



Smart Contract Security Audit Report

Spring game

July 2022

Security Status



www.hacksafe.io



Audit Details



Audited project

Spring game



Deployer address

0x17bcd0515094182cff084471D5beb2bA08Bac327



Client contacts

Spring game team



Blockchain

Binance Smart Chain



Website

<https://spring.game/>

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 TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (HackSafe) owe no duty of care towards you or any other person, nor does HackSafe 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 HackSafe 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, HackSafe hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HackSafe, 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.

Procedure

Step 1 - In-Depth Manual Review

Manual line-by-line code reviews to ensure the logic behind each function is sound and safe from various attack vectors. This is the most important and lengthy portion of the audit process (as automated tools often cannot find the nuances that lead to exploits such as flash loan attacks).

Step 2 - Automated Testing

Simulation of a variety of interactions with your Smart Contract on a test blockchain leveraging a combination of automated test tools and manual testing to determine if any security vulnerabilities exist.

Step 3 – Leadership Review

The engineers assigned to the audit will schedule meetings with our leadership team to review the contracts, any comments or findings, and ask questions to further apply adversarial thinking to discuss less common attack vectors.

Step 4 - Resolution of Issues

Consulting with the team to provide our recommendations to ensure the code's security and optimize its gas efficiency, if possible. We assist project team's in resolving any outstanding issues or implementing our recommendations.

Step 5 - Published Audit Report

Boiling down results and findings into an easy-to-read report tailored to the project. Our audit reports highlight resolved issues and any risks that exist to the project or its users, along with any remaining suggested remediation measures. Diagrams are included at the end of each report to help users understand the interactions which occur within the project.

Background

HackSafe was commissioned by Spring game to perform an audit of smart contracts:

- <https://bscscan.com/address/0xA29fbC10bF1c2E7B108E8704732F4bB0B10C0dA5#code>

The purpose of the audit was to achieve the

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be understood to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contract Details

Token contract details for 06.07.2022

Token Type	: BEP20
Contract name	: Spring
Contract address	: 0xA29fbC10bF1c2E7B108E8704732F4bB0B10C0dA5
Compiler version	: v0.8.9+commit.e5eed63a
Total supply	: 10,000,000
Token Ticker	: SPR
Decimals	: 18
Token Holders	: 260
Top 100 token holder's dominance	: 99.99%
Transactions count	: 5,400
Contract deployer address	: 0x17bcd0515094182cff084471D5beb2bA08Bac327
Owner address	: 0x17bcd0515094182cff084471D5beb2bA08Bac327
Contract owners length	: 1

Social profiles

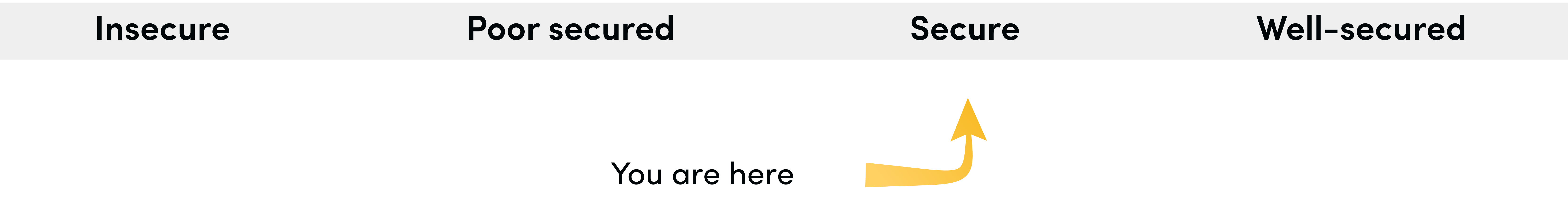
Twitter Profile	: https://twitter.com/spring_gamefi
Whitepaper link	: https://docs.spring.game/spring-game/introduction
Telegram Profile	: https://t.me/springgame
Coinmarketcap profile	: https://coinmarketcap.com/currencies/spring-game/

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
<p>Tokenomics :</p> <ul style="list-style-type: none">• Name : Spring• Symbol : SPRING• Decimals : 18• Protocol : BEP20• Total supply : 10,000,000• Contract address :0xA29fbC10bF1c2E7B108E8704732F4bB0B10C0dA5	<p>Yes, This is valid.</p>

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are “Secure”. This token contract does contain owner control, which do not make it fully decentralized as owner does have control over smart contract.



