HackSafe

Smart Contract Security Audit Report

VeriSafe

October 2022

Security Status

★ ★ ★ ★ ★

www.hacksafe.io

Audit Details

Audited project
VeriSafe

Deployer address
0x7C1bEDC9bD8f7AB32dA3e7e00bf6EA32f15D733d

Client contacts
VeriSafe 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 VeriSafe to perform an audit of smart contracts:

- https://etherscan.io/address/0xac9ce326e95f51b5005e9fe1dd8085a01f18450c#code

The purpose of the audit was to achieve the following:

- Ensutre 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 03.10.2022

Token Type	: ERC20
Contract name	: VeriSafe
Contract address	: 0xAC9ce326e95f51B5005e9fE1DD8085a01F18450c
Total supply	: 19,395,726,268.995742969380698663
Token ticker	: VSF
Decimals	: 18
Token holders	: 714
Transactions count	: 8,653
Compiler version	: v0.5.2+commit.1df8f40c
Contract deployer address	: 0x7C1bEDC9bD8f7AB32dA3e7e00bf6EA32f15D733d
Owner address	: No owner

Social profiles

Twitter Profile	: https://twitter.com/VeriSafeProject
Telegram profile	: https://t.me/VeriSafe
Coinmarketcap profile	: https://coinmarketcap.com/currencies/verisafe/
Coingecko profile	: https://www.coingecko.com/en/coins/verisafe/

Audit Summary

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

Insecure	Poor secured	Secure	Well-secured

You are here 

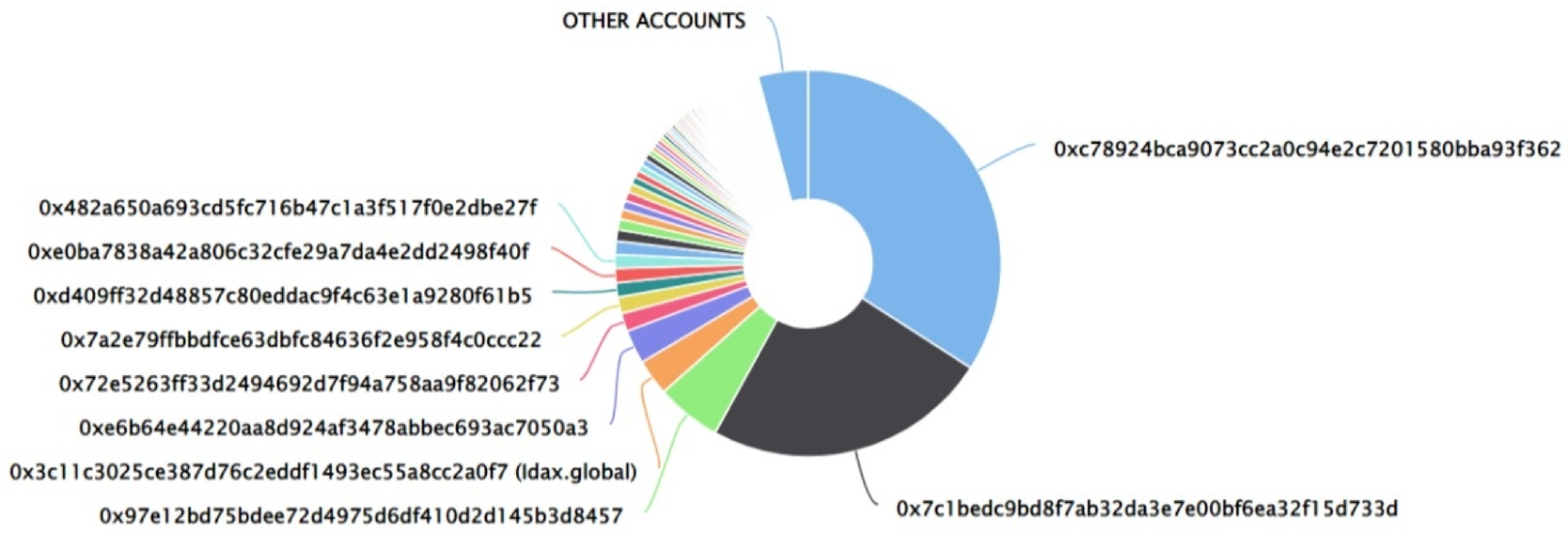
We used various tools like Slither, Mythril and Remix IDE. At the same time this finding is based on critical analysis of the manual audit. All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the issues checking status.

