NRGY Smart Contract Code Audit

March 1, 2021

Security Audit and Analysis

Footprints Tech
https://footprintstech.in/
Ayusha Chitransh
audit@footprints.tech.in

Table of Contents

1. DISCLAIMER

This is a report of our finding in accordance with good industry practices as of the date of this audit report, in relation to: vulnerabilities and all detectable issues discovered by our firm in the smart contract source code, the items of which are set out in this report include but are not limited to, auditing the source code, compiling, deploying, and checking the performance of the intended functions. It is important to note that you should not solely rely on this audit report and agree to hold Footprints Tech harmless for any findings of this audit, or any method performed in the production of this comprehensive audit. By reading this audit in whole or in part, you agree to the terms of this disclaimer. This report is intended to provide technical information to assist you in evaluating the security of the smart contracts covered in this audit. This audit does not constitute investment advice, endorsement or constitutes a guarantee of any kind. You agree to hold Footprints Tech and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) harmless for any representations contained in this audit report. We make no warranty as to the accuracy of the information contained herein or the completeness of this report. The audit report is provided "as is", without any conditions, warranties, except as set out in this disclaimer. This audit makes no statements or warrantees about the utility, safety of the smart contract code and or its suitability of this projects business model, regulatory regime. Other than what is contained in the audit report, we will make no further statements or comments about fitness or status of this protocol or the merits of the smart contract. The analysis was derived by the application of both proprietary and open-source analysis tools. The output of various security and smart contract auditing tools were used in this analysis. In addition to our detailed analysis these tools contributed significantly in the production of this audit report.

2. INTRODUCTION

2.1. DOCUMENT PURPOSE

The findings of the review are presented in this document.

Project Name	NRGY
Description	Staking Protocol
Platform	Ethereum, Solidity
Delivery Date	March 1, 2021
Method of Audit	Dynamic Analysis, Manual Review

2.2. REVIEW GOALS & FOCUS

Sound Architecture

This review and comprehensive audit include both the objective findings from the contract code as well as subjective assessments of the overall code architecture and design choices. Given the subjective nature of certain findings the NRGY code contributors have been given the opportunity to include its responses in this audit report. The resources for this audit have been provided by a number of NRGY contract participant.

Smart Contract Best Practices

This review evaluated whether the codebase follows the current established best practices for smart contract development.

Code Correctness

This review will evaluate whether the code does what it is intended to do and through our best efforts determine that the code is error free and secure.

Code Quality

This review will evaluate whether the code has been written in a way that ensures readability maintainability and that it contains no known security issues.

Security

This review will look for exploitable security vulnerabilities.

2.3. TERMINOLOGY

This review uses the following terminology:

Severity Terms

Measure the magnitude of an issue.

Minor

Minor issues are generally subjective in nature, or potentially deal with topics like "best practices" or "readability". Minor issues in general will not indicate an actual problem or bug in code. The maintainers should use their own judgment as to whether addressing these issues improves the codebase.

Major

Major issues will be things like bugs or detectable security vulnerabilities. These issues may not be directly exploitable such as requiring a specific condition to arise to be exploited. Left unaddressed these issues are highly likely to cause problems with the operation of the contract or lead to a situation which allows the system to be exploited in some way.

Critical

Critical issues are directly exploitable bugs or security vulnerabilities. Left unaddressed these issues are highly likely or guaranteed to cause major problems or potentially a full failure in the operations of the contract.

3. SOURCE CODE

3.1. REPOSITORY

This audit has been done for smart contracts in the NRGY repository.

3.2. SMART CONTRACTS

The code audit carried out was limited to the smart contract(s) files shared over the repository as below:

1. ENERGY.sol
2. NRGYMarketMaker.sol

3.3. STRUCTURE & ACCESS CONTROL

The contract is designed to be immutable. In other word, the smart contracts are fully decentralized. There is no governance provisions or central ownership inside the contract.

4. FINDINGS

This section contains detailed issues and analysis.

4.1 CRITICAL ISSUES

No Critical Issues Discovered

4.2 MAJOR ISSUES

Issue#: 001 No variable to update fee rewards

Contract Name#: NRGYMarketMaker.sol

Contract Location#: /contracts/NRGYMarketMaker.sol

Codes commit#:

1ecc2f8f53719f2ced170c929ae114020c6628c9

Description:

In function *_distributeFees (uint256 _amount)* there is no variable to calculate the sum of fee rewards.

Action: Fixed the issue at Commit# 26d6e3cc5eb6e8a53b62b6a2899e7a3f5648ea14

4.3 MINOR ISSUES

Issue#: 001 Variable initialization can be removed and variable can be defined constant to save gas cost

Contract Name#: NRGYMarketMaker.sol

Contract Location#: /contracts/NRGYMarketMaker.sol

Codes commit#:

1ecc2f8f53719f2ced170c929ae114020c6628c9

Description:

We can remove the initialization of variable *totalRewards, totalFeeRewards, rewardsAvailableInContract, feeRewardsAvailableInContract, feeRewardsCount, totalStakeUsers* and make variable *percentageDivider, unstakeFees* constant.

Action: Fixed the issue at Commit# 26d6e3cc5eb6e8a53b62b6a2899e7a3f5648ea14

Issue#: 002 Variable initialization can be removed and variable can be defined constant to save gas cost

Contract Name#: ENERGY.sol

Contract Location#: /contracts/ENERGY.sol

Codes commit#:

1ecc2f8f53719f2ced170c929ae114020c6628c9

Description:

We can remove the initialization of variable *weekCount* and make variable *totalWeeks, timeStep* constant.

Action: Fixed the issue at Commit# 26d6e3cc5eb6e8a53b62b6a2899e7a3f5648ea14

Outdated Solidity version: The latest solidity version at the time of this writing is 0.8.0. The contract uses version 0.7.6.

Action: Acknowledged. Didn't changed the versions because thorough testing had been done on current version.

5. GENERAL COMMENTS

Following are some general comments which should be fixed/modified for better code quality:

- The contracts can be optimized by using require and assert at their proper places. Here is a link to study about it, https://ethereum.stackexchange.com/questions/15166/difference-between-requireand-assert-and-the-difference-between-revert-and-thro

- Consider adding prefix underscore to internal methods and variable to differentiate from state.