We used various tools like Slither, Mythril and Remix IDE. At the same time this finding is based on critical analysis of the manual audit. All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the issues checking status.

We found 0 critical, 0 high, 1 medium and 1 low and some very low-level issues. These issues are not critical ones.

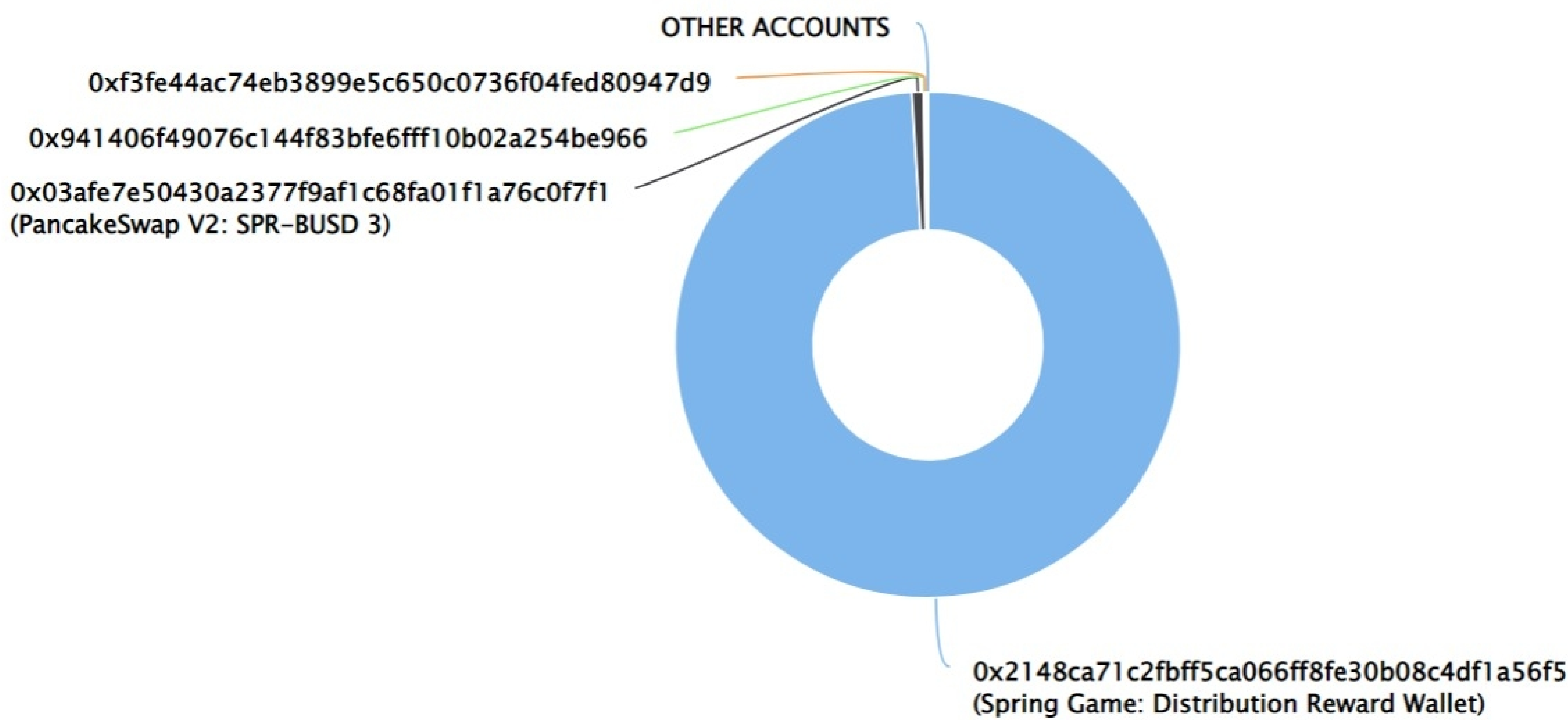
Spring Token Distribution

💡 The top 100 holders collectively own 99.99% (9,998,967.82 Tokens) of Spring

💡 Token Total Supply: 10,000,000.00 Token | Total Token Holders: 257

Spring Top 100 Token Holders

Source: BscScan.com



Spring Token Distribution

Spring Token Top 20 Token Holders

A total of 255 token holders



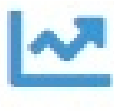




















First

<

Page 1 of 6

>

Last

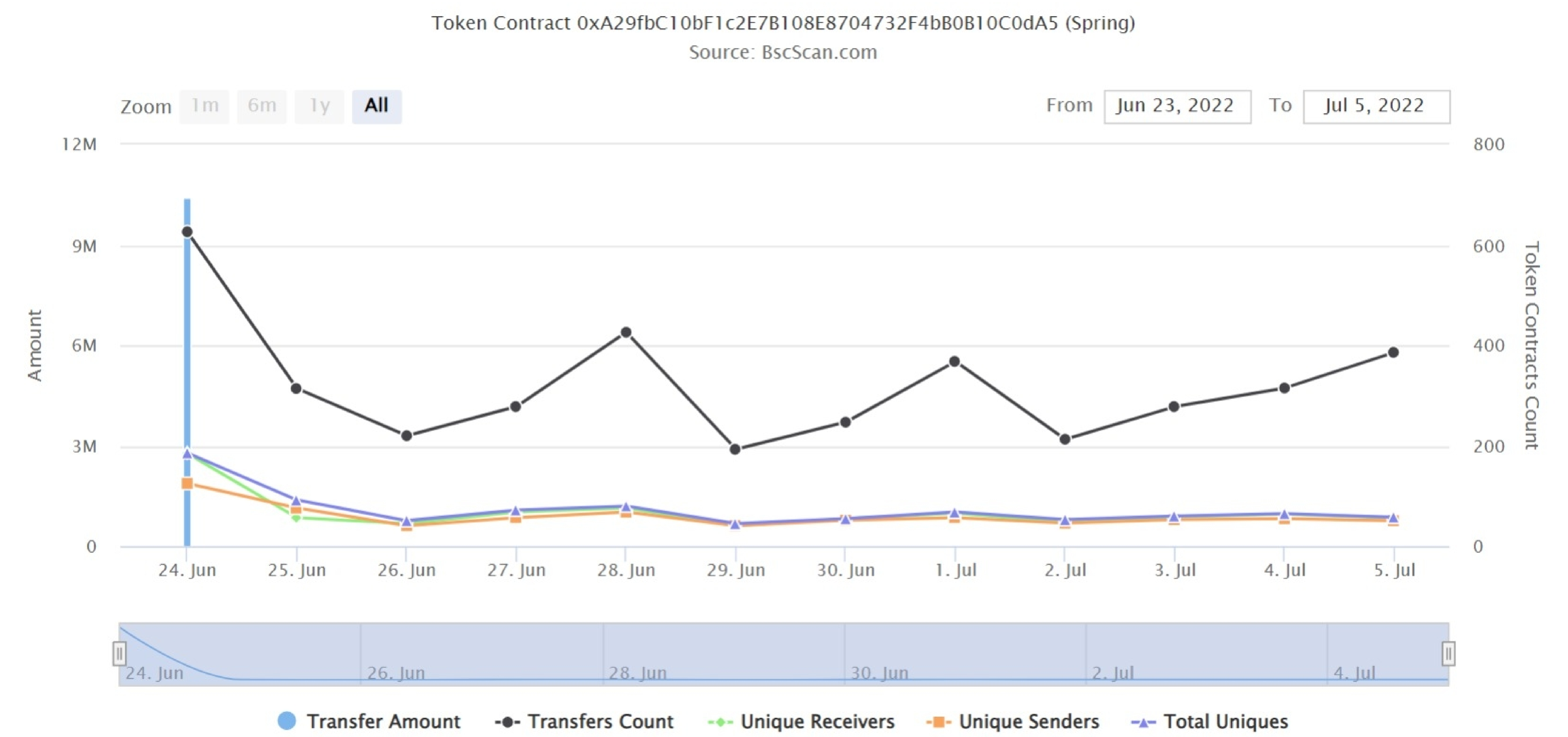
Rank	Address	Quantity	Percentage	Analytics
