CatNIP

Security Assessment

KichiCoin
(KICH)

June 20, 2021

About KichiCoin

KICH is an innovative cryptocurrency that has a weekly lottery system. By holding a minimum amount of tokens, holders are given tickets into the pool. Each week winners are chosen, and every day they are given their reward amount. After the winner gets all their rewards a new cycle starts, a new winner is chosen, and they get rewards.

The token works by tracking the lotto participants in an array. By using some unique programming logic it is inexpensive to add people to the array, remove from the array, and to pick the winner.

The lottery is chosen at random by using Chainlink VRF. This means that it is 100% random so participants can know they won by chance.

The following tokenomics taxes are on each transaction: 3% Redistribution, 2% Liquidity, 3% Charity, 2% Burn, 5% Lottery. These taxes are taken in the RFI way, where reflections are given to the holders and the specific wallets.

There are mechanisms to control most aspects of the system, so turning on and off taxes and the lottery system is given to the owner.

Overview

Project Summary

Project Name	KichiCoin
Description	KichiCoin is a token with a lottery drawing feature.
Platform	Binance Smart Chain
Language	Solidity
Code	https://github.com/yoshiko2000/kichi Will be made public

Audit Summary

Delivery Date	June 16, 2021
Manual Review	Yes
Video Review	No
Auditor	Yoshiko Shinonome

Findings Summary

Total Issues	3
🔴 Major	0
🟡 Moderate	0
● Minor	3

Findings List

Severity	Description	Solution	Status
● Minor	4% of supply held by team wallets.	• N/A Acceptable Amount of tokens held.	Acknowledged
● Minor	Team wallets can participate in lottery.	• Restrict wallets from lottery entry.	Acknowledged
● Minor	Lottery array gas usage.	• Reduce number of entrants into the lottery if problems.	Acknowledged

Result: A

This project is safe to invest in. The code contains no malicious exploits that can harm investors. There are only minor problems as detailed.

There are minor problems with the lottery array, team wallet lottery participation, and team wallet holding amounts.

There are no moderate problems to address.

There are no major problems to address.

Detailed Overview

Contract Imports

SafeMath.sol, Address.sol, Context.sol

SafeMath, Address, Context are audited contracts from OpenZeppelin. These contracts are safe, and no extra functions were added. Each of these contracts provide helpful functionality to the contract to operate safely and correctly.

IBEP20.sol

IBEP20 is an interface for other BEP20 tokens on the Binance Smart Chain. This is a standard contract almost identical to IERC20.sol. There are concerning issues with the contract.

SafeBEP20.sol

SafeBEP20 is similar to IBEP20, but provides a safer way to transfer tokens. It is generally not used for most transfers, but specifically for rescuing funds to the KICHI Team.

IPancakeFactory.sol, IPancakeRouter01.sol, IPancakeRouter02.sol

These 3 contracts come from PancakeSwap and are used with their product. As PancakeSwap is a clone of Uniswap, the functionality can be considered the same as the Uniswap Factory Contracts and the Router contracts. All three are safe and used for PancakeSwap interactions.

VRFConsumerBase.sol

This is the standard contract from Chainlink to perform VRF functions. Nothing was modified and nothing is malicious.

Declaration Variables

At the top of the contract are many variables declared. This section will go through any significant variable. Most are generic and associated to the LIQ and RFI systems. No out of place variables were found.

releaseUnixTimeStampV1
records the deployment time of the contract

isLotterySystemEnabled
shows if the lotto system is enabled or not

isExcludedFromLottery
tracks if an account is excluded from the lottery system

lottoPool
contains the addresses that are in the lottery pool

lottoPoolTemp
temp array that is created to help remove and add lotto addresses

hasEnoughTokensForLottery
mapping for addresses saying if they have enough tokens or not to be included in the lottery

numberOfEntriesIntoLottery
tracks how many entries each address has into the lotto

lotteryTime
tracks the time of the lottery that will happen

currentLotteryWinner	records who the current lotto winner is
maxDrawingChances	max number of lotteries a holder can have
amountNeededForDrawingChance	amount needed to be entered into the lotto
amountToDisperseInDrawingTotal	amount of token a lotto winner is awarded
amountToDisperseInDrawingPerPeriod	how much token to be rewarded per period
amountToDisperseInDrawingLeft	remainder of total award amount
periodsToDisperse	how many periods the lottery will be dispersed in
hoursInPeriodToDisperse	how many hours between each dispersal
dispersalTime	when the next dispersal time is to award lotto winners
keyHashForLINK	used in Chainlink VRF to verify the functionality
feeForLINK	used in Chainlink VRF to determine the VRF fee

randomResultFromLINKVRF

catches the result of using Chainlink VRF to determine the lotto winner

linkTokenAddress

address for the LINK token

vrfCoordinatorAddress

address for the LINK VRF Coordinator

Constructor

In the constructor variables are associated values. The ones of note are listed below. No variables were given unusual values.

