

# Smart Contract Security Audit

**AUDIT RATE TECH**

**for**

**CALORIE**



**Audit Rate  
Tech**



## ***Disclaimer***

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and AUDIT RATE TECH and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (AUDIT RATE TECH) owe no duty of care towards you or any other person, nor does AUDIT RATE TECH make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and AUDIT RATE TECH hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, AUDIT RATE TECH hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against AUDIT RATE TECH, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

## ***Audit details:***

Audited project: CALORIE

Total supply: 10,000,000

Token ticker: KCAL

Decimals: 18

Contract address: 0x6551bC10B4c5CCd6bDB78b0Db69a73cb2B94b8D0

Languages: Solidity (Smart contract)

Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Mythril,

Contract Library

Compiler Version: v0.8.7+commit.e28d00a7

Optimization Enabled: No with 200 runs

Contract Deployer Address: 0xF1656A6cc47C6427e2160Acd739A559023F434EB

Blockchain: Binance Smart Chain

Project website: <http://runandburn.app/>

The audit items and results:

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

Audit Result: **Not passed (PROXY)**

Audit Date: July 04, 2022

Audit Team: AUDIT RATE TECH

<https://www.auditrate.tech/>

## ***Introduction***

This Audit Report mainly focuses on the overall security of CALORIE Smart Contract. With this report, we have tried to ensure the reliability and correctness of their smart contract by complete and rigorous assessment of their system's architecture and the smart contract codebase.

### ***Auditing Approach and Methodologies applied***

The AUDIT RATE TECH team has performed rigorous testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line by line inspection of the Smart Contract to find any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In the Unit testing Phase, we coded/conducted custom unit tests written for each function in the contract to verify that each function works as expected.

In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was tested in collaboration of our multiple team members and this included -

- Testing the functionality of the Smart Contract to determine proper logic has been followed throughout the whole process.
- Analyzing the complexity of the code in depth and detailed, manual review of the code, lineby-line.
- Deploying the code on testnet using multiple clients to run live tests.
- Analyzing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analyzing the security of the on-chain data.

## ***Audit Goals***

The focus of the audit was to verify that the Smart Contract System is secure, resilient and working according to the specifications. The audit activities can be grouped in the following three categories:

**Security**

Identifying security related issues within each contract and the system of contract.

**Sound Architecture**

Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

**Code Correctness and Quality**

A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

## ***Issue Categories***

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

### ***High level severity issues***

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

### ***Medium level severity issues***

Issues on this level could potentially bring problems and should eventually be fixed.

### ***Low level severity issues***

Issues on this level are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

### ***Manual Audit:***

For this section the code was tested/read line by line by our developers. We also used Remix IDE's JavaScript VM and Kovan networks to test the contract functionality.

### ***Automated Audit***

**Remix Compiler Warnings**

It throws warnings by Solidity's compiler. If it encounters any errors the contract cannot be compiled and deployed. No issues found.

# Issues Checking Status

| SWC ID  | Description   | Checking status |
|---------|---|-----------------|
| SWC-100 | Function Default Visibility                             | Passed          |
| SWC-101 | Integer Overflow and Underflow                          | Passed          |
| SWC-102 | Outdated Compiler Version                               | Passed          |
| SWC-103 | Floating Pragma   | Passed          |
| SWC-104 | Unchecked Call Return Value                             | Passed          |
| SWC-105 | Unprotected Ether Withdrawal                            | Passed          |
| SWC-106 | Unprotected SELFDESTRUCT Instruction                    | Passed          |
| SWC-107 | Reentrancy  | Passed          |
| SWC-108 | State Variable Default Visibility                       | Passed          |
| SWC-109 | Uninitialized Storage Pointer                           | Passed          |
| SWC-110 | Assert Violation  | Passed          |
| SWC-111 | Use of Deprecated Solidity Functions                    | Passed          |
| SWC-112 | Delegatecall to Untrusted Callee                        | Passed          |
| SWC-113 | DoS with Failed Call                                    | Passed          |
| SWC-114 | Transaction Order Dependence                            | Passed          |
| SWC-115 | Authorization through tx.origin                         | Passed          |
| SWC-116 | Block values as a proxy for time                        | HIGH            |
| SWC-117 | Signature Malleability                                  | Passed          |
| SWC-118 | Incorrect Constructor Name                              | Passed          |
| SWC-119 | Shadowing State Variables                               | Passed          |
| SWC-120 | Weak Sources of Randomness from Chain Attributes        | Passed          |
| SWC-121 | Missing Protection against Signature Replay Attacks     | Passed          |
| SWC-122 | Lack of Proper Signature Verification                   | Passed          |
| SWC-123 | Requirement Violation                                   | Passed          |
| SWC-124 | Write to Arbitrary Storage Location                     | Passed          |
| SWC-125 | Incorrect Inheritance Order                             | Passed          |
| SWC-126 | Insufficient Gas Griefing                               | Passed          |
| SWC-127 | Arbitrary Jump with Function Type Variable              | Passed          |
| SWC-128 | DoS With Block Gas Limit                                | Passed          |
| SWC-129 | Typographical Error                                     | Passed          |
| SWC-130 | Right-To-Left-Override control character (U+202E)       | Passed          |
| SWC-131 | Presence of unused variables                            | Passed          |
| SWC-132 | Unexpected Ether balance                                | Passed          |
| SWC-133 | Hash Collisions With Multiple Variable Length Arguments | Passed          |
| SWC-134 | Message call with hardcoded gas amount                  | Passed          |
| SWC-135 | Code With No Effects                                    | Passed          |
| SWC-136 | Unencrypted Private Data On-Chain                       | Passed          |

