HackSafe

Smart Contract Security Audit Report

Rasta Token

November 2022

Security Status

★★★★⯪

www.hacksafe.io

Audit Details

Audited project
Rasta Token

Deployer address
0xA0AC1B72b7b5BC604f1EaA57B80D54FFB51b7E68

Client contacts
Rasta Token Team

Blockchain
Binance smart chain

Website
https://zionlabs.info/

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 Rasta Token to perform an audit of smart contracts:

- https://bscscan.com/token/0xE3e8cC42DA487d1116D26687856e9FB684817c52#code

The purpose of the audit was to achieve the following:

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

The information in this report should be 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 11.11.2022

Token Type	: UTILITY
Contract name	: RASTAToken
Contract address	: 0xE3e8cC42DA487d1116D26687856e9FB684817c52
Total supply	: 4,009,749.865068
Token Ticker	: RASTA
Decimals	: 18
Token Holders	: 3,659
Transactions count	: 290,471
Compiler version	: v0.6.12+commit.27d51765
Contract deployer address	: 0xA0AC1B72b7b5BC604f1EaA57B80D54FFB51b7E68
Owner address	: 0xec89be665c851ffbae2a8ded03080f3e64116539

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.

Insecure	Poor secured	Secure	Well-secured

You are here

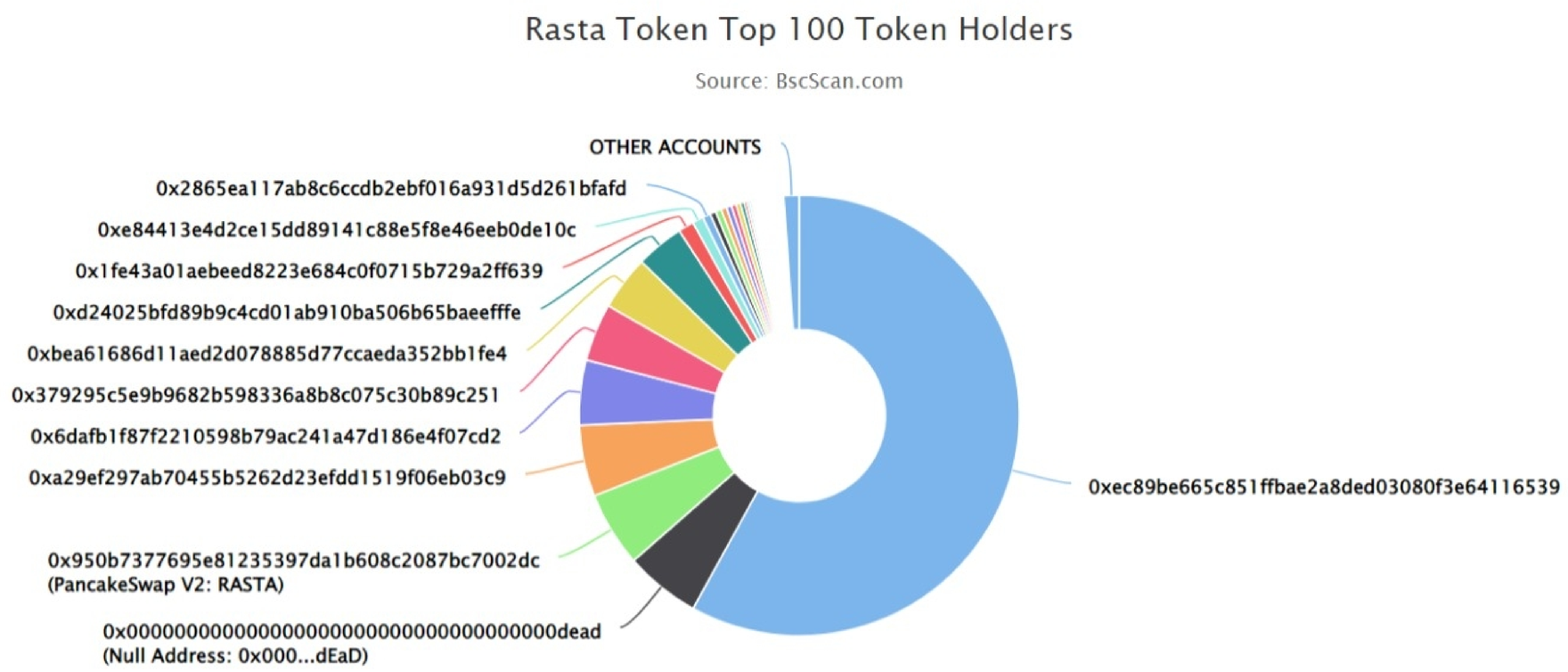
We used various tools like Slither, Mythril and Remix IDE. At the same time this finding is based on critical analysis of the manual audit. All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the issues checking status.