We found 0 critical, 0 high, 0 medium and 2 low and some very low-level issues. These issues are not critical ones.

VeriSafe Token Distribution

The top 100 holders collectively own 95.84% (18,589,172,210.15 Tokens) of VeriSafe

Token Total Supply: 19,395,726,269.00 Token | Total Token Holders: 714



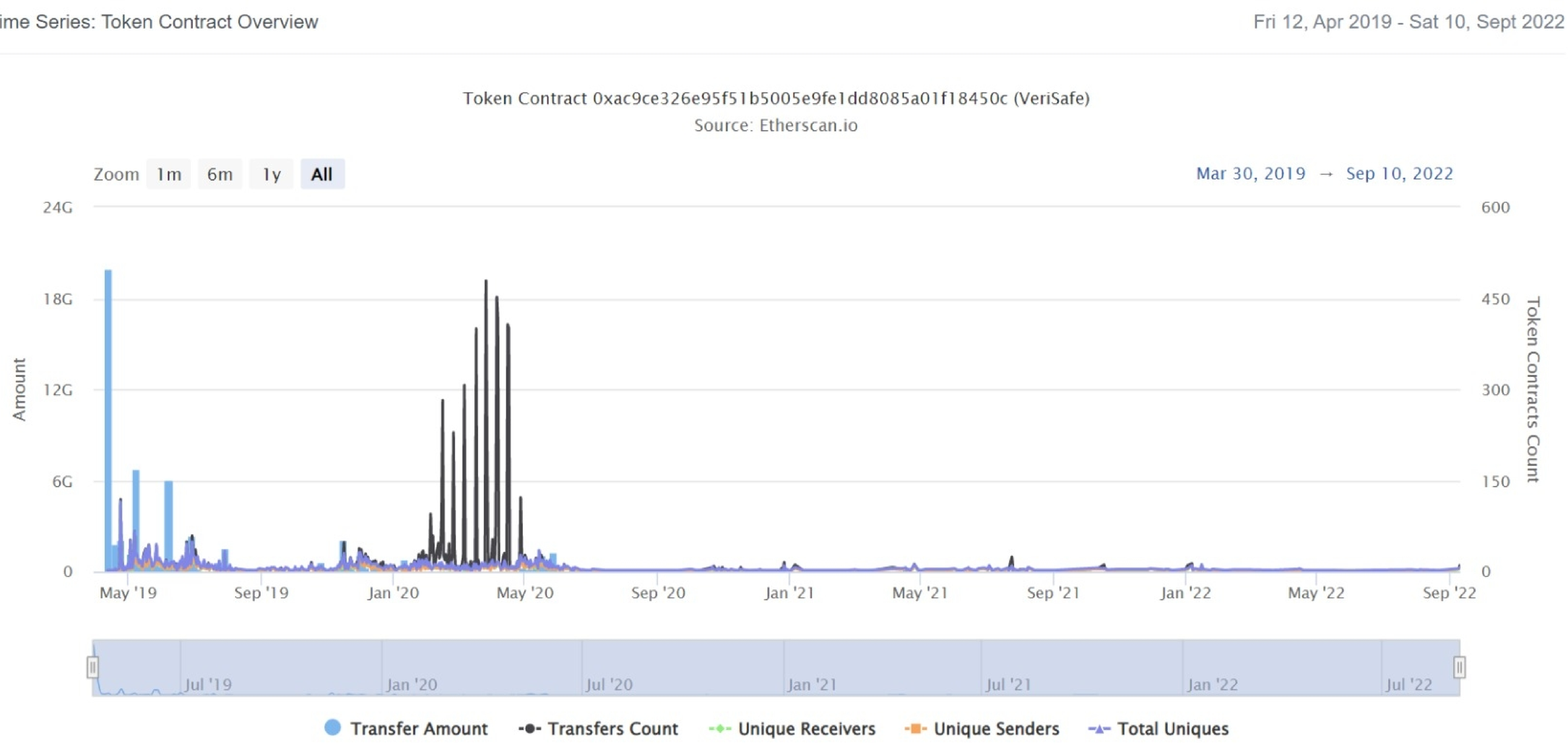
VeriSafe Top 20 Token Holders

(A total of 18,589,172,210.15 tokens held by the top 100 accounts from the total supply of 19,395,726,269.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0xc78924bca9073cc2a0c94e2c7201580bba93f362	6,613,967,245.5092	34.1001%
2	0x7c1bedc9bd8f7ab32da3e7e00bf6ea32f15d733d	4,636,826,383.578037048718227289	23.9064%
3	0x97e12bd75bdee72d4975d6df410d2d145b3d8457	1,054,017,291.853771845847549952	5.4343%
4	Idax.global	589,159,783.979634159785828352	3.0376%
5	0xe6b64e44220aa8d924af3478abbec693ac7050a3	550,240,472.243896	2.8369%
6	0x72e5263ff33d2494692d7f94a758aa9f82062f73	292,906,128.695617015906951164	1.5102%
7	0x7a2e79ffbbdfce63dbfc84636f2e958f4c0ccc22	264,822,123.000000000285212672	1.3654%
8	0xd409ff32d48857c80eddac9f4c63e1a9280f61b5	235,406,500.58830671	1.2137%
9	0xe0ba7838a42a806c32cfe29a7da4e2dd2498f40f	233,066,240.36682375614680496	1.2016%
10	0x482a650a693cd5fc716b47c1a3f517f0e2dbe27f	220,020,962.84965635018501382	1.1344%
11	0x817d7ec561f58c18b66138b093bf7a890b0f4947	214,205,366.750400502153950208	1.1044%
12	0xc597a5d92f5dc1137d847db733cfe02c6d209f13	186,368,111.90133672324299776	0.9609%
13	0xa8cba3415ca5594461c9f07131ceb4ef0e1bc24d	175,006,791.000000001811939328	0.9023%
14	0x5294057157f9c4c4c63850644c9494d3e63f0b06	157,290,930	0.8110%
15	0xd00995a10db2e58a1a90270485056629134b151b	147,807,884.809489396098883075	0.7621%
16	0xea57907a40b73d25f33e669ab10aff185eaa72a3	145,726,314.095065290645431296	0.7513%
