KISHIELD

Security Audit

Calories Token

April 29, 2022

Table of Contents

Audit Summary

This report has been prepared for Calories Token on the Binance Chain network. KISHIELD provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Ensuring contract logic meets the specifications and intentions of the client without exposing the user's funds to risk.
- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Project Overview

Token Summary

Parameter	Result
Address	0xD1A5498Bd3AC9878814Fff4115503c08f75c3853
Name	Calories
Token Tracker	Calories (Calories)
Decimals	9
Supply	100,000,000
Platform	Binance Chain
compiler	v0.8.2+commit.661d1103
Optimization	Yes with 200 runs
LicenseType	None
Language	Solidity
Codebase	https://bscscan.com/address/0xD1A5498Bd3AC9878814Fff4115503c08f75c3853
Url	https://calories.finance/

Main Contract Assessed

Name	Contract	Live
Calories	0xD1A5498Bd3AC9878814Fff4115503c08f75c3853	Yes

Smart Contract Vulnerability Checks

Vulnerability	Automatic Scan	Manual Scan	Result
Unencrypted Private Data On-Chain	Complete	Complete	Low / No Risk
Code With No Effects	Complete	Complete	Low / No Risk
Message call with hardcoded gas amount	Complete	Complete	Low / No Risk
Hash Collisions With Multiple Variable Length Arguments	Complete	Complete	Low / No Risk
Unexpected Ether balance	Complete	Complete	Low / No Risk
Presence of unused variables	Complete	Complete	Low / No Risk
Right-To-Left-Override control character (U+202E)	Complete	Complete	Low / No Risk
Typographical Error	Complete	Complete	Low / No Risk
DoS With Block Gas Limit	Complete	Complete	Low / No Risk
Arbitrary Jump with Function Type Variable	Complete	Complete	Low / No Risk
Insufficient Gas Griefing	Complete	Complete	Low / No Risk
Incorrect Inheritance Order	Complete	Complete	Low / No Risk
Write to Arbitrary Storage Location	Complete	Complete	Low / No Risk
Requirement Violation	Complete	Complete	Low / No Risk
Missing Protection against Signature Replay Attacks	Complete	Complete	Low / No Risk
Weak Sources of Randomness from Chain Attributes	Complete	Complete	Low / No Risk

Vulnerability	Automatic Scan	Manual Scan	Result
Authorization through tx.origin	Complete	Complete	Low / No Risk
Delegatecall to Untrusted Callee	Complete	Complete	Low / No Risk
Use of Deprecated Solidity Functions	Complete	Complete	Low / No Risk
Assert Violation	Complete	Complete	Low / No Risk
Reentrancy	Complete	Complete	Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	Low / No Risk
Unprotected Ether Withdrawal	Complete	Complete	Low / No Risk
Unchecked Call Return Value	Complete	Complete	Low / No Risk
Outdated Compiler Version	Complete	Complete	Low / No Risk
Integer Overflow and Underflow	Complete	Complete	Low / No Risk
Function Default Visibility	Complete	Complete	Low / No Risk

Contract Ownership

The contract ownership of Calories is not currently renounced. The ownership of the contract grants special powers to the protocol creators, making them the sole addresses that can call sensible ownable functions that may alter the state of the protocol.

The current owner is the address 0xE5d66B4C93028fE3F1AB08Dea295fDd669600D56 which can be viewed from:
HERE

The owner wallet has the power to call the functions displayed on the priviliged functions chart below, if the owner wallet is compromised this privileges could be exploited.

We recommend the team to renounce ownership at the right timing if possible, or gradually migrate to a timelock with governing functionalities in respect of transparency and safety considerations.

Important Notes To The Users:

- The owner cannot mint tokens after intial deployment.
- The owner cannot stop Trading.
- The owner cannot change the min tx amount.
- The owner cannot change tax fee amounts.
- If a user gets added to the bot address blacklist, the antibot mechanism will only trigger if the tx amount is less than the antiBotAmount and the time of the tx is less than the antiBotTime.
- The owner can add/remove WALLETS and contracts addresses to the bot blacklist.
- The owner can withdraw all the tokens of the contract to an external address using the function SetMintContract().
- The owner can include/exclude addresses from fees.
- No high-risk Exploits/Vulnerabilities Were Found in token Source Code.

