#HashEx
BLOCKCHAIN SECURITY

Kangoo

smart contracts final audit report

December 2021

hashex.org

contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice 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 disclaimer below – please make sure to read it in full.

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 HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx 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 HashEx 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, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, 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 the 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. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org).

2. Overview

HashEx was commissioned by the Kangoo team to perform an audit of their smart contracts. The audit was conducted between 07.11.2021 and 19.11.2021.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts.
- Formally check the logic behind given smart contracts.

Information in this report should be used to understand the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The audited contracts are deployed to the Binance Smart Chain mainnet:

KangarooToken 0x58957DdfbaAa4b643A7dC3aDef6Df6e4606FcB9e,

KangarooStake 0x28d7257356A8a152c4a672134f6a40400a61664B,

KangarooDistribution 0x64f4022B3BDfddE9Eb58Fd11C1d68C3303d1056e, and

KangarooMatrix 0x6CF735f8eAE7aBcA215729AF9f464eF1971e3c43.

Sufficient documentation was provided by the team.

Recheck was performed on updated contracts in the Binance Smart Chain mainnet:

KangarooStake 0x5Db7190EC23379acf94f9b28ea3F0b597Dd87C37,

KangarooDistribution 0xAb9800c3987217Ae9b586cfaf3fdD7AA51Dd66c3, and

KangarooMatrix 0x36160A1D24d30440866346819A917b719bD7D257.

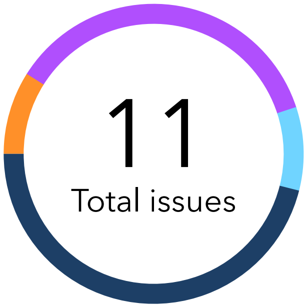
2.1 Summary

Project name	Kangoo
URL	https://kangoo.group
Platform	Binance Smart Chain
Language	Solidity

2.2 Contracts

Name	Address
KangarooStake	`0x5Db7190EC23379acf94f9b28ea3F0b597Dd87C37`
KangarooMatrix	`0x36160A1D24d30440866346819A917b719bD7D257`
KangarooDistribution	`0xAb9800c3987217Ae9b586cfaf3fdD7AA51Dd66c3`
KangarooToken	`0x58957DdfbaAa4b643A7dC3aDef6Df6e4606FcB9e`

3. Found issues



●	High	1 (9%)
●	Medium	4 (36%)
●	Low	1 (9%)
●	Info	5 (46%)

C1. KangarooStake

ID	Severity	Title	Status
C1-01	● High	Wrong rewards distribution	✓ Resolved
C1-02	● Info	emergencyWithdraw() restrictions	✓ Acknowledged
C1-03	● Info	Insufficient amount of events	✓ Acknowledged
C1-04	● Info	firstStaking() problem	✓ Acknowledged

C2. KangarooMatrix

ID	Severity	Title	Status
C2-01	● Medium	_updateMatrixesTokenRate() calling	✓ Acknowledged
C2-02	● Medium	switchToOpenSale() problem	✓ Resolved

C3. KangarooDistribution

ID	Severity	Title	Status
C3-01	● Low	Percentage restrictions	✓ Acknowledged
C3-02	● Info	distribute() problem	✓ Acknowledged

C4. KangarooToken

ID	Severity	Title	Status
C4-01	● Medium	trasferFrom() function problem	✓ Acknowledged
C4-02	● Medium	Token distribution	✓ Resolved
C4-03	● Info	Insufficient amount of events	✓ Acknowledged

4. Contracts

C1. KangarooStake

Overview

This is a contract where users can stake LP tokens

(KangarooToken/BUSD-T pair)

for 3 different time durations. Rewards are accrued in KangarooToken and distributed from KangarooMatrix via KangarooDistribution contract.

Issues

C1-01 Wrong rewards distribution ● High ✓ Resolved

3 staking pools have different minimal staking periods and specific rewards distribution: 50% of rewards should be distributed over all the users, 30% over the users in 1-year and 6-months pools, and 20% goes to the 1-year pool exclusively. However, the deployed contract pools were set in reverse order, so the 1-month pool receives the most reward.

The Kangaroo team is aware of this problem and assures that the contract will be redeployed with the state from the snapshot.

Update

The contracts were redeployed in order to fix this issue. Matrix contract state was migrated to the new one.

C1-02 emergencyWithdraw() restrictions ● Info ✓ Acknowledged

The function `emergencyWithdraw()` won't work in the case of early withdrawal.