17	0xd730dd90cc83549407a2f8c305cefd102d56f247	140,835,172.661846760563773172	0.7261%
18	0x58175d33774ade676e459448d0bd0178dbf86544	121,248,674.32422284985237504	0.6251%
19	0xce03fa2dfad2d94ba940ef0cfb8919f2dd39e758	111,974,318	0.5773%
20	Null Address: 0x00...dEaD	108,761,886.237744788045446952	0.5608%

VeriSafe Token Distribution

VeriSafe Contract Overview



Contract functions details

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

+[Lib] Address
    -[Int] isContract

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

+ApproveAndCallFallBack
    -[Pub] receiveApproval

+ERC20 (IERC20)
    -[Pub] totalSupply
    -[Pub] balanceOf
    -[Pub] allowance
    -[Pub] transfer #
    -[Pub] approve #
    -[Pub] transferFrom #
    -[Pub] increaseAllowance #
    -[Pub] decreaseAllowance #
    -[Int] _transfer #
    -[Int] _mint #
    -[Int] _burn #
    -[Int] _approve #
    -[Int] _burnFrom #
    -[Pub] approveAndCall

+ ERC20Burnable (ERC20)
    -[Pub] burn #
    -[Pub] burnFrom #

Contract functions details

+ ERC20Detailed (IERC20)
    -[Pub] <constructor>
    -[Pub] name
    -[Pub] symbol
    -[Pub] decimals

+[Lib] SafeERC20
    -[Int] safeTransfer
    -[Int] safeTransferFrom
    -[Int] safeApprove
    -[Int] safeIncreaseAllowance
    -[Int] safeDecreaseAllowance
    -[Pvt] callOptionalReturn

+ VeriSafe (ERC20, ERC20Detailed, ERC20Burnable)
    -[Pub] <constructor>

($) = 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 issues found.

- **Medium Severity Issues**
  No medium severity issues found.

- **Low Severity Issues**
  Two 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 ^0.5.2 the contract should contain the following line:

  pragma solidity 0.5.2;

2. Too 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.

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.