Smart Contract Audit Report

1. Overview

- **Project Name:** ICO
- **Contracts Audited:**
  - File: Token.sol
  - File: ICO.sol
  - File: Claim.sol
- **Code base:** Provided by file
- **Date:** 24/06/2025

2. Executive Summary

This audit aims to assess the security of the smart contracts for the ICO protocol. We identified issues:

- High Risk
- Medium Risk
- Low Risk
- Informational / Gas Optimization Suggestions

All high and medium severity issues have been addressed / partially addressed / not addressed by the development team at the time of publishing this report.

3. Methodology

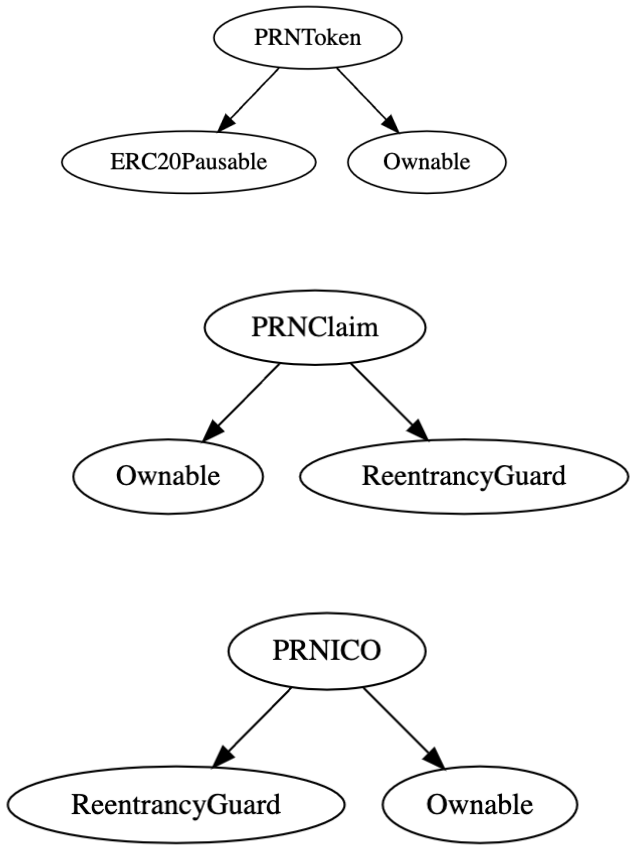
Our audit involved the following steps:

1. **Manual Code Review** – Analyzed logic for correctness, authorization, error handling, etc.
2. **Automated Testing** – Used static analysis tools such as Slither, MythX, or Hardhat plugins.
3. **Gas Usage Review** – Identified areas for optimization.
4. **Dependency Analysis** – Assessed third-party libraries (e.g., OpenZeppelin).

4. Risk Classification

Severity	Description
High	Critical vulnerabilities that can lead to loss.
Medium	Significant issues but not critical.
Low	Minor issues, often best practices.
Informational	No direct risk but useful suggestions.

5. Inheritance Graph



6. Ownership Privileges

Token.sol

- The owner can pause/unpause the token.
- The owner can burn tokens without approval.

ICO.sol

- The owner can set the presale start time (setPresaleStart).
- The owner can set the public sale start time (setPublicStart).
- The owner can set the end time (setEndTime).
- The owner can pause/unpause the ICO (pauseICO/resumeICO).
- The owner can set the presale price in cents (setPresalePriceCents).
- The owner can set the public sale price in cents (setPublicPriceCents).
- The owner can set a soft cap in cents (setSoftCapCents).
- The owner can set a hard cap in cents (setHardCapCents).
- The owner can withdraw ETH from the contract (withdrawETH).
- The owner can recover leftover tokens (recoverLeftoverTokens).

Claim.sol

- The Owner can withdraw any ERC20 tokens present in the contract.
- The Owner can pause/resume the claiming functionality.

7. Findings

7.1 High Severity

[H-01] Unrestricted Mid-Sale Parameter Modifications

- **Description:** The contract allows the owner to modify critical ICO parameters (prices, caps, and timelines) during an active sale without any restrictions or timelocks. This creates significant centralization risk and potential for manipulation, as the owner could alter sale conditions after users have already participated. For example, the owner could increase prices, extend timelines, or modify caps to manipulate market dynamics, directly impacting participant positions and undermining the fairness and transparency of the ICO.

- **Mitigation:** Implement a parameter locking mechanism that activates when the sale starts. Any future parameter changes should require a timelock period (e.g., 48 hours) and only affect subsequent sale phases. Add a transparent upgrade schedule and emergency pause functionality for critical issues instead of direct modifications.

