Voom
Security Audit Report

PREPARED FOR:

BlockSafe

ARCADIA CONTACT INFO
Email: audits@arcadiamgroup.com

Telegram: https://t.me/thearcadiagroup

Revision history

Date	Reason	Commit
12/16/2023	Addition of Customer Responses	

Table of Contents

Executive Summary

Introduction

Blocksafe engaged Arcadia to perform a security audit of their main smart contracts.

Review Team

1. Tuan “Anhnt” Nguyen - Security Researcher and Engineer
2. Joel Farris - Project Manager

Coverage

For this audit, we performed research, test coverage, investigation, and review of Voom’s main contracts followed by issue reporting, along with mitigation and remediation instructions as outlined in this report. The following code repositories, files, and/or libraries are considered in scope for the review.

Contracts	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/VoomToken/IVoomStaking.sol	4	4	2	1	1
contracts/VoomToken/VoomVesting.sol	73	73	50	9	26
contracts/VoomToken/IVoomVesting.sol	18	18	10	7	1
contracts/VoomToken/VoomStaking.sol	28	28	20	1	16
contracts/VoomToken/VoomToken.sol	34	34	22	4	22
contracts/VoomToken/VoomClaim.sol	64	64	44	7	42
contracts/VoomToken/IVoomToken.sol	10	7	4	1	7
contracts/VoomToken/IVoomClaim.sol	4	4	2	1	1

Methodology

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.

The followings are the steps we have performed while auditing the smart contracts:

- Investigating the project and its technical architecture overview through its documentation
- Understanding the overview of the smart contracts, the functions of the contracts, the inheritance, and how the contracts interface with each others thanks to the graph created by Solidity Visual Developer
- Manual smart contract audit:
    - Review the code to find any issue that could be exploited by known attacks listed by Consensys
    - Identifying which existing projects the smart contracts are built upon and what are the known vulnerabilities and remediations to the existing projects
    - Line-by-line manual review of the code to find any algorithmic and arithmetic related vulnerabilities compared to what should be done based on the project's documentation
    - Find any potential code that could be refactored to save gas
    - Run through the unit-tests and test-coverage if exists
- Static Analysis:
    - Scanning for vulnerabilities in the smart contracts using Static Code Analysis Software
    - Making a static analysis of the smart contracts using Slither
- Additional review: a follow-up review is done when the smart contracts have any new update. The follow-up is done by reviewing all changes compared to the audited commit revision and its impact to the existing source code and found issues.

Summary

There were **2** issues found, **0** of which were deemed to be 'critical', and **0** of which were rated as 'high'. At the end of these issues were found throughout the review of a rapidly changing codebase and not a final static point in time.

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

Findings in Manual Audit

(VM-1) Lack of validation for token amount received during token transfers into the contract.

Status

Addressed

Risk Level

Severity: Informational, likelihood: Low

Code Segment

```
function stake(uint256 _amount) external {
     voom.transferFrom(msg.sender, address(this), _amount);
     stakedBalance[msg.sender] += _amount;
```

```
  }
```

Description

Many tokens apply taxes to user transfers, resulting in the received amount not matching the transfer parameter. This emphasizes the need to validate the received token amounts.

Code location

```
contracts/VoomToken/VoomStaking.sol
```

Recommendation

Perform a double check on the received token amount following token transfers.

Response

The current intended usage of the token does not intend to support any tokens that have transfer taxes, as such this issue is not relevant to the current codebase.

(VM-2) Absence of validating input parameters of addVesting.

Status

Addressed

Risk Level

Severity: Medium, likelihood: Medium.

Code Segment

```
  function addVesting(address _beneficiary, VestingScheme calldata _vestingScheme) external
onlyOwner {
       require(_vestingScheme.cliffEnd < _vestingScheme.vestEnd, "VoomVesting: Cliff must
end before vest end");
       require(_vestingScheme.claimed == 0, "VoomVesting: Claimed must be 0");
       vestingSchemes[_beneficiary] = _vestingScheme;
   }
```

Description

Insufficient validation for the **_beneficiary** parameter to prevent overriding vesting schemes. Additionally, **vestEnd** should be greater than **block.timestamp.**

Code location

```
contracts/VoomToken/VoomVesting.sol
```

Recommendation

Ensure the validation of input parameters for both.

Response

The team is aware of this and will exercise caution when updating their vesting schedules as to not accidentally overwrite. In a future version, safeguards will be added.

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, Arcadia 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.