C1-03 Insufficient amount of events ● Info ✓ Acknowledged

Functions `createUser()` and `startOpenSale()` don't have appropriate events.

C1-04 firstStaking() problem ● Info ✓ Acknowledged

Function `firstStaking()` can be sandwich-attacked if being called during the publicly open transfers period.

C2. KangarooMatrix

Overview

A contract for the MLM referral system.

Issues

C2-01 _updateMatrixesTokenRate() calling ● Medium ✓ Acknowledged

In the functions `buyNewLevel()`, `registrationX3()` and `registrationX6()` the call of the `_updateMatrixesTokenRate()` function should be performed before the call of the `getLevelPrice()` function. In this version, the user pays an amount that is calculated from values from the last call of one of these functions. But between the previous and the current calls, there can be large changes in rates.

C2-02 switchToOpenSale() problem ● Medium ✓ Resolved

In the function `switchToOpenSale()` there should be a `_updateMatrixesTokenRate()` function call. Because in the first call of `buyNewLevel()` , `registrationX3()` or `registrationX6()` function the user will pay in KangarooToken and with an incorrect value.

C3. KangarooDistribution

Overview

This is a contract that will collect and redistribute 20% of KangarooMatrix income tokens. By default, this amount is separated into 3 parts:

1. A specific amount goes to the KangarooStake contract (initially 50%)
2. A specific amount will be burned (initially 40%)
3. An amount to the immutable philanthropy address (initially 10%)

These percentages can be changed by the `setDistributionPercents()` onlyOwner function.

Issues

C3-01 Percentage restrictions ● Low ✓ Acknowledged

With the function `setDistributionPercents()` the owner can set zero percentages to staking or burning. There should be a check of the new values for minimum and maximum values in this function.

C3-02 distribute() problem ● Info ✓ Acknowledged

The function `distribute()` can be sandwich-attacked if it's called during the publicly open transfers period.

C4. KangarooToken

Overview

The main token that will be used in this system. Meets BEP-20 and ERC-20 standards. Transfers are restricted until the owner calls the `startOpenSale()` function. The initial supply was 21M ROO tokens, by Nov'12 2021 total supply was slightly more than 15.4M, the difference was burned.

Issues

C4-01 trasferFrom() function problem ● Medium ✓ Acknowledged

In the contract, there are restrictions on the `transfer()` function, but the `transferFrom()` function isn't restricted. If the owner transfers some of the team tokens to a user before opening transfers to the public, that user would be able to sell tokens via `approve()` & `transferFrom()`.

Team response

We are aware of `transfer()` and `transferFrom()` restrictions. `transferFrom()` function was left open intentionally for the vesting of team tokens. We're not going to distribute tokens to users before the project start.

C4-02 Token distribution ● Medium ✓ Resolved

Almost 50% of all tokens are held by EOA. 90% of PancakeSwap LP tokens supply (KangarooToken/BUSD-T pair) are also held by another EOA.

Recommendation

Secure these tokens by locking them for a fixed time period.

Update

About 41% of all ROO tokens and 81% of LP tokens were vested with CryptEx Locker.

C4-03 Insufficient amount of events ● Info ✓ Acknowledged

Functions `setWhitelistAddress()` and `startOpenSale()` don't have appropriate events.

5. Conclusion

1 high severity issue was found and fixed with the update. The KangarooStake, KangarooDistribution, and KangarooMatrix contracts have been redeployed after the initial version of this report was issued. Staking pools of KangarooStake contracts have been fixed, mitigating the high severity issue, KangarooMatrix state (its users, referrals, and levels) has been migrated to the new contract.

KangarooMatrix contract is highly dependent on the owner’s account. Users have to trust the owner and that the owner's account is properly secured.

This audit includes recommendations on the code improving and preventing potential attacks.

The updated contracts are deployed to the Binance Smart Chain mainnet:

KangarooToken 0x58957DdfbaAa4b643A7dC3aDef6Df6e4606FcB9e,

KangarooStake 0x5Db7190EC23379acf94f9b28ea3F0b597Dd87C37,

KangarooDistribution 0xAb9800c3987217Ae9b586cfaf3fdD7AA51Dd66c3, and

KangarooMatrix 0x36160A1D24d30440866346819A917b719bD7D257.

Appendix A. Issues’ severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Informational.** Issues that do not impact the contract operation. Usually, informational severity issues are related to code best practices, e.g. style guide.

Appendix B. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

✉ contact@hashex.org

@hashexbot

blog.hashex.org

linkedin

github

twitter

#HashEx
BLOCKCHAIN SECURITY