The variables are correctly set as intended by the project. Each lottery will happen in 24 hours, winnings dispersed over 7 periods, and lottery time is set 7 days from the deployment of the contract.

The owner is set to the deployer and the tokens are generated to the owner's address.

All fees are set to 0 for use with a presale and distribution of tokens.

The number of tokens to sell to liquidity in the LIQ system is set to 0.03% which is better than SafeMoon's 0.05%.

The PancakeSwap pair is created in the constructor.

The owner and contract addresses are excluded from the fee.

The team addresses, the contract address, pair address, and PancakeSwap are excluded from being included in the lottery. The owner address is included in the lotto, but it should not have tokens. The tokens should be kept in an excluded wallet. If the team does not keep good track of which wallets they hold their funds in, then they could end up with a team member winning the lottery. **This is detailed in the minor problems section.**

The Chainlink VRF keyhash and fee is set.

Owner Functions

owner(), getOwner(), onlyOwner(), transferOwnership()
functions that deal with showing who the owner is, requiring the owner to activate the function, and transferring ownership if needed. Nothing is unusual about them.

Basic Functions

decimals(), symbol(), name(), totalSupply(), balanceOf(), transfer(), allowance(), approve(), transferFrom(), increaseAllowance(), decreaseAllowance()
basic functions that are standard in most BEP20 contracts. Nothing is unusual about them.

approveInternal()
a private function that allows approvals to happen from within the contract if needed. This is called by other approval and allowance functions.

getNowBlockTime()
shows current block timestamp.

releaseUnixTimeDate()
shows the timestamp of when the contract was deployed.

Reflect Functions

KICHI uses the RFI system. There were no problems found, and everything works as intended.

totalFees()
records how much the reflect fees have taken

deliverReflectTokens()
burns the amount of reflect tokens from the caller of this function. This is not a burn of anyone else’s tokens.

reflectionFromToken()
handles RFI mechanics for the system

tokenFromReflection()
determines how many reflections tokens from the amount

isExcludedFromReward(), includeInReward()
control who is listed in the RFI reward

takeReflectFee()
gets the fee of the reflect and add it to the totalFeeAmount

getReflectRate()
provides the rate of the reflect system based on total supplies

getCurrentSupplyTotals()
provides the total supplies of the tokens for RFI

getReflectionValues()
gets the reflection values of each tax

getTaxAndReflectionValues()
gets the tax and reflection values and passes them to caller. This calls getReflectionValues().

LIQ Functions

KICH uses LIQ for liquidity boosting. The number of tokens sold into liquidity is 0.03% of the total supply, which is lower and better than SafeMoon. It doesn't affect the price as much and still provides the liquidity at a good rate.

setSwapAndLiquifyEnabled()
sets the swap and liquify system

swapAndLiquify()
swaps tokens and adds them back into liquidity

swapTokensForEth()
swaps tokens for BNB

addLiquidity()
adds the tokens into the liquidity

setNumberOfTokensToSwapAndLiquify()
allows the owner to set the number of tokens threshold to activate the swap and liquify function

Fee Functions

excludeFromFee(), includeInFee(), isExcludedFromFee(), setTaxFeePercent(), setBurnFeePercent(), setCharityFeePercent(), setLotteryFeePercent(), setLiquidityFeePercent(), takeLiquidityFee(), takeCharityFee(), takeBurnFee(), takeLotteryFee(), removeAllFee(), restoreAllFee(), getTaxValues()
These are functions that control the fees. Nothing is unusual about them. Most are controlled by the owner only and can only be set between 0 and 5%, which safeguards the investor.

Transfer Functions

KICH makes use of transfer functions that are heavily modified and do more than just transfer functions.

transferInternal()

the internal transfer function called by public transfer functions where swap and liquify will activate if conditions are met, and checks to see if fees are taken or not. It then calls transferTokens() to do the token transfer functionality.

transferTokens()

In this function several things happen:

- Fees removed and restored as determined in transferInternal()
- Tax and Reflection values are gathered
- Checks if senders and receivers are excluded from rewards
- Takes the fees from the amount being transferred
- Transfers the fee amounts as needed

These are normal functions that are standard in most Safemoon-like contracts. Each of these processes are safe and non-malicious.