1	Spring Game: Distribution Reward Wallet	9,898,358.310275092591743462	98.9836%	
2	 PancakeSwap V2: SPR-BUSD 3	76,856.104972801281291814	0.7686%	
3	0x941406f49076c144f83bfe6fff10b02a254be966	6,512.958474629146192137	0.0651%	
4	0xf3fe44ac74eb3899e5c650c0736f04fed80947d9	1,019.772583000988025023	0.0102%	
5	0xbe3efab08a8239551a20ba8c6b2b7a23e44834d9	976.051239770057615283	0.0098%	
6	 0x1942b7ac6b03f3c288b8f5acfff9e6effb1d99aa	776.005073449660455969	0.0078%	
7	0x4bd76c534fe89f2892860a97923031a580a73425	723.518966228306352021	0.0072%	
8	0xea19625d1e68dc41822c4c815c892baa599f75c8	712.968597749240434957	0.0071%	
9	0xd9605ec7b483ad75b7e4b9f2e9300eda8826d09e	582.903918935994861347	0.0058%	
10	0x12a5918386317058a47ffb2f3d1c05b23beaceb8	572.097840686645037756	0.0057%	
11	0x89f8cdf54c35ac80ed10cac4a23c61ee14230377	538.667848232376242224	0.0054%	
12	0xd97cbc25ebac856cae70d4b62ae8911087235b49	470.763130363533221815	0.0047%	
13	0x9d58b5bad8cf10f9766b3efa81cd2742c4875520	470	0.0047%	
14	0x408329a24c6a3ae7dbab6e12237e921b1cb9450f	411.792076384315818894	0.0041%	
15	0x12d2ed4b0dc8cee49a5760b0cca801e6632d1224	369.568577895997653332	0.0037%	
16	0x892256f9f90b550a0742c8ec82d9ec3cb2c73bb7	359.717641830473957324	0.0036%	
17	0x96a8df8d1dbe6879f16818c5dcc72c4572095f5c	356.776686876323599277	0.0036%	
18	0x817e23e314d11a2902b3aaa5051eba5810daf9d6	328.93580901618922449	0.0033%	
19	0xc4858770895462814143943ca912af74aa78689b	301.4493266233128342	0.0030%	
20	 0x733bfb50b726d58be72f01d347b7d4ee2420de1b	287.020444243671701927	0.0029%	

Spring Token Distribution

Spring Token Contract Overview

Time Series: Token Contract Overview

Fri 24, Jun 2022 - Tue 5, Jul 2022



Contract functions details

ERC20.sol

+ ERC20 (Context, IERC20, IERC20Metadata)