We found 0 critical, 0 high, 0 medium and 2 low.

Rasta Token Distribution

The top 100 holders collectively own 98.87% (3,964,438.54 Tokens) of Rasta Token

Token Total Supply: 4,009,749.87 Token | Total Token Holders: 3,659



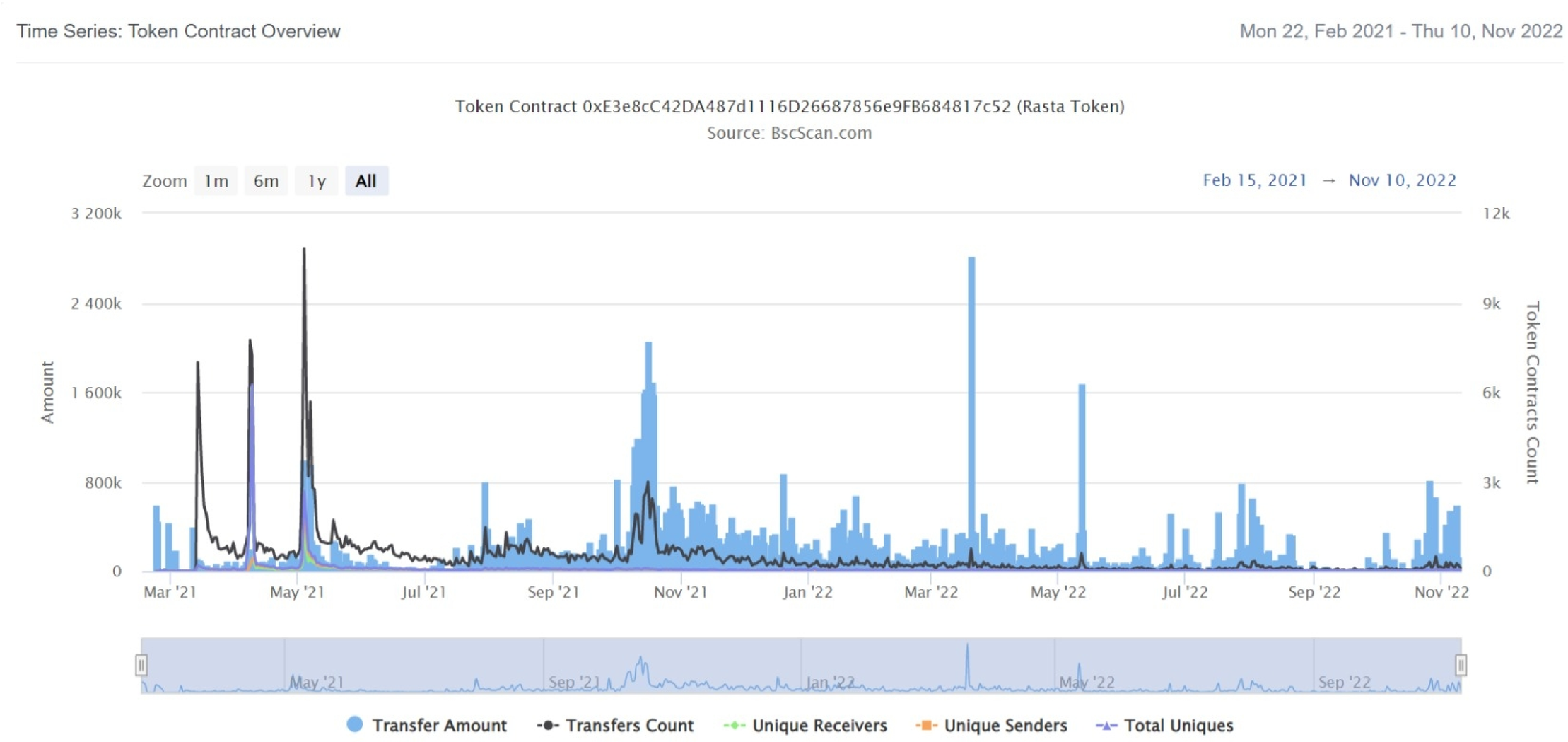
Rasta Token Top 20 Token Holders

(A total of 3,964,438.54 tokens held by the top 100 accounts from the total supply of 4,009,749.87 token)

