Smart Contract Audit Report

Introduction

A smart contract security review of the **Vesting contract** was conducted by **Piyush Shukla**, with a focus on the security aspects of the application's smart contract implementation.

Disclaimer

A smart contract security review can never guarantee the complete absence of vulnerabilities. This is a time, resource, and expertise-bound effort, where I attempt to find as many vulnerabilities as possible. I cannot 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 and reviews. Currently, he's ranked 3rd globally in Smart Contract Security on HackerRank. You can reach out to him on Twitter: Piyushshukla__.

About ProtocolName

(Explanation of what the protocol does, architectural comments, technical documentation)

Observations

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) - **Vesting contract**

(Fixes review commit hash) -

Scope

The following smart contracts were in scope of the audit:

- `Vesting Contract`

Findings Summary

ID	Title	Severity	Status
[H-01]	Missing refund token parameter in revoke function which causes a loss of funds	High	Fixed
[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	Fixed
[M-01]	Follow CEI pattern in claim function	Medium	Fixed
[L-01]	Missing zero check in payee address	Low	Fixed
[L-02]	Missing event in claim function	Low	Fixed

Detailed Findings

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

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);`

Severity

High

Status

Fixed

[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

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.

Severity

High

Status

Fixed

[M-01] Missing zero check in payee address

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" )
```

Severity

Medium

Status

Fixed

[L-01] Not follow CEI pattern in claim function

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

Severity

Low

Status

Fixed

[L-02] Missing event in claim function

Missing an event in the claim function.

Severity

Low

Status

Fixed

Powered by Safe Edges