- < constructor> #
- [Pub] name
- [Pub] symbol
- [Pub] decimals
- [Pub] totalSupply
- [Pub] balanceOf
- [Pub] transfer #
- [Pub] allowance
- [Pub] approve #
- [Pub] transferFrom #
- [Pub] increaseAllowance
- [Pub] decreaseAllowance
- [Int] _transfer #
- [Int] _mint #
- [Int] _burn #
- [Int] _approve #
- [Int] _spendAllowance #
- [Int] _beforeTokenTransfer
- [Int] _afterTokenTransfer

IERC20.sol

+ [Int] IERC20

- [Ext] totalSupply
- [Ext] balanceOf
- [Ext] transfer
- [Ext] allowance
- [Ext] approve
- [Ext] transferFrom

IERC20Metadata.sol

+ [Int] IERC20Metadata (IERC20)

- [Ext] name
- [Ext] symbol
- [Ext] decimals

Context.sol

+ Context

- [Int] _msgSender
- [Int] _msgData

Contract functions details

Owners.sol

+ Owners

- <constructor> #
- [Ext] addOwner
 - modifiers: onlySuperOwner
- [Ext] removeOwner
 - modifiers: onlySuperOwner
- [Ext] getOwnersSize

Spring.sol

+Spring (ERC20, Owners)

- <constructor>
- [Ext] mint
 - modifiers: onlyOwners
- [Ext] setOpenToTrading
 - modifiers: onlyOwners
- [Ext] setBlacklist
 - modifiers: onlyOwners
- [Ext] setWhitelist
 - modifiers: onlyOwners
- [Int] _beforeTokenTransfer

(\$) = payable function

= non-constant function

Issues Checking Status

No.	Title	Status
1.	Unlocked Compiler Version	Low issue
2.	Missing Input Validation	Passed
3.	Race conditions and Reentrancy. Cross-function race conditions.	Passed
4.	Possible delays in data delivery	Passed
5.	Oracle calls.	Passed
6.	Timestamp dependence.	Passed
7.	Integer Overflow and Underflow	Passed
8.	DoS with Revert.	Passed
9.	DoS with block gas limit.	Passed
10.	Methods execution permissions.	Passed
11.	Economy model of the contract.	Passed
12.	Private use data leaks.	Passed
13.	Malicious Event log.	Passed
14.	Scoping and Declarations.	Passed
15.	Uninitialized storage pointers.	Passed
16.	Arithmetic accuracy.	Passed
17.	Design Logic.	Medium issue
18.	Safe Open Zeppelin contracts implementation and usage.	Passed
19.	Incorrect Naming State Variable	Passed
20.	Too old version	Passed

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.

Security Issues

✔ Critical Severity Issues

No critical severity issue found.

✔ High Severity Issues

No high severity issue found.

✔ Medium Severity Issues

One medium severity issues found.

1. Design logic

- **Description**

The Spring.sol file contains direct import of openzeppelin ERC20 smart contract, any changes in their repository can create trouble for this token smart contract as main smart contract file is inherited from that.

- **Recommendation**

It advisable to not direct import any file from github repository, contract have already declared the ERC20 contract and their interfaces so contract do not need to import the openzeppelin ERC20 contract.

✔ Low Severity Issues

One low severity issue found.

1. Unlocked Compiler Version.

- **Description**

The contract utilizes an unlocked compiler version. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

- **Recommendation**

It is advisable that the compiler version is alternatively locked at the lowest version possible so that the contract can be compiled. For example, for version ^0.8.0 the contract should contain the following line:

```
pragma solidity 0.8.9;
```

Centralization

Owner Privileges :

- Spring Token Contract:
 - Owner can add and remove owner.
 - Owner can mint tokens.
 - Owner can set contract for open or close for trading.
 - Owner can add blacklisted and whitelisted addresses in contract.

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble as smart contract ownership cannot be renounced. Following are Admin functions:

- Addowner
- Removeowner
- Mint
- Setopentotrading
- Setblacklist
- Setwhitelist

Conclusion

Smart contract contains low and medium severity issues! The further transfer and operations with the fund raised are not related to this particular contract.

HackSafe 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.