Audit Passed

Findings Summary

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited.Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Info	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

Findings

Severity	Found
High	0
Medium	1
Low	1
Info	4
Total	6

Findings

Arithmetic error

ID	Severity	Contract	Function
01	Medium	Calories	Function _transfer()

Description

When users buy tokens (sender == uniswapV2Pair) fee is calculated using buyFeeRate (8%) when it comes to the distribution of this fee to the mintContract and addressForMarketing only 70% of that 8% is ditributed. When users sell tokens (recipient == uniswapV2Pair) fee is calculated using sellFeeRate (8%) when it comes to the distribution of this fee to the mintContract, addressForMarketing, and BurnAddr only 80% of that 8% is ditributed.

Recommendation

Notice that the divisions are made on the result of the 8% of the amount, and make the required changes.

Unclear Logic

ID	Severity	Contract	Function
02	Informational	Calories	Function _transfer()

Description

`_mint(mintContract, tokensForRewards);` tokensForRewards is MAX_TOKENS_FOR_REWARDS which is 0.

Recommendation

We recommend deleting code that makes no changes.

Unclear Logic

ID	Severity	Contract	Function
03	Informational	Calories	Function _transfer()

Description

`unlockTime > block.timestamp` when the unlock time is not over, there is a set of the variable 'values' (which is not used by the contract logic) in similar fashion the comparison `values>MAX_LOCK` is unclear as MAX_LOCK variable is not used by the contract logic.

Recommendation

We recommend deleting code that make changes to unused variables.

Variables could be declared as constant

ID	Severity	Contract	Function
04	Informational	Calories	variables BurnAddr, MAX_LOCK, MAX_TOKENS_FOR_REWARDS, MAX_TOTAL_SUPPLY, antiBotDuration, buyFeeRate, sellFeeRate

Description

Gas Optimization. Variables that are never changed could be declared as constant.

Recommendation

We recommend declaring those variables as constant.

Public function that could be declared external

ID	Severity	Contract	Function
05	Informational	Calories	Functions includeInFee, excludeFromFee, getvalue, getUnlockTime, burn, burnFrom

Description

Gas Optimization. Public function that could be declared external

Recommendation

Public functions that are never called by the contract should be declared external to save gas.

Division before Multiplication

ID	Severity	Contract	Function
06	Low	Calories	function _transfer()

Description

Precision Loss. '_fee = amount.mul(sellFeeRate).div(100) => super._transfer(sender,mintContract,_fee.mul(10).div(100))'. Division before multiplication can result in truncation and less accurate results

Recommendation

Multiplication should be performed before division to not lose precision.

Priviliged Functions (onlyOwner)

Function Name	Parameters	Visibility
renounceOwnership	none	public
transferOwnership	address newOwner	public
getUnlockTime	none	public
setUnlockTime	none	external
setBotAddresses	calldata _addresses	external
addBotAddress	address _address	external
antiBot	uint256 amount	external
getvalue	none	public
excludeFromFee	address account	public
includeInFee	address account	public
SetMintContract	address contrat_addr	external

Statistics

Liquidity Info

Parameter	Result
Pair Address	0x971afD0F3b2b97C85D85494F6Ccd5bC27A0FFe88
Calories Reserves	0.00 Calories
BNB Reserves	0.00 BNB
Liquidity Value	$0 USD

Token (Calories) Holders Info

Parameter	Result
Calories Percentage Burnt	0.00%
Calories Amount Burnt	0 Calories
Top 10 Percentage Own	100.00%
Top 10 Amount Owned	100,000,000 Calories
Top 10 Aprox Value	$NaN USD

LP (Calories/BNB) Holders Info

Parameter	Result
Calories/BNB % Burnt	0.00%
Calories/BNB Amount Burnt	0 Calories
Top 10 Percentage Owned	0.00%
Top 10 Amount Owned	0 Calories
Locked Tokens Percentage	0.00%
Locked Tokens Amount	0 Calories

* All the data diplayed above was taken on-chain at block 17383619
* The tokens on industry-standard burn wallets are not included on the top 10 wallets calculations

Liquidity Ownership

The token does not have liquidity at the moment of the audit, block 17383619

KISHIELD

Disclaimer

KISHIELD has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided ‘as is’, and KISHIELD is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will KISHIELD or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

The assessment services provided by KISHIELD is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.