



SMART CONTRACT SECURITY AUDIT

Qoodo

Vesting Contract

CLIENT

Qoodo

LANGUAGE

Solidity

REVIEW

October 2023

DOCUMENT

Document information

Project	Qodo, Vesting Contract
Client	Qodo
Lead auditor	Piyush Shukla
Platform	Not recorded in the original report
Language	Solidity
Repository	Not recorded in the original report
Commit	Not recorded in the original report
Methods	Not recorded in the original report
Report date	October 2023

Document history

VERSION	DATE	CHANGE
1.0	October 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
	H-01 Missing refund token parameter in revoke function which causes a loss of funds	8
	H-02 In addvesting schedule function missing only owner modifier. So any user can create a vesting schedule with infinite tokens. It's a direct loss of funds	8
	M-01 Missing zero check in payee address	9
	L-01 Not follow CEI pattern in claim function	9
	L-02 Missing event in claim function	9
	Disclaimer	10

Executive summary

About Qodo

Qodo's vesting contract creates vesting schedules for payees, lets them claim vested tokens, and lets the owner revoke revocable grants.

Summary of the review

Safe Edges reviewed Qodo's vesting contract. Five issues were identified: two High, one Medium and two Low. The High issues were a missing owner check on `addVestingSchedule`, which let anyone create a schedule with unlimited tokens, and a revoke path that left unclaimed tokens stranded. Qodo fixed all five.

Findings by severity



Findings by status

STATUS	ISSUES
Resolved	5

Scope

Repository	Not recorded in the original report
------------	-------------------------------------

Commit	Not recorded in the original report
--------	-------------------------------------

Platform	Not recorded in the original report
----------	-------------------------------------

Language	Solidity
----------	----------

IN-SCOPE FILES

Vesting Contract

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
H-01	Missing refund token parameter in revoke function which causes a loss of funds	High	● Resolved
H-02	In addvesting schedule function missing only owner modifier. So any user can create a vesting schedule with infinite tokens. It's a direct loss of funds	High	● Resolved
M-01	Missing zero check in payee address	Medium	● Resolved
L-01	Not follow CEI pattern in claim function	Low	● Resolved
L-02	Missing event in claim function	Low	● Resolved

Detailed findings

H-01 Missing refund token parameter in revoke function which causes a loss of funds

Severity **High** Status ● Resolved

DESCRIPTION

In the current contract context, you are using the 'addVestingSchedule' function, which is responsible for scheduling the 'revoke' function in the 'VestingTrustee.' The 'revoke' function in 'VestingTrustee' allows the owner to revoke an address's vesting grant in case it is revocable. However, if those tokens are not claimed by users and the owner revokes their vesting, all the unclaimed tokens are lost because there is no refund or token transfer function in place. This results in a direct loss of the tokens.

MITIGATION

Implement `_token.safeTransfer(owner(), remainingAmount);`

H-02 In addvesting schedule function missing only owner modifier. So any user can create a vesting schedule with infinite tokens. It's a direct loss of funds

Severity **High** Status ● Resolved

DESCRIPTION

There is an access control issue in the 'addVestingSchedule' function. It lacks the onlyOwner modifier, which means that any user can create a vesting schedule with an infinite number of tokens. This could result in a direct loss of tokens.

MITIGATION

Add the onlyOwner modifier in the addVestingSchedule function.

M-01 Missing zero check in payee address

Severity **Medium** Status ● Resolved

DESCRIPTION

In the 'addVestingSchedule' function, there is a missing check for zero addresses. If accidentally, a zero address is added as the payee, all the vested tokens could be locked forever, resulting in a loss of funds.

MITIGATION

```
require(_payee != address(0), "TokenVesting: payee is the zero address" )
```

L-01 Not follow CEI pattern in claim function

Severity **Low** Status ● Resolved

DESCRIPTION

Not following CEI in many functions, including the claim function, which is a bad practice.

L-02 Missing event in claim function

Severity **Low** Status ● Resolved

DESCRIPTION

Missing an event in the claim function.

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