



# Security Audit Report for AQUA

**Date:** November 05, 2024 **Version:** 1.0

**Contact:** [contact@blocksec.com](mailto:contact@blocksec.com)

# Contents

<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 About Target Contracts . . . . .	1
1.2 Disclaimer . . . . .	1
1.3 Procedure of Auditing . . . . .	1
1.3.1 Software Security . . . . .	2
1.3.2 DeFi Security . . . . .	2
1.3.3 NFT Security . . . . .	2
1.3.4 Additional Recommendation . . . . .	2
1.4 Security Model . . . . .	2
<b>Chapter 2 Findings</b>	<b>4</b>
2.1 Note . . . . .	4
2.1.1 Pontential centralization risk . . . . .	4

## Report Manifest

Item	Description
Client	Aqualoan
Target	AQUA

## Version History

Version	Date	Description
1.0	November 05, 2024	First release

## Signature

**About BlockSec** BlockSec focuses on the security of the blockchain ecosystem and collaborates with leading DeFi projects to secure their products. BlockSec is founded by top-notch security researchers and experienced experts from both academia and industry. They have published multiple blockchain security papers in prestigious conferences, reported several zero-day attacks of DeFi applications, and successfully protected digital assets that are worth more than 14 million dollars by blocking multiple attacks. They can be reached at [Email](#), [Twitter](#) and [Medium](#).

# Chapter 1 Introduction

## 1.1 About Target Contracts

Information	Description
Type	Smart Contract
Language	Solidity
Approach	Semi-automatic and manual verification

The target of this audit is the on-chain address **AQUA**<sup>1</sup> of Aqualoan.

## 1.2 Disclaimer

This audit report does not constitute investment advice or a personal recommendation. It does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Any entity should not rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset.

This audit report is not an endorsement of any particular project or team, and the report does not guarantee the security of any particular project. This audit does not give any warranties on discovering all security issues of the smart contracts, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit cannot be considered comprehensive, we always recommend proceeding with independent audits and a public bug bounty program to ensure the security of smart contracts.

The scope of this audit is limited to the code mentioned in Section 1.1. Unless explicitly specified, the security of the language itself (e.g., the solidity language), the underlying compiling toolchain and the computing infrastructure are out of the scope.

## 1.3 Procedure of Auditing

We perform the audit according to the following procedure.

- **Vulnerability Detection** We first scan smart contracts with automatic code analyzers, and then manually verify (reject or confirm) the issues reported by them.
- **Semantic Analysis** We study the business logic of smart contracts and conduct further investigation on the possible vulnerabilities using an automatic fuzzing tool (developed by our research team). We also manually analyze possible attack scenarios with independent auditors to cross-check the result.
- **Recommendation** We provide some useful advice to developers from the perspective of good programming practice, including gas optimization, code style, and etc.

We show the main concrete checkpoints in the following.

---

<sup>1</sup><https://bscscan.com/address/0xD1Fb55d61c2494ced556536F898D969e2F6Dd2Bf>

### 1.3.1 Software Security

- \* Reentrancy
- \* DoS
- \* Access control
- \* Data handling and data flow
- \* Exception handling
- \* Untrusted external call and control flow
- \* Initialization consistency
- \* Events operation
- \* Error-prone randomness
- \* Improper use of the proxy system

### 1.3.2 DeFi Security

- \* Semantic consistency
- \* Functionality consistency
- \* Permission management
- \* Business logic
- \* Token operation
- \* Emergency mechanism
- \* Oracle security
- \* Whitelist and blacklist
- \* Economic impact
- \* Batch transfer

### 1.3.3 NFT Security

- \* Duplicated item
- \* Verification of the token receiver
- \* Off-chain metadata security

### 1.3.4 Additional Recommendation

- \* Gas optimization
- \* Code quality and style



**Note** The previous checkpoints are the main ones. We may use more checkpoints during the auditing process according to the functionality of the project.

## 1.4 Security Model

To evaluate the risk, we follow the standards or suggestions that are widely adopted by both industry and academy, including OWASP Risk Rating Methodology <sup>2</sup> and Common Weak-

---

<sup>2</sup>[https://owasp.org/www-community/OWASP\\_Risk\\_Rating\\_Methodology](https://owasp.org/www-community/OWASP_Risk_Rating_Methodology)

ness Enumeration <sup>3</sup>. The overall *severity* of the risk is determined by *likelihood* and *impact*. Specifically, likelihood is used to estimate how likely a particular vulnerability can be uncovered and exploited by an attacker, while impact is used to measure the consequences of a successful exploit.

In this report, both likelihood and impact are categorized into two ratings, i.e., *high* and *low* respectively, and their combinations are shown in Table 1.1.

**Table 1.1:** Vulnerability Severity Classification

<b>Impact</b>	<i>High</i>	High	Medium
	<i>Low</i>	Medium	Low
		<i>High</i>	<i>Low</i>
		<b>Likelihood</b>	

Accordingly, the severity measured in this report are classified into three categories: **High**, **Medium**, **Low**. For the sake of completeness, **Undetermined** is also used to cover circumstances when the risk cannot be well determined.

Furthermore, the status of a discovered item will fall into one of the following four categories:

- **Undetermined** No response yet.
- **Acknowledged** The item has been received by the client, but not confirmed yet.
- **Confirmed** The item has been recognized by the client, but not fixed yet.
- **Fixed** The item has been confirmed and fixed by the client.

---

<sup>3</sup><https://cwe.mitre.org/>

## Chapter 2 Findings

In total, we found **one** note.

ID	Severity	Description	Category	Status
1	-	Pontential centralization risk	Note	-

The details are provided in the following sections.

### 2.1 Note

#### 2.1.1 Pontential centralization risk

**Description** [AQUA](#) is an [ERC20](#) token that inherits from the [ERC20](#) library in [OpenZeppelin](#), the token decimal is [1e18](#). Potentially used for governance and to incentivize user interactions with the protocol. During contract initialization, [1e26 AQUA](#) tokens minted and allocated to a specified account<sup>1</sup>. If the private key of this account is lost or maliciously exploited, it could pose a risk to the protocol.

**Feedback from the Project** The team will transfer all tokens to multisig wallet to avoid private key leak.

---

<sup>1</sup><https://bscscan.com/address/0x108bd6239ed7a331d553838bd62b4d4009ae24a0>