At the end of transferTokens(), the mechanics of the lottery system are started. The functions called at the end of transferTokens() are checkForLotteryParticipationOrRemoval(), weeklyLottery(), and lotteryDisperseFromDrawingWallet(). Please refer to the lottery section for descriptions of what these functions do.

Rescue Functions

KICH has the power to rescue funds accidentally sent to contracts. If BNB, any BEP20 token, or KICH itself is sent to the contract, then the team has the power to rescue them and return to sender. KICH is a payable contract, which can cause these mistakes to happen. It should be payable to allow the LIQ functionality to occur.

withdrawBNBSentToContractAddress(),
withdrawBEP20SentToContractAddress(),
rescueAllContractToken(),
rescueAmountContractToken()
These each have the power to remove tokens from the contract address. Nothing malicious was found.

PancakeSwap Functions

PancakeSwap is the DEX of choice for KICH. There are some functions that allow KICH to update router and pair addresses as necessary.

They are setRouterAddress() and setPairAddress(). Each function is safe to use and only able to be called by the owner of the contract.

Lottery Functions

The lottery functions are the core of KICH. Each function is explained here.

weeklyLottery()

chooses a lottery winner. It checks first to see if the lotto pool has at least 1 participant and if it is time for the lottery to commence. It then checks to make sure the amountToDispersInDrawingLeft is zero to ensure all of the award is paid out first before moving on to a new winner.

If the KICH team needs to accelerate the winner, they should have access to the drawing wallet and can manually award funds from it. They can also adjust the periods of dispersal and the time between dispersals.

It then checks to see if there is enough LINK in the contract to perform the Chainlink VRF functionality of choosing a winner.

It then checks to see if there are tokens in the lottery wallet; otherwise, there would be nothing to award the winner.

It then transfers tokens from the lottery wallet to the drawing wallet.

It checks the balance of the drawing wallet, determines how much to award to the winner, sets the dispersal time, and sets the next lotto time.

Finally it randomly chooses a winner using Chainlink VRF, and then excludes the lottery winner from participating again and removes the winner from the lotto pool.

weeklyLotteryManual()

The owner may call to choose a lotto winner manually. This is not expected to be used, but provides a failsafe if needed.

lotteryDisperseFromDrawingWallet()

This function gives the award amount to the lottery winner. It first checks to see if there is enough amount to disperse left in the drawing wallet. It then checks to make sure it is time to disperse.

It then gets the balance of the drawing wallet and, depending on how much it has, the winner is given the amount to disperse per period or whatever is remaining in the drawing wallet. This way, no remainder is left over in the drawing wallet. It also helps catch if the balance of the drawing wallet is modified in some way. For example, if the KICHI team wants to add more to the pot, they can do so in this way.

lotteryDisperseFromDrawingWalletManual()

The owner may call to disperse the lottery manually. This is not expected to be used, but provides a failsafe if needed.

checkForLotteryParticipationOrRemoval()

Checks to see if the address is available to be in the lottery or not. If it is, then add it to the lottery array.

It first checks to see if the address is excluded, then checks the balance of the address, then adds the address to the array based on the amount of entries into the lottery the address should receive.

If the address must be removed because they do not have enough tokens, the address is removed.

transferTokensForLotteryToDrawingOrWinner()

This is a slight modification to the usual transfer function. It will take the fees needed, but will send them to the drawing or lottery wallet. This is needed to not clash with the basic transfer function. Nothing malicious was found in the code.

removeIndexFromLotteryArray()

removes a specific index from the lottery array. This is used when cleaning up the lottery array. It removes a specific index.

removeIndexFromLotteryArrayOwnerOnly()

provides a manual call for the owner to remove a specific index from the array. This is not expected to be used, but provides a failsafe for the owner.

removeAddrFromLottoPoolCompletely()

removes an address completely. Will go through the lotto pool, find the address, then delete the array’s element if the address is found. It does this for each element found as holders can have more than one ticket. It will then call the cleanUpLotteryArray() function.

cleanUpLotteryArray()

This creates a temporary lottery array. It goes through the existing lottery array and pushes them to the new temporary array if the element is not deleted. It then sets the original array to this cleaned up temporary array. There is a minor problem with gas usage. It is a concern, but should work as it did in testing. **This is detailed in the minor problems section.**

getSecondsUntilNextLotto()

determines how many seconds until the next lottery.

