Smart Contract Audit Report
TestContract

Contract Address: N/A
Audit Date: 2025-03-24

Executive Summary

Overall Risk Rating: low
This audit identified 3 vulnerabilities and 1 gas optimization opportunities.

Vulnerabilities

Reentrancy (High)
Location: Line 26 in contract.sol
Description: Function transfer makes an external call and then changes state afterward. This allows an attacker to re-enter the function before state changes are applied.
Recommendation: Follow the checks-effects-interactions pattern: first perform all checks, then make state changes, and finally interact with external contracts.

Reentrancy (High)
Location: Line 41 in contract.sol
Description: Function transferFrom makes an external call and then changes state afterward. This allows an attacker to re-enter the function before state changes are applied.
Recommendation: Follow the checks-effects-interactions pattern: first perform all checks, then make state changes, and finally interact with external contracts.

Reentrancy (High)
Location: Line 63 in contract.sol
Description: Function mint makes an external call and then changes state afterward. This allows an attacker to re-enter the function before state changes are applied.
Recommendation: Follow the checks-effects-interactions pattern: first perform all checks, then make state changes, and finally interact with external contracts.

Gas Optimization

Loop condition reads array length in each iteration
Location: Line 74 in contract.sol
Gas Saved: Up to 100+ gas per loop iteration
Recommendation: Cache array length in a local variable before the loop to save gas.

Compliance Results

ERC20

Status: Non-compliant
Missing Requirements: Missing function: totalSupply - Returns the total token supply, Missing function: balanceOf - Returns the account balance of an account, Missing function: allowance - Returns the amount of tokens the spender is allowed to spend on behalf of the owner, Missing event: Transfer - Must be emitted when tokens are transferred, including zero value transfers, Missing event: Approval - Must be emitted when approval is set or changed
Recommendations: Consider inheriting from the OpenZeppelin ERC20 implementation to ensure full compliance., Always emit Transfer and Approval events when token balances or allowances change., The transfer function should emit the Transfer event., Consider adding checks to prevent transfers to the zero address (0x0).

ERC721

Status: Non-compliant
Missing Requirements: Missing function: balanceOf - Returns the number of NFTs owned by an address, Missing function: ownerOf - Returns the owner of a specific NFT, Missing function: safeTransferFrom - Safely transfers the ownership of a given token ID to another address with data parameter, Missing function: safeTransferFrom - Safely transfers the ownership of a given token ID to another address, Missing function: getApproved - Gets the approved address for a token ID, Missing function: setApprovalForAll - Approves or removes operator for all tokens for the caller, Missing function: isApprovedForAll - Tells whether an operator is approved by an owner, Missing event: Transfer - Emitted when tokenId is transferred from from to to, Missing event: Approval - Emitted when owner approves tokenId to approved, Missing event: ApprovalForAll - Emitted when owner enables or disables operator for all tokens
Recommendations: Consider inheriting from the OpenZeppelin ERC721 implementation to ensure full compliance., Always emit appropriate events (Transfer, Approval, ApprovalForAll) when state changes., Consider adding reentrancy protection to safeTransferFrom functions to prevent potential reentrancy attacks., Consider adding checks to prevent transfers to the zero address (0x0) to avoid permanent loss of NFTs., Ensure tokenURI and ownerOf functions check that the tokenId exists before returning data., Consider implementing ERC721Metadata interface (name, symbol, tokenURI) for better NFT compatibility., Consider implementing ERC721Enumerable interface for better marketplace and indexer support.

Recommendations

- [object Object]
- [object Object]
- [object Object]

Contract Source

```
// SPDX-License-Identifier: MITÐ
pragma solidity ^0.8.0;Ð
Ð
/**Ð
 * @title SampleTokenÐ
 * @dev A sample ERC20 token with some deliberate vulnerabilities for testingÐ
 */Ð
contract SampleToken {Ð
    string public name = "Sample Token";Ð
    string public symbol = "SMPL";Ð
    uint8 public decimals = 18;Ð
    uint256 public totalSupply;Ð
    Ð
    mapping(address => uint256) public balanceOf;Ð
    mapping(address => mapping(address => uint256)) public allowance;Ð
    Ð
    address public owner;Ð
    Ð
    constructor(uint256 initialSupply) {Ð
        totalSupply = initialSupply * 10 ** uint256(decimals);Ð
        balanceOf[msg.sender] = totalSupply;Ð
        owner = msg.sender;Ð
    }Ð
    Ð
    // Vulnerability 1: No event emissionÐ
    function transfer(address to, uint256 value) public returns (bool success) {Ð
        require(balanceOf[msg.sender] >= value, "Insufficient balance");Ð
        Ð
        balanceOf[msg.sender] -= value;Ð
        balanceOf[to] += value;Ð
        return true;Ð
    }Ð
    Ð
    // Vulnerability 2: No check for zero addressÐ
    function approve(address spender, uint256 value) public returns (bool success) {Ð
        allowance[msg.sender][spender] = value;Ð
        return true;Ð
    }Ð
    Ð
    // Vulnerability 3: No SafeMath, potential for overflow/underflow in older Solidity versionsÐ
    function transferFrom(address from, address to, uint256 value) public returns (bool success) {Ð
        require(balanceOf[from] >= value, "Insufficient balance");Ð
        require(allowance[from][msg.sender] >= value, "Insufficient allowance");Ð
        Ð
        balanceOf[from] -= value;Ð
        balanceOf[to] += value;Ð
        allowance[from][msg.sender] -= value;Ð
        return true;Ð
    }Ð
    Ð
    // Vulnerability 4: Reentrancy vulnerabilityÐ
    function withdraw(uint256 amount) public {Ð
        require(balanceOf[msg.sender] >= amount, "Insufficient balance");Ð
        Ð
        // Vulnerability: state change after external callÐ
        (bool success, ) = msg.sender.call{value: amount}("");Ð
```

```
        require(success, "Transfer failed");Ð
        Ð
        balanceOf[msg.sender] -= amount;Ð
    }Ð
    Ð
    // Vulnerability 5: Centralization riskÐ
    function mint(address to, uint256 amount) public {Ð
        require(msg.sender == owner, "Only owner can mint");Ð
        Ð
        totalSupply += amount;Ð
        balanceOf[to] += amount;Ð
    }Ð
    Ð
    // Vulnerability 6: Gas inefficiencyÐ
    function batchTransfer(address[] memory recipients, uint256[] memory values) public returns (bool) {Ð
        require(recipients.length == values.length, "Arrays length mismatch");Ð
        Ð
        for (uint i = 0; i < recipients.length; i++) {Ð
            require(transfer(recipients[i], values[i]));Ð
        }Ð
        Ð
        return true;Ð
    }Ð
    Ð
    // Missing some standard ERC20 functions for compliance checker testingÐ
}
```