



SMART CONTRACT SECURITY AUDIT

Qoodo

Token Contracts

CLIENT

Qoodo

LANGUAGE

Solidity

REVIEW

August 2023

DOCUMENT

Document information

Project	Qoodo, Token Contracts
Client	Qoodo
Lead auditor	Piyush Shukla
Platform	Not recorded in the original report
Language	Solidity
Repository	Not recorded in the original report
Commit	f0c64de
Methods	Not recorded in the original report
Report date	August 2023
Fix review commit	efa12ef

Document history

VERSION	DATE	CHANGE
1.0	August 2023	Original report issued View original report
1.1	October 2026	Reformatted into the Safe Edges standard template. Findings unchanged

This report was reformatted into the Safe Edges standard template in October 2026. Findings, severities, recommendations and code references are unchanged from the original report, which is preserved unchanged in the [Safe Edges Audit Portfolio on GitHub](#).

CONTENTS

Contents

01	Executive summary	3
02	Scope	4
03	Methodology	5
04	Severity and status	6
05	Findings summary	7
06	Detailed findings	8
	L-01 whenPaused modifier always be first modifier	8
	L-02 Not follow CEI Pattern	8
	G-01 use ++i instead of i++	8
	G-02 There's no need to initialize i to its default value	8
	Disclaimer	9

Executive summary

About Qoodo

Qoodo's token contracts cover the QOD token, an ecosystem pool, token staking and yearly vesting.

Summary of the review

Safe Edges reviewed Qoodo's token contracts: EcoSystemPool, QoodoToken, StakingQodTokens and YearlyVesting. Four Low issues were identified, two of them gas optimisations. Qoodo fixed three, with the fixes reviewed at commit efa12ef, and acknowledged one, which concerns the Checks-Effects-Interactions pattern.

Findings by severity



Findings by status

STATUS	ISSUES
Resolved	3
Acknowledged	1

Scope

Repository	Not recorded in the original report
Commit	f0c64de
Platform	Not recorded in the original report
Language	Solidity

IN-SCOPE FILES

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

Only the code at the commit above was reviewed. Changes made after this commit are not covered by this report.

Methodology

Methods recorded for this engagement: Not recorded in the original report.

Manual review

Auditors read the in-scope code line by line, tracing control flow, state changes and trust boundaries against the intended behaviour described by the client.

Threat modelling

We map privileged roles, external dependencies and value flows, then test the assumptions the code makes about each of them.

Functional and automated testing

Where tooling exists for the language and chain, we run static analysis and targeted tests to confirm suspected issues and check edge cases.

Fix review

After the client addresses the findings, we review the updated code and record the final status of each issue.

Severity and status

Severity levels

SEVERITY	DEFINITION
Critical	Direct loss or permanent lock of funds, or full compromise of the protocol, with few or no preconditions. Must be fixed before deployment.
High	Significant loss of funds, broken core functionality or manipulation of critical state under realistic conditions. Should be fixed before deployment.
Medium	Limited loss or degraded functionality, or issues that need specific conditions or privileged access to exploit. Should be fixed.
Low	Minor impact, low likelihood, or deviations from best practice. Fix when practical.
Informational	Code quality, documentation, gas and style observations with no direct security impact.

Status definitions

STATUS	DEFINITION
Resolved	The issue was fixed and the fix was verified.
Partially resolved	The risk was reduced but not fully removed.
Acknowledged	The client accepted the issue without changing the code.
Open	The issue was not addressed at the time of the report.
Not recorded	The original report did not record a resolution status for this issue.

Findings summary

ID	TITLE	SEVERITY	STATUS
L-01	whenPaused modifier always be first modifier	Low	● Resolved
L-02	Not follow CEI Pattern	Low	● Acknowledged
G-01	use ++i instead of i++	Low	● Resolved
G-02	There's no need to initialize i to its default value	Low	● Resolved

Detailed findings

L-01 whenPaused modifier always be first modifier

Severity **Low** Status ● Resolved

DESCRIPTION

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

RECOMMENDATIONS

Implement WhenPaused before nonReentrant modifier

L-02 Not follow CEI Pattern

Severity **Low** Status ● Acknowledged

G-01 use ++i instead of i++

Severity **Low** Status ● Resolved Category Gas

G-02 There's no need to initialize i to its default value

Severity **Low** Status ● Resolved Category Gas

DESCRIPTION

use `uint256 i;` instead `uint256 i = 0;`

Disclaimer

This report is based on the code at the commit listed in the Scope section and on information the client provided during the review. It reflects Safe Edges' assessment at the time of the review and is not a guarantee that the code is free of vulnerabilities.

Changes made after the reviewed commit, deployment configuration, off-chain components and third-party dependencies are outside the scope of this report unless stated otherwise.

Security is ongoing. We recommend continued testing, monitoring and bug bounty coverage alongside any audit.

This report is not investment advice and is not an endorsement of the project, its team or any token.



Safe Edges is a Web3 security firm and official security partner for the Fuel, ENI and XI EcoChain ecosystems. We work across smart contract audits, Rust and WASM reviews, ZK circuit review, AI and MCP security, and penetration testing.

200+

PROJECTS AUDITED

\$4B+

TVL SECURED

0

POST-AUDIT HACKS