Smart Contract Security Audit Report

Introduction

A smart contract security review of the **Qoodo token contracts** was done by **Piyush shukla**, with a focus on the security aspects of the application's smart contracts implementation.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

About Auditor

Piyush shukla,is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Curretly he's Secured to 3 Hacker Rank globally in Smart Contract Security Platform) or reach out on Twitter Piyushshukla__

About ProtocolName

explanation what the protocol does, some architectural comments, technical documentation

Severity classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Impact - the technical, economic and reputation damage of a successful attack

Likelihood - the chance that a particular vulnerability gets discovered and exploited

Severity - the overall criticality of the risk

Security Assessment Summary

review commit hash **-** f0c64de

fixes review commit hash **-** efa12ef

Scope

The following smart contracts were in scope of the audit:

- `EcoSystemPool.sol`
- `QoodoToken.sol`
- `StakingQodTokens.sol`
- `YearlyVesting.sol`

Findings Summary

ID	Title	Severity	Status
[L-01]	whenPaused modifier always be first modifier	low	Fixed
[L-02]	Not follow CEI Pattern	low	Acknowledge
[G-01]	There’s no need to initialize i to its default value	low	Fixed
[G-02]	use ++i instead of i++	low	Fixed

Detailed Findings

[L-01] whenPaused modifier always be first modifier

In emergencyWithdraw() function . WhenPaused modifier always implement before any other modifier

Severity

Low

Recommendations

Implement WhenPaused before nonReentrant modifier

Status

Fixed

[L-02] Not follow CEI Pattern

Severity

Low

Status

Acknowledge

Gas Optimization

[G-01] use `++i` instead of `i++`

Status

Fixed

[G-02] There’s no need to initialize i to its default value

use uint256 i; instead uint256 i = 0;

Status

Fixed