expiration` time at least greater than 0 to avoid possible manual mistakes when putting expiration time in calling the functions.

Action Recommended:

Add `require` or any check for `_expiration` to those functions.

3. Checking payout address when claiming

Issue: **OPENQ-3**
Severity: **MEDIUM**

Target:**BountyV0.sol**
Finding Type: **DYNAMIC**

The payout address `_payoutAddress` when claiming is only one sided confirmation from the `openQ` contract when calling the `claim` function. As funding will be transferred from deposited funds by a funder, the `funder` should be able to set/confirm `payoutAddress` in the contract for the bounty before openQ can call the `claim` function.

Action Recommended:

Add

- Function that allows the funder to set payout address for the corresponding deposit
- Check whether the `_payoutAddress` parameter in `claim` function matches with the confirmed address from the funder

4. Close function can lock funds forever

Issue: **OPENQ-4**	Target:**BountyV0.sol**
Severity: **LOW**	Finding Type: **DYNAMIC**

Function `close` can be accidently called by the `openQ` owner. Once called, the contract would leave the deposited funds locked forever as status becomes `CLOSED` and the `claim` function would be reverted.

Action Recommended:

The `close` function should have more safe checks before closing the bounty: the function should check all deposits must either be claimed or refunded before changing status to `CLOSED`.

Note: This issue is considered LOW severity as the contract OpenQV0 will call to claim all deals before calling close function of the bounty contract. This would resolve the issue with locked funds.

5. Lacks of events

Issue: **OPENQ-5**	Target:**BountyV0.sol**
Severity: **INFORMATIONAL**	Finding Type: **DYNAMIC**

All critical functions in the contract do not emit important events. This could create issues when dealing with transaction histories of the bounty contract and showing transaction history on the front-end.

Action Recommended:

For each critical function including `receiveFunds`, `receiveNft`, `refundDeposit`, `claim`, and `close` should have corresponding events.

Note: This issue is considered INFORMATIONAL severity as all corresponding events are emitted in the OpenQV0 contract.

6. Potential out-of-gas

Issue: **OPENQ-6**	Target:**OpenQV0.sol**
Severity: **HIGH**	Finding Type: **DYNAMIC**

Function `claimBounty` loops over all deposits to claim all deposits before closing the bounty contract. There can be many deposits in the bounty contract, which potentially makes `claimBounty` out-of-gas while calling the claim function for all deposits.

Action Recommended:

Refactor `OpenQV0` contract to either

- Add function allowing for `claimBounty` to claim all deposits within a specified range of deposit indexes (in the `deposits` array)
- Or only claim a fixed number of deposits when `claimBounty` is called to avoid out-of-gas. The `claimBounty` function can then be called multiple times, each of which stores the last claim deposit index in the `OpenQV0` contract, and the `close` function of `BountyV0` can only be called once the last deposit is claimed.

Conclusion

Arcadia identified issues that occurred at hash **#46d09e18c5819f031d7fc75bc45d51495bed3ac5**. Remediations have occurred on later commits.

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, The Arcadia Group and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.