
# Security Audit Report

## POOLX Token

**30/3/2023**

**Revision: 30/3/2023**

**PREPARED FOR:**
**Poolz, Poolz.Finance**

**ARCADIA CONTACT INFO**
**Email:** audits@arcadiamgroup.com

**Telegram:** https://t.me/thearcadiagroup

# Table of Contents

# Executive Summary

## 1. Introduction and Audit Scope

A Representative Party of POOLZ ("**CLIENT**") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the following Poolz smart contracts on the **BNB Smart Chain** at contract address **0xbAeA9aBA1454DF334943951d51116aE342eAB255**

# 2. Audit Summary

### a. Audit 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
- Automated smart contract audit:
    - 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.

### b. Summary

There were **1** 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 | 0 |
| MEDIUM | 0 | 0 |
| LOW | 0 | 0 |
| INFORMATIONAL | 0 | 0 |

# Findings in Manual Audit

## 1. Minting should be granted to multi-signature wallets only

### Issue ID

POOLX-1

### Risk level

Severity: High, likelihood: Medium

### Code segment

```
abstract contract ERC20Mintable is Context, MinterRole, ERC20 {
    constructor() {
        Minter[_msgSender()] = true;
    }

    function mint(address to, uint256 amount) external virtual
onlyMinter {
        _mint(to, amount);
    }
}
```

### Description

The token **POOLX** contract has a mint function that could generate new tokens by minters. As token price could be highly affected by the change in token supply, it is highly recommended to have only one minter as a multisignature contract or a timelock contract to avoid any risks associated with leaking private keys if minters are EOAs (External Owned Account).

### Code location

`ERC20Mintable.sol`

### Proof of concept

–

### Recommendation

The team should set up the token contract to have only one minter as the multisignature contract.

### Updates regarding contract minting

The initial minter was revoked at transaction, and a new minter multisignature wallet was added by transaction.


# Automated Audit

## Static Analysis with Slither

We run a static analysis against the source code using Slither, which is a Solidity static analysis framework written in Python 3. Slither runs a suite of vulnerability detectors, prints visual information about contract details. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses.

We reviewed the results, all of the other issues found by Slither are false positives.

# Conclusion

Arcadia identified issues that occurred at contract **0xbAeA9aBA1454DF334943951d51116aE342eAB255** on the BNB Smart Chain.

# 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.