HackSafe

Smart Contract Security Audit Report

VRYNT

June 2022

Security Status

★★★★☆

www.hacksafe.io

Audit Details

Audited project
VRYNT

Deployer address
0x680aC01862FdAd7d2B09756fCd897589A8252FeA

Client contacts
VRYNT

Blockchain
Avalanche

Website
https://vrynt.io/

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 VRYNT Token to perform an audit of smart contracts:

- https://snowtrace.io/address/0xe457c5fe85d0563b2ca389354da307516a8edd87#code

The purpose of the audit was to achieve the

- 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 08.06.2022

Token Type	: ERC20
Contract name	: VRYNT
Contract address	: 0xE457c5Fe85d0563B2CA389354Da307516a8EDD87
Compiler version	: v0.8.4+commit.c7e474f2
Total supply	: 10,000,000,000
Token Ticker	: VRYNT
Decimals	: 18
Token Holders	: 8
Top 100 token holder's dominance	: 100%
Transactions count	: 20
Contract deployer address	: 0x680aC01862FdAd7d2B09756fCd897589A8252FeA
owner address	: 0x680aC01862FdAd7d2B09756fCd897589A8252FeA

Social profiles

CoinmarketCap profile	: https://coinmarketcap.com/currencies/vrynt/
Twitter Profile	: https://twitter.com/VRYNT_
Telegram Profile	: https://t.me/VRYNT_io
LinkedIN Profile	: https://www.linkedin.com/company/vrynt/
Medium Profile	: https://medium.com/vrynt

Audit Summary

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

Insecure	Poor secured	Secure	Well-secured

You are here 

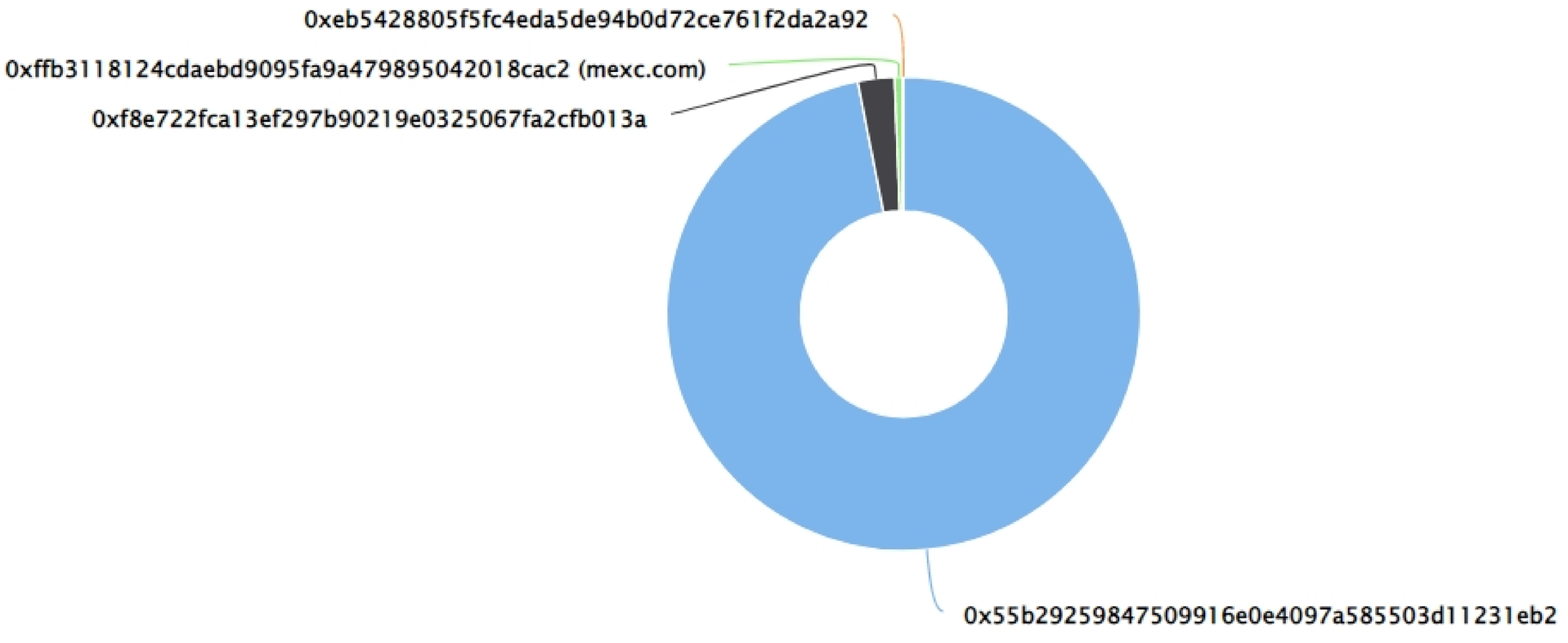
We used various tools like Slither, Mythril and Remix IDE. At the same time this finding is based on critical analysis of the manual audit. All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 1 critical, 0 high, 0 medium and 3 low and some very low-level issues. These issues are critical ones.

