KISHIELD

Security Audit

Sesame Token & Game

March 19, 2022

Table of Contents

Audit Summary

This report has been prepared for Sesame Token & Game 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	0x1E4d02b43FA67F8A0fec6632F8c636D45A527d4B
Name	Sesame
Token Tracker	Sesame (SESA)
Decimals	18
Supply	100 million
Platform	Binance Chain
compiler	v0.8.7+commit.e28d00a7
Optimization	Yes with 1 runs
LicenseType	MIT
Language	Solidity
Codebase	https://github.com/Sesame-Blockchain/Sesame
Url	sesame.io

Main Contract Assessed

Name	Contract	Live
SesameGame.sol	---	No

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	✓ Fixed
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 Sesame Protocol does not exist, instead it uses access control to set up roles

The current MINTER_ROLE is the address 0x31c7a216c7d3a6b9c49c3d04eea315c0bc470662 which can be viewed from: HERE

The current DEFAULT_ADMIN_ROLE is the address 0x31C7A216c7D3a6B9c49c3d04Eea315C0BC470662 which can be viewed from: HERE

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

Important Notes To The Users:

- The owner cannot mint more tokens than the supply cap (100 million)
- All the issues in the finding summary have been fixed.
- The owner cannot stop Trading.
- The token does not have buy/sell taxes.
- There is a 5% tax in the lottery.
- The transfer function is implemented correctly.
- The owner cannot change the max tx amount.
- The contract complies with the BEP-20 token standard.
- No high-risk Exploits/Vulnerabilities Were Found in token Source Code that puts users in danger.

Audit Passed

Findings Summary

Classification of Issues

All Issues have been resolved by the team.

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	1
🟠 Medium	2
🟡 Low	1
🟣 Info	1
Total	5
Fixed	5

Findings

Reentrancy

ID	Severity	Contract	Function
01	🟠 Medium	SesameGame	function _refund(uint256) (contracts/ SesameGame.sol#199-216)

Description

Possible reentrancy by making state variables changes after sending ether. Variable ticketMap[_round][player] is updated after send transaction, given that _refund is controlled by governance the risk is mitigated

Recommendation

We advise making use of the OpenZeppelin ReentrancyGuard.sol modifier, make use of the check-effects-interactions pattern.

Reentrancy

ID	Severity	Contract	Function
02	🔴 High	SesameGame	function enter(uint256) (contracts/ SesameGame.sol#98-140)

Description

Possible reentrancy by making state variables changes after sending ether. If the user buys a ticket and the count goes over ticketPerRound, their money is refunded but by making the ticket substraction after the ether send transaction an exploiter can call enter() once more and be sent the extra amount. This can go on until the balance of the contract reaches 0.

Recommendation

We advise of modifing the logic so that any transaction that tries to buy a ticket when the amount of tickets is equal to ticketPerRound fails or making use of the OpenZeppelin ReentrancyGuard.sol modifier, make use of the check-effects-interactions pattern

Reentrancy

ID	Severity	Contract	Function
03	🟠 Medium	SesameGame	function pickWinner(uint256[],uint256) (contracts/ SesameGame.sol#146-179)

Description

Possible reentrancy by making state variables changes after sending ether. _closeRound(_round) is called after send ETH transaction, given that _refund is controlled by onlyRNG the risk is mitigated

Recommendation

We advise making use of the OpenZeppelin ReentrancyGuard.sol modifier, make use of the check-effects-interactions pattern.

Uninitialized local variables

ID	Severity	Contract	Function
04	🟡 Low	SesameGame	function enter(uint256) and _refund(uint256)

Description

Variables'i' in for-loop are uninitialized

Recommendation

Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

Public function that could be declared external

ID	Severity	Contract	Function
05	● Informational	SesameGame	Functions getTicketCount() and getUserTicketCount(uint256,address)

Description

Public function that could be declared external

Recommendation

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

Priviliged Functions (onlyGovernance && onlyRNG)

Function Name	Parameters	Visibility
pickWinner	none	external
activate	none	external
deactivate	none	external

Statistics

Liquidity Info

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

Token (SESA) Holders Info

Parameter	Result
SESA Percentage Burnt	0.00%
SESA Amount Burnt	0 SESA
Top 10 Percentage Own	0.00%
Top 10 Amount Owned	0 SESA
Top 10 Aprox Value	$0 USD

LP (SESA/BNB) Holders Info

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

* All the data diplayed above was taken on-chain at block 16199467
* 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 16199467

KISHIΞLD

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.