



Energyfi

Token

SMART CONTRACT AUDIT

04.02.2022

Made in Germany by Chainsulting.de



Table of contents

1. Disclaimer.....	3
2. About the Project and Company	4
2.1 Project Overview.....	5
3. Vulnerability & Risk Level	6
4. Auditing Strategy and Techniques Applied.....	7
4.1 Methodology	7
4.2 Used Code from other Frameworks/Smart Contracts	8
4.3 Tested Contract Files	9
4.4 Metrics / CallGraph.....	10
4.5 Metrics / Source Lines & Risk.....	11
4.6 Metrics / Capabilities	12
4.7 Metrics / Source Unites in Scope	13
5. Scope of Work.....	14
5.1 Manual and Automated Vulnerability Test.....	15
5.1.1 Missing natspec documentation.....	15
5.2 SWC Attacks	16
5.3 Verify Claims	20
6. Executive Summary.....	20
7. Deployed Smart Contract	20

1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Energyfi. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (01.02.2022)	Layout
0.2 (01.02.2022)	Test Deployment
0.5 (01.02.2022)	Automated Security Testing Manual Security Testing
0.6 (02.02.2022)	Testing SWC Checks
0.7 (02.02.2022)	Verify Claims
0.9 (03.02.2022)	Summary and Recommendation
1.0 (03.02.2022)	Final document
1.1 (04.02.2022)	Added deployed contract addresses

2. About the Project and Company

KYC APPROVED

Website: <https://www.energyfi.io>

Twitter: https://twitter.com/Energyfi_io

Medium: <https://energyfi.medium.com>

Telegram: https://t.me/Energyfi_official



2.1 Project Overview

Energyfi is a large-scale effort focused at promoting Green Blockchain usage. Decentralized Finance has its mastodons (Uniswap, Pancakeswap, Aave, and Unicrypt for exemple), Energyfi is combining all these features in one place to offer a green All-in-One solution. The ultimate goal is to facilitate the adoption of green networks and the establishment of environmentally friendly decentralized financing. Energyfi is very useful for assisting customers in utilizing Green networks and establishing a sustainable Decentralized Finance system. Its purpose is remarkable as it Increases consciousness on the environment, empowers decentralization, enhances transparency and Interoperability between chains.

Energyfi covers important functionalities like:

- Cross-chain Launchpad.
- Decentralized Exchange.
- Staking/Farming.
- Lending/borrowing platform.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