VRYNT Token Distribution

VRYNT Top 100 Token Holders

Source: snowtrace.io



VRYNT Token Top 8 Token Holders

(A total of 10,000,000,000.00 tokens held by the top 100 accounts from the total supply of 10,000,000,000.00 token)

Rank	Address	Quantity (Token)	Percentage
1	0x55b29259847509916e0e4097a585503d11231eb2	9,688,999,990	96.8900%
2	0xf8e722fca13ef297b90219e0325067fa2cfb013a	250,000,000	2.5000%
3	mexc.com	57,493,568.41	0.5749%
4	0xeb5428805f5fc4eda5de94b0d72ce761f2da2a92	3,500,005	0.0350%
5	0x38bce86ae32c45765e5b11539be27193c94b99c6	5,834.19	0.0001%
6	0xdc04ddc1b14135cf806b84b0863bd987dffe52b4	480.4	0.0000%
7	0x624a6738fcabef1b015333dda68e7e38e45a8603	117	0.0000%
8	0x44d5f145d4f4f70ec5cf5e57f29367016783f1e5	5	0.0000%

Contract functions details

VRYNT.sol
+ VRYNT (ERC20, Ownable)

-< constructor >#

-[Ext] mint #

-modifiers: onlyOwner

-[Ext] burn #

-modifiers:onlyOwner

Ownable.sol
+ Ownable (Context)

-< constructor > #

-[Pub] owner

-[Pub] renounceOwnership #

-modifiers: onlyOwner

-[Pub] transferOwnership #

-modifiers: onlyOwner

-[Int] _transferOwnership #

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 #

Contract functions details

Context.sol
+ Context

    -[Int] _msgSender
    -[Int] _msgData

IERC20.sol
+ [Int] IERC20

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

IERC20Metadata.sol
+ [Int] IERC20Metadata

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

($) = 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.	Critical issue
11.	Economy model of the contract.	Passed
12.	Private use data leaks.	Passed
13.	Malicious Event log.	Passed
14.	Scoping and Declarations.	Low issue
15.	Uninitialized storage pointers.	Passed
16.	Arithmetic accuracy.	Passed
17.	Design Logic.	Passed
18.	Safe Open Zeppelin contracts implementation and usage.	Low issue
19.	Incorrect Naming State Variable	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.

1. Methods execution permissions.

Unused function.

- **Description**

  Burn function can called by only owner which can create more trouble for users.

- **Location**

  VRYNT.sol -> burn function.

- **Recommendation**

  We advise you to remove onlyOwner modifier form burn function as owner have access to burn any amount of tokens of any users which is the critical issue.

High Severity Issues

No high severity issue found.

Medium Severity Issues

No medium severity issues found.

Low Severity Issues

Three 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 v0.8.4 the contract should contain the following line:

  pragma solidity 0.8.4;

2. Scoping and Declarations.

Unused function.

- **Description**
  The _msgData functions do nothing.
  Library Address does nothing.

- **Location**
  _msgData functions.

- **Recommendation**
  We advise to remove unused code which can help you to develop clean coding style and save some computational gas too.

3. Safe Open Zeppelin contracts implementation and usage.

Import openzeppelin files.

- **Description**
  Contract have import openzeppeline files direct form their github repository.

- **Location**
  VRYNT.sol file.

- **Recommendation**
  We advise to remove import lines form particular file because any modifications in github file can affect your contract too..

Centralization

Owner Privileges (in the period when the owner is not renounced) :

- VRYNT CHAIN Contract:
    - Owner can transfer ownership.
    - Owner can renounce ownership.
    - Owner can mint new tokens.
    - Owner can burn tokens of any address.

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. Following are Admin functions:

- **Mint** : Owner can mint new tokens.
- **Burn** : Owner can burn any amount of user’s tokens.
- **Transferownership** : owner can transfer ownership.
- **Renounceownership** : Owner can renounce ownership, if owner pause transfers and renounce wnership then it may happen that user will not be able transfer their tokens as only owner can npause the transfers in this case owner will be renounce to zero address.

Conclusion

Smart contract contains few 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.