SMART CONTRACT AUDIT REPORT

BunnyNotes Contract

By: Alexis Davidson

Introduction

This audit report highlights the overall security of the BunnyNotes smart contract.
With this report, I have tried to ensure the reliability of the smart contract to the best of my capability by completing the assessment of their system's architecture and smart contract codebase.

Auditing approach and Methodologies applied

In this audit I consider the following crucial features of the code.

- Whether the code is secure
- Whether the code meets the best coding practices
- Whether the code is easily readable
- Whether the code is gas optimized

The audit has been performed according to the following procedure:

1. Running the tests and checking their coverage of the requirements
2. Running automated analysis tools (like slither, linters, static analysis, etc), manually verifying (reject or confirm) all the issues found by tools
3. Performing a manual Analysis
    a. Inspecting the code line by line and compare them with the specification
    b. Manually analyzing the code for security vulnerabilities
    c. Assessing the overall project structure, complexity & quality
    d. Checking whether all the libraries used in the code of the latest version
    e. Analysis of security on-chain data
    f. Analysis of the failure preparations to check how the smart contract performs in case of bugs and vulnerability
4. Discussing (and repeating steps as needed)
5. Writing the report: All the gathered information is described in this report

Audit details

Project Name: BunnyNotes
Language: Solidity
Platform and tools: Hardhat, VScode, slither and other tools mentioned in the automated analysis section.

Audit Goals

The focus of this audit was to verify whether the smart contract is secure, resilient, and working properly according to the specs. The audit activity can be grouped in three categories.

Security: Identifying the security-related issue within each contract and system of contracts.
Sound architecture: Evaluating the architect of a system through the lens of established smart contract best practice and general software practice.

Code correctness and quality: A full review of contract source code. The primary area of focus includes:

- Correctness
- Section of code with high complexity
- Readability
- Quantity and quality of test coverage

Security

Every issue in this report was assigned a severity level from the following:

High severity issues

Issues mentioned here are critical to smart contract performance and functionality and should be fixed before moving to mainnet.

Medium severity issues

This could potentially bring the problem in the future and should be fixed.

Low severity issues

These are minor details and warnings that can remain unfixed but would be better if it got fixed in the future.

No. of issue per severity

Severity	High	Medium	Low
Open	0	1	2

Manual audit

Following are the reports from our manual analysis.

High severity issues

No high severity issue found

Medium severity issues

1. Reentrancy

Reentrancy in BunnyNotes.withdraw(uint256[8],bytes32,bytes32,address) (contracts/BunnyNotes.sol#175-224):
External calls:
- require(bool,string)(verifier.verifyProof((_proof[0],_proof[1]),((_proof[2],_proof[3]),(_proof[4],_proof[5])),(_proof[6],_proof[7]),(uint256(_nullifierHash),uint256(_commitment),uint256(uint160(_recipient)))),Invalid Withdraw proof) (contracts/BunnyNotes.sol#186-198)
State variables written after the call(s):
- commitments[_commitment].recipient = _recipient (contracts/BunnyNotes.sol#201)

- commitments[_commitment].spentDate = block.timestamp (contracts/BunnyNotes.sol#202)
- nullifierHashes[_nullifierHash] = true (contracts/BunnyNotes.sol#200)

Description
Detection of the reentrancy bug. Do not report reentrancies that involve Ether

Recommendation
Apply the check-effects-interactions pattern.

Low severity issues

1. lacks a zero-check on

BunnyNotes.constructor(IVerifier,address)._feelessToken (contracts/BunnyNotes.sol#71) lacks a zero-check on :
- feelessToken = _feelessToken (contracts/BunnyNotes.sol#74)

BunnyNotes.setFeelessToken(address).newFeelesstoken (contracts/BunnyNotes.sol#82) lacks a zero-check on :
- feelessToken = newFeelesstoken (contracts/BunnyNotes.sol#84)

Description
Detect missing zero address validation.

Recommendation
Check that the address is not zero.

2. is not in mixedCase

Parameter BunnyNotes.depositEth(bytes32,uint256)._commitment (contracts/BunnyNotes.sol#94) is not in mixedCase
Parameter BunnyNotes.depositToken(bytes32,uint256,address)._commitment (contracts/BunnyNotes.sol#125) is not in mixedCase
Parameter BunnyNotes.withdraw(uint256[8],bytes32,bytes32,address)._proof (contracts/BunnyNotes.sol#176) is not in mixedCase
Parameter BunnyNotes.withdraw(uint256[8],bytes32,bytes32,address)._nullifierHash (contracts/BunnyNotes.sol#177) is not in mixedCase
Parameter BunnyNotes.withdraw(uint256[8],bytes32,bytes32,address)._commitment (contracts/BunnyNotes.sol#178) is not in mixedCase

Parameter BunnyNotes.withdraw(uint256[8],bytes32,bytes32,address)._recipient (contracts/BunnyNotes.sol#179) is not in mixedCase
Parameter BunnyNotes.isSpent(bytes32)._nullifierHash (contracts/BunnyNotes.sol#230) is not in mixedCase
Variable BunnyNotes._owner (contracts/BunnyNotes.sol#37) is not in mixedCase
Constant BunnyNotes.feeDivider (contracts/BunnyNotes.sol#41) is not in UPPER_CASE_WITH_UNDERSCORES

Description
Solidity defines a naming convention that should be followed.

Recommendation
Follow the Solidity naming convention.

Note

The unit tests are running green and covering the smart contract's code properly.
The code is cleanly written and easy for the public to read.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. I always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the code.

Summary

The use of the smart contract is simple and the code is relatively small. Altogether the code is well written and easily readable.
There was no critical error found in the existing code and the unit tests have a very good coverage percentage. The few issues found are negligible are mostly about naming conventions.