- File: ICO.sol

- Line: 81-95, 264-268, 284-291

- **Status:** Open

7.2 Medium Severity

[M-01] The owner can lock the contract.

- **Description:** The current contract allows the owner to pause token transfers indefinitely with no time restrictions or community safeguards. This creates a severe centralization risk where tokens could be permanently frozen, either maliciously or due to compromised owner keys. It is recommended to implement the maximum pause duration.

- **Mitigation:** Add a force-unpause mechanism after the max duration expires. Require a time lock delay before pausing execution.

- File: Token.SOL

- Line: 19-25

- **Status:** Open

[M-02] Unauthorized Token Burning Power

- **Description:** The contract's burn function allows the owner to burn tokens from any address without prior approval.
- **Mitigation:** Replace the current burn function with two safer alternatives: the Self-burn function, where users can only burn their own tokens, or the BurnFrom function requires explicit allowance approval.
- File: Token.sol
- Line: 40-44
- **Status:** Open

[M-03] Unlimited ICO Pause Duration

- **Description:** The Owner can pause the ICO indefinitely without time restrictions. This centralized control could be abused to manipulate market timing or prevent user participation, effectively locking the ICO state and blocking purchases for an unlimited duration. Implement maximum pause duration and allow forced resume after timeout.
- **Mitigation:** Add a pause cooldown period between consecutive pauses.
- File: ICO.sol
- Line: 96-99, 101-105
- **Status:** Open

[M-04] Missing Parameter Thresholds and Validation.

- **Description:** Critical contract parameters lack validation checks and thresholds. The owner can set prices to zero or extremely high values, modify caps without maintaining soft cap < hard cap relationship, and set illogical timeline sequences. This could break core contract functionality or enable manipulation. For example, setting presale price higher than public price, or setting hard cap below soft cap would create invalid sale conditions and potentially lock user funds.
- **Mitigation:** Implement strict bounds checking and logical validation for all parameter modifications. Add minimum/maximum thresholds for prices, enforce cap relationships, and validate timeline sequences. Use constants for boundaries and require proper validation in all setter functions.
- File: ICO.sol
- Line: 264-268, 284-291
- **Status:** Open

7.3 Low Severity

[L-01] Missing events arithmetic.

- **Description:** It is recommended to emit all the critical parameter changes.
- File/Line: Token.sol[L19-44], Claim.sol[122-125], ICO.sol[260, 264, 268, 284, 288]
- **Status:** Open

[L-02] Floating pragma solidity version.

- **Description:** Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.
- File: Token.sol, Claim.sol, ICO.sol
- Line: 2
- **Status:** Open

[L-03] Missing Validation for Vesting Parameters.

- **Description:**
  The claim() functionaccepts startTime and durationYears from ico.vestingPreferences() with out validation. This could lead to tokens being immediately claimable (if startTime=0) or locked indefinitely (if startTime/durationYears are extreme values). Add require statements to validate that startTime is reasonable (not 0/too far future) and durationYears is within acceptable bounds
- File: Claim.sol
- Line: 52-56
- **Status:** Open

[L-04] Missing Zero Address Validation.

- **Description:** The function allows withdrawal to any address, including a zero address. If the owner accidentally provides an address(0), tokens could be permanently lost. Add a require statement to validate user address is not zero.
- File: Claim.sol
- Line: 122-125
- **Status:** Open

7.4 Informational/Optimization Severity

[I-01] Missing Zero Address Validation.

- **Description:** Add a require statement to validate user address is not zero, though this is more of a code quality suggestion than a security issue. The function is otherwise secure and serves its purpose well as a view function to check claimable amounts.
- File: Claim.sol[L118], ICO.sol[L65,66]
- Line: 118
- **Status:** Open

[I-02] Missing non-reentrant protection.

- **Description:** The function performs external token operations (balanceOf and transfer) without the nonReentrant modifier. While most ERC20 implementations are safe, malicious tokens could potentially exploit reentrancy during these calls, especially since state reads and transfers occur without protection. Adding nonReentrant would only increase gas costs unnecessarily
- File: Claim.sol[122-126], ICO.sol[L204-210]
- **Status:** Open

[I-03] Missing isContract check.

- **Description:** The missing isContract check is NOT a security issue or vulnerability. Adding it would reduce functionality without providing any security benefits.
- File: ICO.sol
- Line: 64-73
- **Status:** Open

8. Conclusion

The smart contracts are generally well-written and follow modern best practices. After addressing the findings noted, the contracts should be considered safe for deployment, assuming no changes are made without further auditing.

9. Disclaimer

This audit report is not the final deployed version of the contract. So, any changes made after the audit will be considered out of the scope of the audit, and we will not be responsible for any changes related to that.