HackSafe

Smart Contract Security Audit Report

CryCash

September 2022

Security Status

★ ★ ★ ★ ⯪

www.hacksafe.io

Audit Details

Audited project
CryCash

Deployer address
0x40410023209bb53A104b395088d9c1595f0c9D13

Client contacts
CryCash Team

Blockchain
Ethereum

Website
https://crycash.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 CryCash to perform an audit of smart contract:

- https://etherscan.io/token/0xf41e5fbc2f6aac200dd8619e121ce1f05d150077#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 17.09.2022

Token Type	: ERC20
Contract name	: CryCashToken
Contract address	: 0xF41e5Fbc2F6Aac200Dd8619E121CE1f05D150077
Compiler version	: v0.4.19+commit.c4cbbb05
Total supply	: 7,057,137.099868197347603743
Token ticker	: CRC
Decimals	: 18
Token holders	: 4,160
Transactions count	: 11,838
Contract deployer address	: 0x40410023209bb53A104b395088d9c1595f0c9D13
Owner address	: No owner

Social profiles

Twitter Profile	: https://twitter.com/cry_cash
Telegram profile	: https://t.me/crycash
Facebook profile	: https://www.facebook.com/crycash.io/
Coinmarketcap profile	: https://coinmarketcap.com/currencies/crycash/
Coingecko profile	: https://www.coingecko.com/en/coins/crycash/

Claimed Smart Contract Features

Claimed Feature Detail

Tokenomics :

- **Name** : CryCashToken
- **Symbol** : CRC
- **Decimals** : 18
- **Protocol** : ERC20
- **Total supply** : 7,057,137.099868197347603743
- **Contract address** : 0xF41e5Fbc2F6Aac200Dd8619E121CE1f05D150077

Our Observation

YES, this is valid.

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are “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.