Rank	Address	Quantity (Token)	Percentage
1	0xec89be665c851ffbae2a8ded03080f3e64116539	2,326,525.192654364778846571	58.0217%
2	Null Address: 0x000...dEaD	222,420	5.5470%
3	PancakeSwap V2: RASTA	220,887.91751539798598089	5.5088%
4	0xa29ef297ab70455b5262d23efdd1519f06eb03c9	208,917.855710825458886497	5.2102%
5	0x6dafb1f87f2210598b79ac241a47d186e4f07cd2	191,424.274832274673980557	4.7740%
6	0x379295c5e9b9682b598336a8b8c075c30b89c251	170,073.930049570249359067	4.2415%
7	0xbea61686d11aed2d078885d77ccaeda352bb1fe4	159,966.514466948289489699	3.9894%
8	0xd24025bfd89b9c4cd01ab910ba506b65baeefffe	141,654.948984909132594711	3.5328%
9	0x1fe43a01aebeed8223e684c0f0715b729a2ff639	46,059.33	1.1487%
10	0xe84413e4d2ce15dd89141c88e5f8e46eeb0de10c	32,299.308775057471922033	0.8055%
11	0x2865ea117ab8c6ccdb2ebf016a931d5d261bfafd	23,783.927356145770024453	0.5932%
12	0xe6e26a17dc579570686b3158326fd29752aedc80	17,250.664572676232063588	0.4302%
13	0xa62c4ed3fb5b6808e82482cceaa2fe0090a91113	17,170.500078096713785471	0.4282%
14	0x7e9378dcf1179ff217219859e86e75b6d08f0703	15,963.491510582974341752	0.3981%
15	0x62219ffd2b8ab2b21fd9eb9faad7244a9004d490	15,330.098438420305738702	0.3823%
16	0x32a556347711e2cc4c989f63363654345aa21a9d	14,127.780560200061178688	0.3523%
17	0x3f5c3f8f19f072b678ad6c6910d824d384b634bc	13,258.059136334940642932	0.3306%
18	0x75557f2684d87382ea20859b610cb20a74111b1d	11,845.461912087142745996	0.2954%
19	0x4ae240891ecfd36ad459e76ff0e6a7e07a766844	8,286.779443902574544678	0.2067%
20	0x05cd67ebcbe2ee82b0a9b031349ae7d0976cb907	7,201.480603785052904047	0.1796%

Rasta Token Distribution

Rasta Token Contract Overview



Contract functions details

```
+Context
    -[Int] _msgSender
    -[Int] _msgData

+Ownable (Context)
    -[Int] <constructor>
    -[Pub] owner
    -[Pub] renounceOwnership #
      -modifiers: onlyOwner
    -[Pub] transferOwnership #
      -modifiers: onlyOwner

+[Int] IERC20
    -[Ext] totalSupply
    -[Ext] balanceOf
    -[Ext] transfer
    -[Ext] allowance
    -[Ext] approve
    -[Ext] transferFrom

+[Lib] SafeMath
    -[Int] add
    -[Int] sub
    -[Int] sub
    -[Int] mul
    -[Int] div
    -[Int] div
    -[Int] mod
    -[Int] mod

+[Lib] Address
    -[Int] isContract
    -[Int] sendValue
    -[Int] functionCall
    -[Int] functionCall
    -[Int] functionCallWithValue
    -[Int] functionCallWithValue
    -[Int] functionStaticCall
    -[Int] functionStaticCall
```

Contract functions details

```
    -[Int] functionDelegateCall
    -[Int] functionDelegateCall
    -[Pvt] _verifyCallResult

+ERC20 (Context, IERC20, Ownable)
    -[Pub] <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] _setupDecimals #
    -[Int] _beforeTokenTransfer #

+RASTAToken (ERC20)
    -[Pub] <constructor>
    -[Pub] mintTo #
      -modifiers: onlyOwner

($) = 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.	Passed
18.	Safe Open Zeppelin contracts implementation and usage.	Passed
19.	Incorrect Naming State Variable	Passed
20.	Too old version	Low issue

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**
  No medium severity issues found.
- **Low Severity Issues**
  Two low severity issue found.

1.Old compiler version

- **Description**
  Contract has been deployed using too old solidity version.
- **Recommendation**
  It is advisable to deploy contract using any of the latest version of solidity.

2.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 >=0.6.0 <0.8.0 the contract should contain the following line:
  pragma solidity 0.6.12;

Centralization

Owner Privileges:

- Owner can transfer and renounce ownership.
- Owner can mint.

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 has not been renounced. Following are Admin functions :

- Mintto
- Renounceownership
- Transferownership

Conclusion

Smart contract contains low 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.