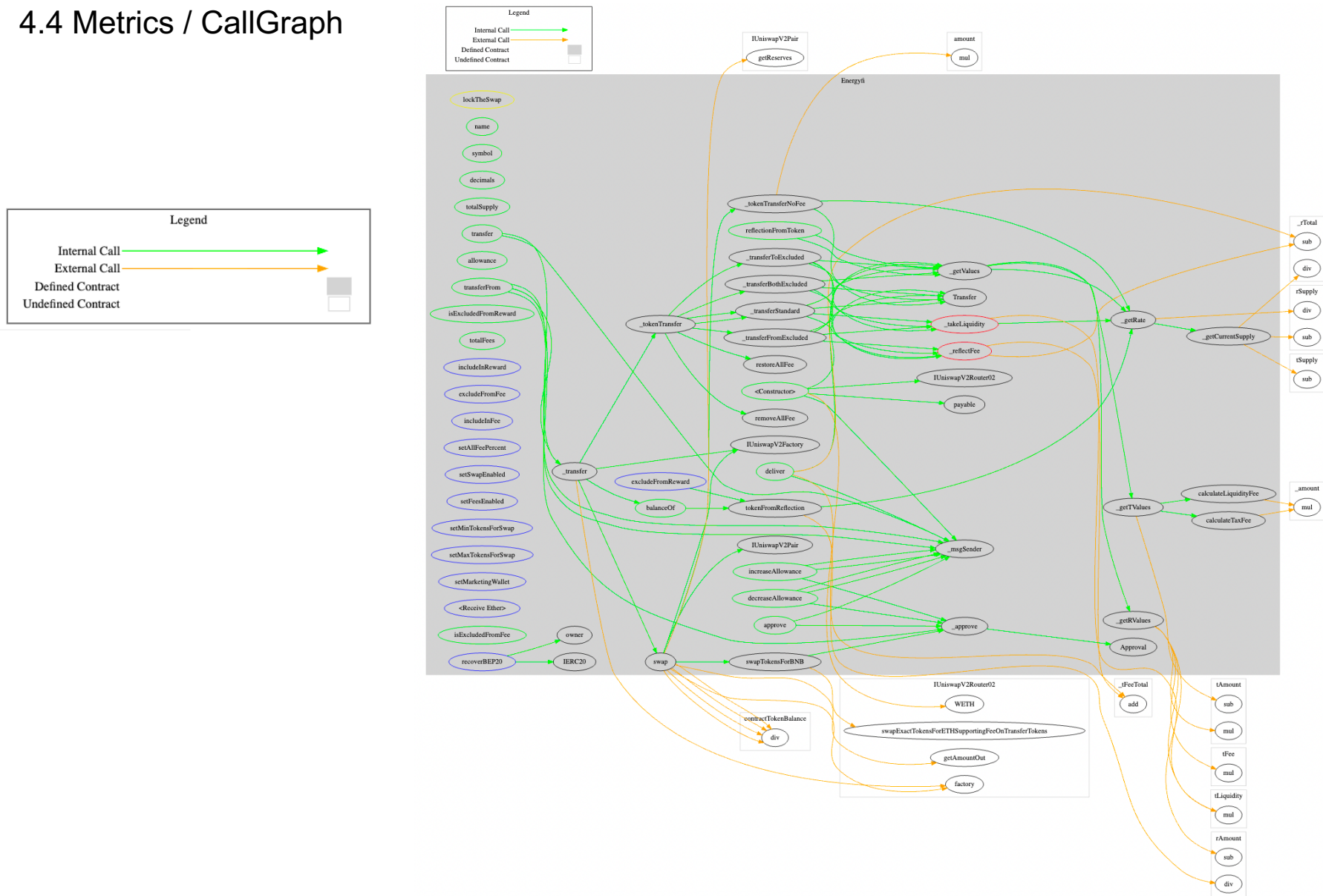
Dependency / Import Path	Source
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/access/Ownable.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/utils/Address.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/utils/Address.sol
@openzeppelin/contracts/utils/Context.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/utils/Context.sol
@openzeppelin/contracts/utils/math/SafeMath.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.4.2/contracts/utils/math/SafeMath.sol
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	https://github.com/Uniswap/v2-core/tree/v1.0.1/contracts/interfaces/IUniswapV2Factory.sol
@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol	https://github.com/Uniswap/v2-core/tree/v1.0.1/contracts/interfaces/IUniswapV2Pair.sol
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	https://github.com/Uniswap/v2-periphery/tree/v1.0.0-beta.0/contracts/interfaces/IUniswapV2Router02.sol

4.3 Tested Contract Files

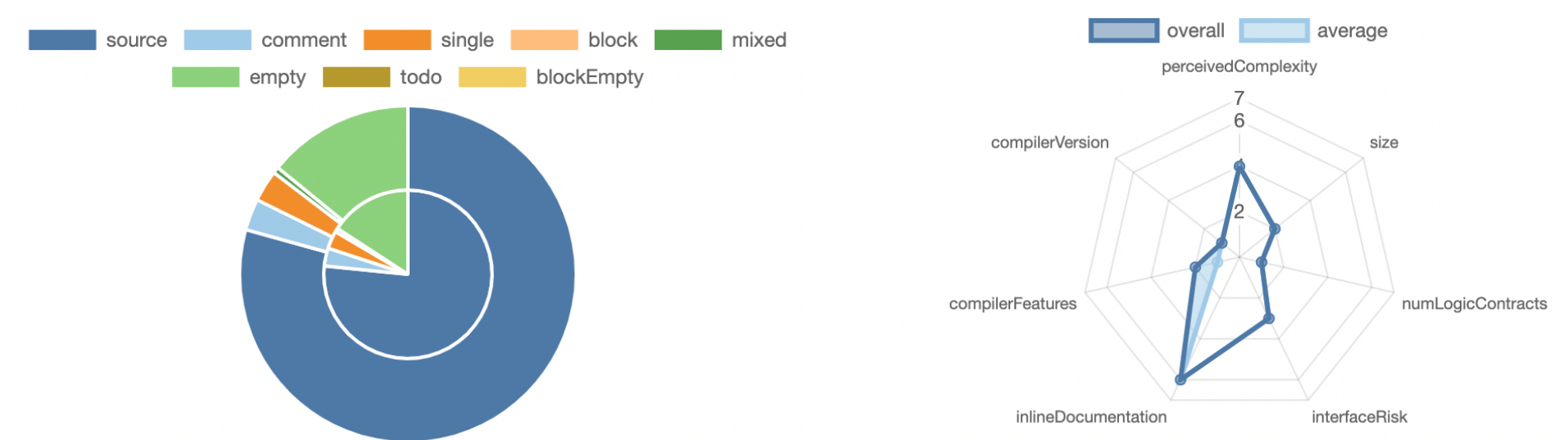
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
./Energyfi.sol	1c50f0f61ef1f931ef7538baf0105c30

