HackSafe

Smart Contract Security Audit Report

SHROOM

March 2023

Security Status

★★★★½

www.hacksafe.io

Audit Details

Audited project
SHROOM

Deployer address
0xda56b93b787da55f3d2f725a37171648ae09f87c

Client contacts
SHROOM Team

Blockchain
Ethereum

Website
Not Provided

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 SHROOM to perform an audit of smart contracts:

- https://etherscan.io/token/0xed0439eacf4c4965ae4613d77a5c2efe10e5f183#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 21.03.2023

Token Type	: DEFI
Contract name	: SHROOMToken
Contract address	: 0xEd0439EACf4c4965AE4613D77a5C2Efe10e5f183
Total supply	: 65,557,424.107142857142856874
Token ticker	: SHROOM
Decimals	: 18
Token holders	: 3,887
Transactions count	: 125,663
Compiler version	: v0.6.12+commit.27d51765
Contract deployer address	: 0xda56b93b787da55f3d2f725a37171648ae09f87c
owner address	: 0x026EafB25dDf7D754D3a66a56C9a6400E99C26C5

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.

Insecure	Poor secured	Secure	Well-secured

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 0 low.

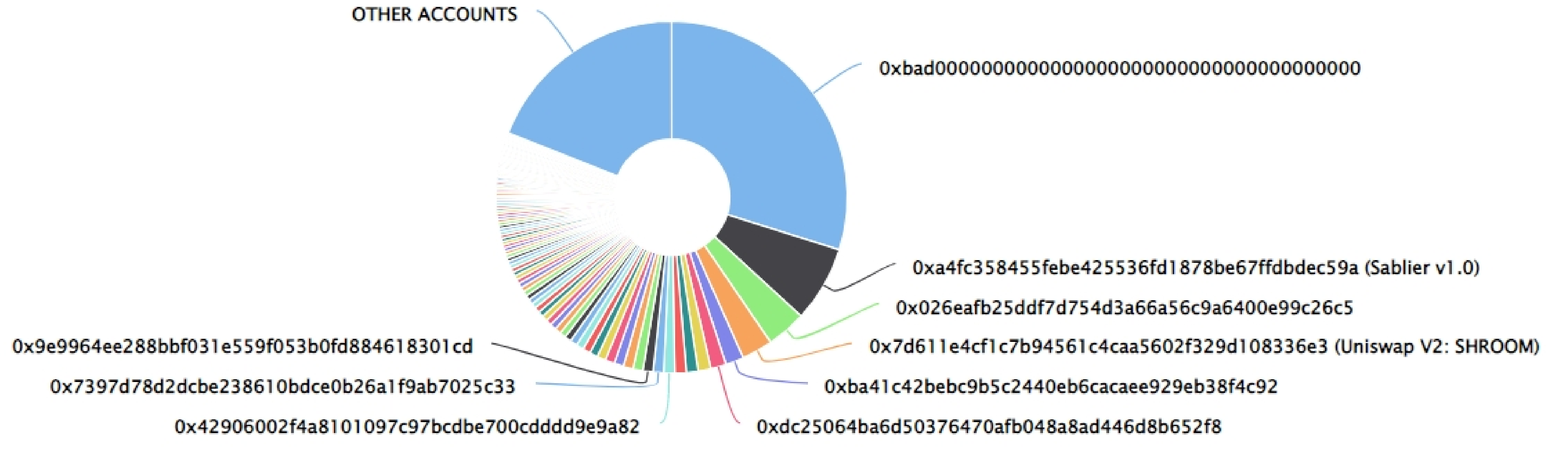
SHROOM TOKEN Distribution

The top 100 holders collectively own 80.97% (53,083,037.26 Tokens) of shroom.finance