setMaxDrawingChances()
sets the maximum number of tickets a holder can have

setAmountNeededForDrawingChance()
sets the threshold for gaining a ticket into the lottery

setPeriodsToDisperse()
sets how many periods to disperse the reward

setHoursInPeriodToDisperse()
sets the hours between each period to disperse

setLotterySystemEnabled()
enables or disables the lottery system

excludeOrIncludeFromLottery()
excludes or includes an account into the lottery

Chainlink VRF

KICH makes use of Chainlink's Verfiably Random Function service for choosing winners in it's lottery.

setKeyHashForLinkVRF(), setFeeForLinkVRF()
allow the owner to change the Key Hash and the fee for VRF calling. It is not expected to ever need to change them, but these functions are provided as a "just in case" mechanism.

getRandomNumber()
picks a random number using VRF.

fulfillRandomness()
gets the random number generated and passes it back to the lottery system to determine who the winner of the lottery is. The number is determined by how many lottery participants there are.

Anonymity

The majority of the team is doxxed, including the founder and CEO, Jimmy Lai. This is very good for the project to have, as it shows they are not trying to hide anything. You can see the team list at https://www.kichicoin.io/#team

Holders

There are 10 Quadrillion KICH tokens. 50% is sent to the burn address.

Of the 5 Quadrillion, 5% is reserved for marketing. 5% is reserved for development. 10% is reserved for the team wallet. 80% of the team wallet, marketing wallet, development wallet amount is locked for 45 days. 20% is unlocked before launch. Which means the team would have access to 4% of total supply before launch

4% is an acceptable limit. There is some minor risk, but as they are locking most of their funds there is very little concern. **This is detailed in the minor problems section.**

Minor Problems

1. Team wallets can participate in the lottery

There is no way to control if the team's addresses are included in the lottery system or not. At the start they are excluded. But if the owner wanted to, they can send tokens to any other wallet.

Recommendations

The team should take care in making sure all team wallets are excluded from the lottery. As long as they ensure that each of their wallets cannot participate, then they will keep the lottery 100% fair.

2. Lottery array gas usage

The way the system works is that it is always adding and removing participants as needed. This is not too much of a problem as gas cost can be reduced by smartly removing and adding addresses. The concern is that when the Lottery Array is duplicated it will cause a large gas increase. In testing this was done for 1000 elements without much of a spike in gas price. But if the array grows to 100,000+ elements, then it may pose a serious issue.

Recommendations

If the array becomes too big and the gas becomes too high then it is recommended to reduce the number of entrants possible. The minimum number of tokens required to hold should also be increased to make the lottery more exclusive. These solutions will reduce the gas required to handle the array operations.

3. Moderate Amount of Tokens Held by Team

On launch the KICH Team has access to ~4% of the token’s supply out of 5 Quadrillion. This is an acceptable amount, but still carries a low risk.

KichiCoin's team has locked many tokens which provides safety for investors. After 45 days the team has access to many more tokens.

Recommendations

In a perfect world they would have very few tokens accessible, but every project has some inherent risk. KICH has an acceptable amount that the team can use at the start for marketing, development, and team usage.

Moderate Problems

n/a - No moderate problems were found.

Recommendations

n/a

Major Problems

n/a - No major problems were found.

Recommendations

n/a

Comments

KICH is unique in how it handles its lottery system. Most tokens rely on off-chain mechanisms to determine lottery winners. This is all on chain and utilizes Chainlink VRF to prove its fairness. For that reason, it should be considered an advancement in lottery-based cryptocurrencies.

Disclosure

The developer of this token and the auditor of this token are the same. Yoshiko Shinonome did both. There is a conflict of interest, but the audit was done in as unbiased of a manner as possible.

ABOUT CATNIP

CatNIP (NIP) is the base cryptocurrency of an NFT-based, decentralized ecosystem on the Binance Smart Chain. The NIP Smart Contract is governed by a Proxy, which is governed by a 72-hour timelock, which is governed by a multisig gnosis safe.

NIP is used for minting game NFTs and is planned to serve a wide variety of purposes such as audit insurance, crypto ETFs, cryptocurrency skill certifications, a NIP NFT marketplace, bridging to other DeFi networks, DeFi partnerships, and more.

Thank you using CatNIP Audits!

-CatNIP Team