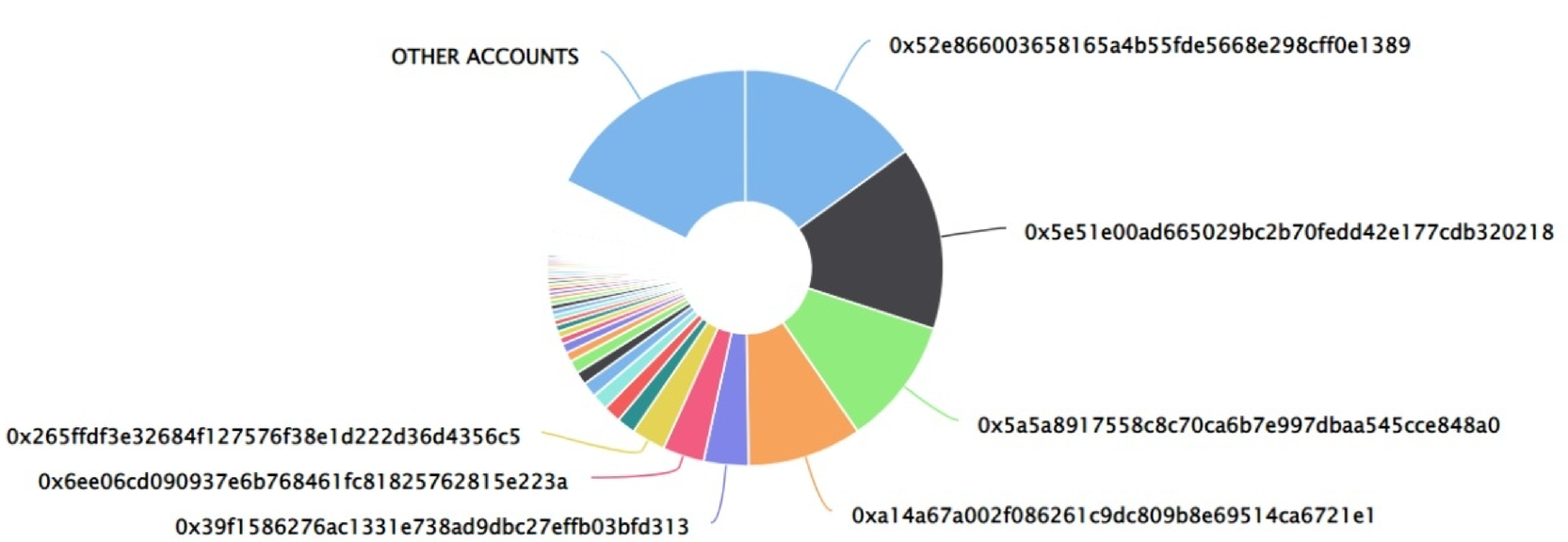
CryCash Token Distribution

The top 100 holders collectively own 82.21% (5,801,939.17 Tokens) of CryCash

Token Total Supply: 7,057,137.10 Token | Total Token Holders: 4,160

CryCash Top 100 Token Holders

Source: Etherscan.io



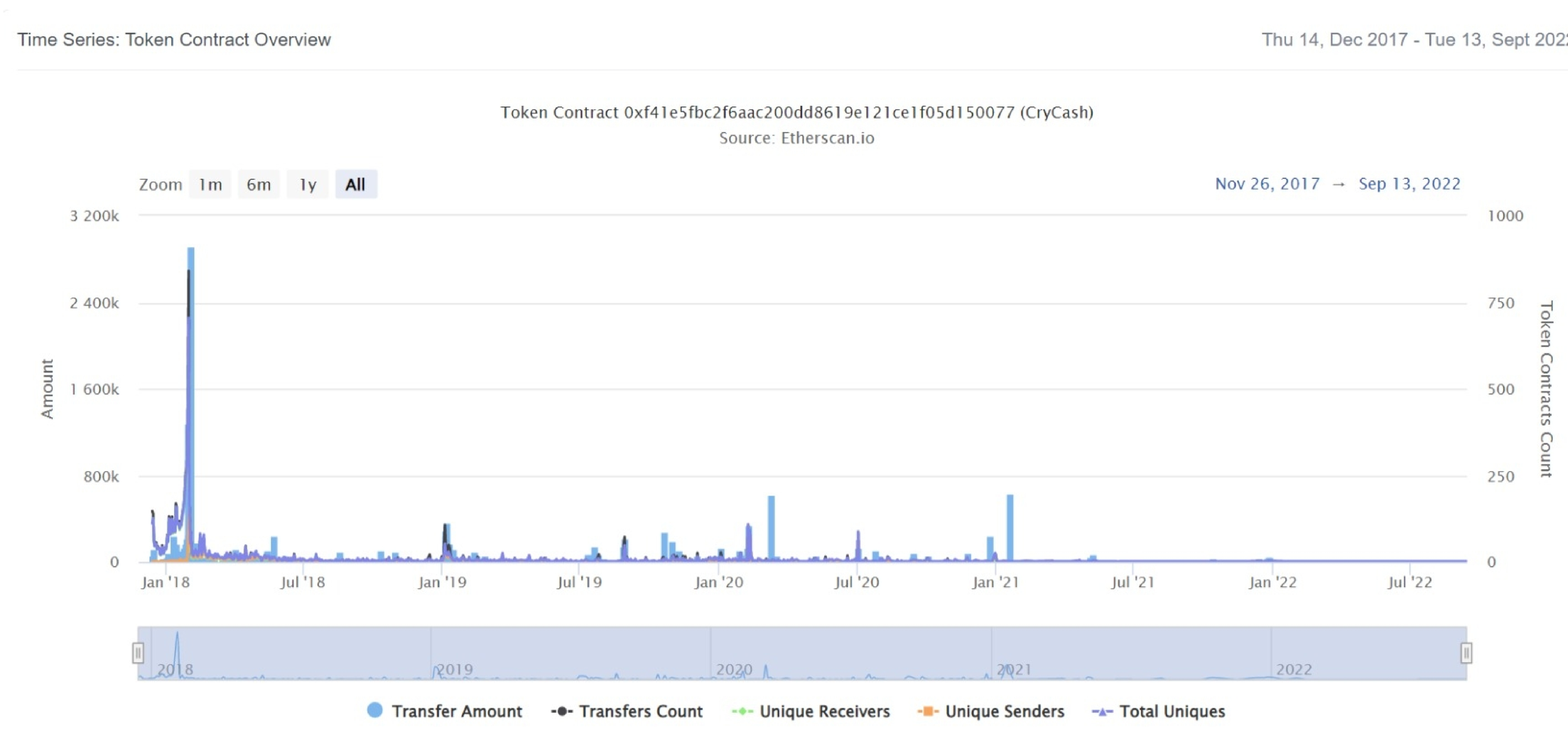
CryCash Top 20 Token Holders

(A total of 5,801,939.17 tokens held by the top 100 accounts from the total supply of 7,057,137.10 token)

Rank	Address	Quantity (Token)	Percentage
1	0x52e866003658165a4b55fde5668e298cff0e1389	1,058,570.564980229602140555	15.0000%
2	0x5e51e00ad665029bc2b70fedd42e177cdb320218	1,058,570.564980229602140555	15.0000%
3	0x5a5a8917558c8c70ca6b7e997dbaa545cce848a0	735,643.878430661736837197	10.4241%
4	0xa14a67a002f086261c9dc809b8e69514ca6721e1	657,711.60712065	9.3198%
5	0x39f1586276ac1331e738ad9dbc27effb03bfd313	257,628.281506193110535481	3.6506%
6	0x6ee06cd090937e6b768461fc81825762815e223a	239,543.292248306127844638	3.3943%
7	0x265ffdf3e32684f127576f38e1d222d36d4356c5	196,656.651873486005888	2.7866%
8	0xf742c62bd8483dd103258c7fbea4ebca69b208f7	105,999.999999999993708544	1.5020%
9	0xfd32191e9ad42a322b95d24b86368d86ebf05eeb	100,060.01198681973476037	1.4179%
10	0x3b05cfb09d198071a779d4c1f96a30b008776673	94,133.89010989	1.3339%
11	0x4f3e7bf5649906ceaab4074a5486d304a2257d4c	86,294.90861248	1.2228%
12	0xa53110ef7dc9ab341a3ef074d7fa1df9a1f403d8	75,000	1.0628%
13	0x752ad1a75af23e08ba5b521a3e705e25d856d4e4	75,000	1.0628%
14	0x48cdebe5d506da65818d5a8df525fb0f58912f95	54,857.88457017876	0.7773%