Token Total Supply: 65,557,424.11 Token | Total Token Holders: 3,886

shroom.finance Top 100 Token Holders

Source: Etherscan.io



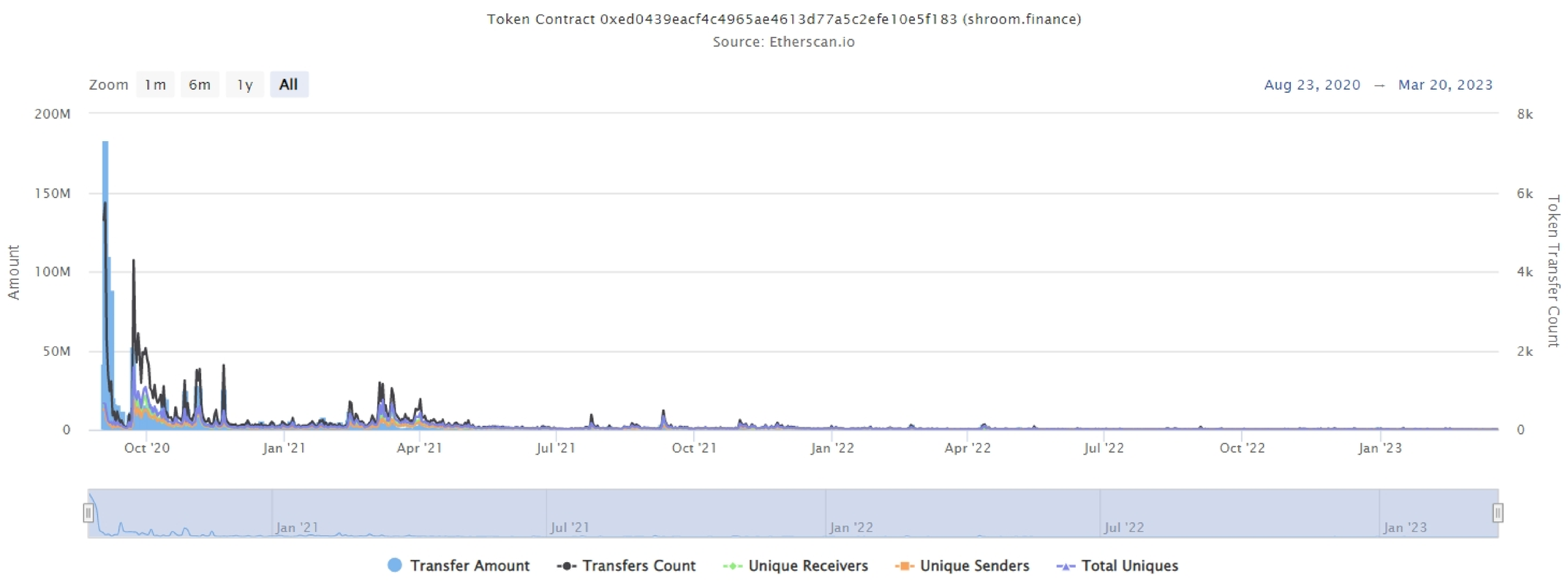
SHROOM Token Top 20 Token Holders

(A total of 53,083,037.26 tokens held by the top 100 accounts from the total supply of 65,557,424.11 token)

Rank	Address	Quantity (Token)	Percentage
1	0xbAd000...00000000	19,564,608.114627099245490681	29.8435%
2	Sablier v1.0	4,580,539.9264399999999996	6.9871%
3	0x026Eaf...E99C26C5	2,381,359.323241481751009048	3.6325%
4	Uniswap V2: SHROOM	1,882,678.350327569633027351	2.8718%
5	0xba41c4...B38f4C92	1,093,990.94194681267895159	1.6688%
6	0xDc2506...D8b652F8	928,667.664905724871233504	1.4166%
7	0x4af918...0a373955	777,719.629826713004242217	1.1863%
8	0x7F40E7...94C4C87B	707,813.763462349	1.0797%
9	0xad2a4F...220d5D94	690,521	1.0533%
10	0x429060...dd9E9a82	651,368	0.9936%
11	0x7397D7...b7025C33	637,619	0.9726%
12	0x9e9964...618301CD	619,666.444969955607744158	0.9452%
13	0xD8bA32...a71FDad3	592,398.184443605178955698	0.9036%
14	0x03d1bb...68B8c33e	586,317.165248612079615206	0.8944%
15	0x8bD774...02cE46DF	573,960.569360788466017853	0.8755%
16	Niftyx Protocol: SHROOM Token	564,355.328109498090756496	0.8609%
17	0x86C800...535c179a	519,370.07356	0.7922%
18	0x5e9BfA...17a26488	486,190.065778242213408853	0.7416%
19	0xd05921...24583852	472,436.279763	0.7206%
20	0x8BeC57...13952206	451,381.459378622293407974	0.6885%

SHROOM TOKEN Distribution

SHROOM Contract Overview



Contract functions details

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

+[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
    -[Pvt] _functionCallWithValue

+ERC20 (Context, IERC20)
    -[Pub] <constructor> #
    -[Pub] name
    -[Pub] symbol
    -[Pub] decimals
    -[Pub] totalSupply
    -[Pub] balanceOf
    -[Pub] transfer #
    -[Pub] allowance

Contract functions details

-[Pub] approve #
-[Pub] transferFrom #
-[Pub] increaseAllowance #
-[Pub] decreaseAllowance #
-[Int] _transfer #
-[Int] _mint #
-[Int] _burn #
-[Int] _approve #
-[Int] _setupDecimals #
-[Int] _beforeTokenTransfer #

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

+ SHROOMToken (ERC20)
-[Pub] mint #
-modifiers: onlyOwner

($) = payable function
= non-constant function

Issues Checking Status

No.	Title	Status
1.	Compiler error	Passed
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	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**
  No medium severity issue found.
- **Low Severity Issues**
  No low severity issue found.

Centralization

Owner Privileges :

- SHROOM Contract:
    - Owner can mint tokens.

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 the owner functions:

- mint

Conclusion

Smart contract contains no 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.