## ***Role address privileges***

1120 grantRole  
1133 revokeRole  
1402 setWhiteLists  
1412 removeWhiteLists  
1425 setAddressForBosses  
1438 setTokenForBosses  
1452 setFeeRate  
1465 setRoleSuperAdmin  
1504 sweepTokenForBosses  
1519 withDraw

## Top Token Holders

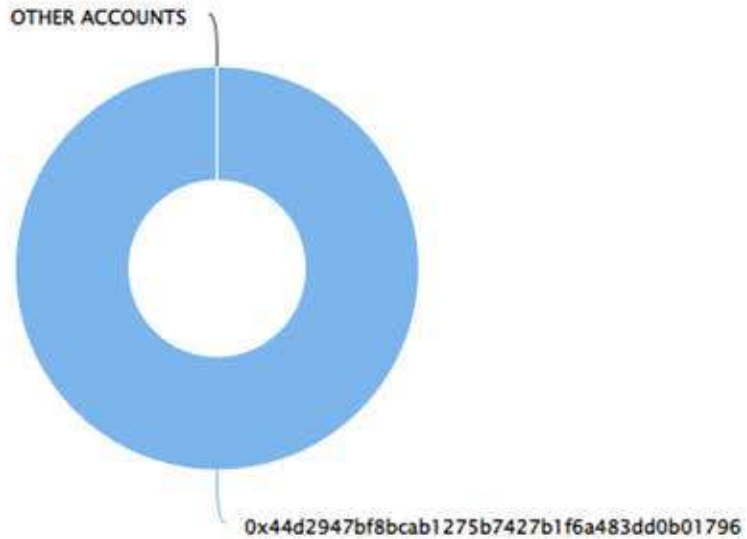
At the time of the audit

The top 3 holders collectively own 100.00% (10,000,000.00 Tokens) of CALORIE

Token Total Supply: 10,000,000.00 Token | Total Token Holders: 1

### CALORIE Top 3 Token Holders

Source: BscScan.com



(A total of 10,000,000.00 tokens held by the top 3 accounts from the total supply of 10,000,000.00 token)

| Rank | Address                                    | Quantity (Token) | Percentage |
|------|--|------------------|------------|
| 1    | 0x44d2947bf8bcab1275b7427b1f6a483dd0b01796 | 10,000,000       | 100.0000%  |

## KYC/Doxx

At the time of the audit, there is no information about the conduct of KYC / Doxx



## Conclusion

### HIGH RISK!

**The contract uses a proxy.**

**This carries a risk for investors, as the contract can be upgradeable and therefore is considered to be a high risk.**

**Recommendation:**

**Don't use proxies in contracts!**

**Owner can set fees up to 20% for sell and 20% for buy**

```
function setFeeRate(uint256 _sellFeeRate, uint256 _buyFeeRate)
    external onlyRole(FEE_CONTROL_ROLE) {
    require(_sellFeeRate > 0 && _sellFeeRate < 20, "sellFeeRate invalid");
    require(_buyFeeRate > 0 && _buyFeeRate < 20, "buyFeeRate invalid");
    sellFeeRate = _sellFeeRate;
    buyFeeRate = _buyFeeRate;
    emit UpdateTransferRate(sellFeeRate, buyFeeRate);}
```

**No mint function found**

**Owner cannot set max tx amount**

**Owner cannot pause trading**

### **Note:**

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner. The analysis of the contract does not give complete security and includes only the analysis that is indicated in the report. We do not analyze locked tokens or LP tokens, the presence of KYC in other companies, and so on. Also, our audit is not a recommendation for investment. All responsibility for the loss of investment lies with you!