15	0x83d8bcfa7b9a1db966d78fef0d39afce65d746c3	54,352	0.7702%
16	0x3cf092fd921d37bd459bc3c1b811dbbd51461000	38,118.510745595956	0.5401%
17	0x6a4e0253a1eb8fc04b15a0fc7e80bd97431cb673	37,272.62593039	0.5282%
18	0xaea7ba6d1d398403d8c8b931591aee0e2e70d9eb	36,437.15734471	0.5163%
19	0xf0b0c5cead98c3d7b164f3597c09119f022cb5cf	30,092.000000000004194304	0.4264%
20	0xc60601d874926b70b79ace4a78ecd3439d9399fa	30,000	0.4251%

CryCash Token Distribution

CryCash Contract Overview



Contract functions details

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

+ERC20Basic
    -[Pub] balanceOf
    -[Pub] transfer

+BasicToken (ERC20Basic)
    -[Pub] transfer #
    -[Pub] balanceOf

+ERC20 (ERC20Basic)
    -[Pub] allowance
    -[Pub] transferFrom
    -[Pub] approve

+StandardToken (ERC20, BasicToken)
    -[Pub] transferFrom #
    -[Pub] approve #
    -[Pub] allowance
    -increaseApproval #
    -decreaseApproval #

+CryCashToken (StandardToken)
    -[Pub] CryCashToken #
    -[Pub] mint #
      -modifiers: canMint
    -[Pub] finishMinting #

($) = 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. Too old compiler version.

- **Description**

  Contract has been deployed using too old compiler version.

- **Recommendation**

  It is advisable that the compiler version of solidity should be among the new compiler versions.

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 ^0.4.17 the contract should contain the following line:

  pragma solidity 0.4.19;

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.