4.4 Metrics / CallGraph



4.5 Metrics / Source Lines & Risk





4.6 Metrics / Capabilities


Solidity Versions observed		 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
<input type="text" value="0.8.6"/>		<input type="text"/>	<input type="text" value="yes"/>	<input type="text"/>	<input type="text"/>
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRrecover	 New/Create/Create2
<input type="text" value="yes"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Exposed Functions





This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable				
29	1				
External	Internal	Private	Pure	View	
12	32	21	1	17	

StateVariables

Total	 Public
31	16

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complexity Score	Capabilities
	contracts/Energyfi.sol	1		633	553	445	20	345	
	Totals	1		633	553	445	20	345	

Legend:

- **Lines:** total lines of the source unit
- **nLines:** normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC:** normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines:** lines containing single or block comments
- **Complexity Score:** a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

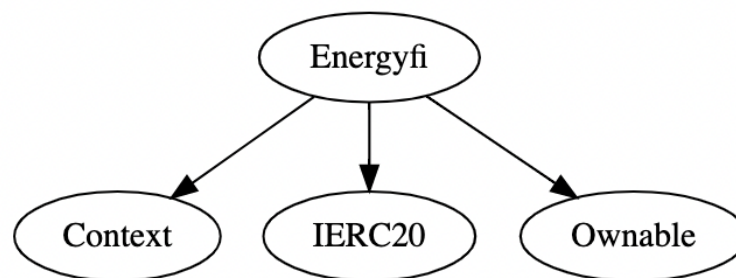
5. Scope of Work

The Energyfi Team provided us with the files that needs to be tested. The scope of the audit is the token contract.

The team put forward the following assumptions regarding the security, usage of the contracts:

- The smart contract is coded according to the newest standards and in a secure way

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract

LOW ISSUES

During the audit, Chainsulting's experts found **no Low issues** in the code of the smart contract

INFORMATIONAL ISSUES

5.1.1 Missing natspec documentation

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: Energyfi.sol

Attack / Description	Code Snippet	Result/Recommendation
Solidity contracts can use a special form of comments to provide rich documentation for function, return variables, and more. This special form is	//...	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract.

named Ethereum Natural Language Specification Format (NatSpec).		
-----------------------------------------------------------------	--	--

5.2 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✓
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓


ID	Title	Relationships	Test Result
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓

ID	Title	Relationships	Test Result
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓

ID	Title	Relationships	Test Result
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✓
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

5.3 Verify Claims

5.3.1. The smart contract is coded according to the newest standards and in a secure way

Status: tested and verified 

6. Executive Summary

Our Chainsulting expert performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the February 02, 2022.

The main goal of the audit was to verify the claims regarding the security of the smart contract. During the audit, no critical issues were found, after the manual and automated security testing and the claim have been successfully verified.

7. Deployed Smart Contract

VERIFIED

<https://bscscan.com/address/0xAe98E63dB1c4646BF5b40B29c664Bc922f71Bc65#code>