CFG NINJA AUDITS

Security Assessment

Spiky Inu Token

August 18, 2022

Table of Contents

Audit Summary

This report has been prepared for Spiky Inu Token on the Binance Smart Chain network. CFGNINJA provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.

Project Overview

Token Summary

Parameter	Result
Address	0x71a06A45e8952470366022 29A9113169cFdB862d
Name	Spiky Inu
Token Tracker	Spiky Inu (SPK)
Decimals	9
Supply	1,000,000,000,000
Platform	Binance Smart Chain
compiler	v0.8.7+commit.e28d00a7
Contract Name	SpikyInu
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/address/0x71a06a45e895247036602229a9113169cfdb862d#code
Payment Tx	0x41a5105616759bd4e0487fa9750effef30bec95dc23e926ce9a9715581f78f49

Project Overview

Risk Analysis Summary

Parameter	Result
Buy Tax	10%
Sale Tax	10%
Is honeypot?	Clean
Can edit tax?	Yes
Is anti whale?	No
Is blacklisted?	No
Is whitelisted?	No
Holders	Clean
Security Score	95/100
Auditor Score	92/100
Confidence Level	Pass

The following quick summary has been added to the project overview, however there are more details about the audit and their results please read every details.

Main Contract Assessed Contract Name

Name	Contract	Live
Spiky Inu	0x71a06A45e8952470366022 29A9113169cFdB862d	Yes

TestNet Contract Assessed Contract Name

Name	Contract	Live
Spiky Inu	0x314f6a2b3aFC4Fa4611DF9C0AB6396ec0164A78f	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
SpikyInu	88b05d934dea5f35459cb6153c8d88e37d189ff5	SpikyInu.sol

Mint Check

The Project Owners of Spiky Inu does not have a mint function in the contract, owner cannot mint tokens after initial deploy

. .

The Project has a Total Supply of 1,000,000,000,000 and cannot mint any more than the Max Supply.

.

Mint Notes:

Auditor Notes: A Mint Function was not found during the code review

Project Owner Notes:

Owner can't mint new coins

Fees Check

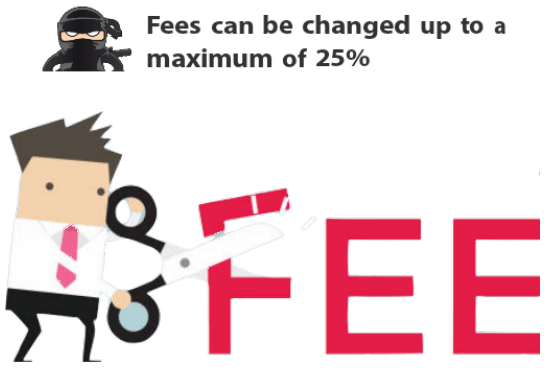
The Project Owners of Spiky Inu does not have the ability to set fees higher than 25% .

Team May have fees defined, however they dont have the ability to set those fees higher than 25%.

Tax Fee Notes:

Auditor Notes: Contract currently have 10% tax and can be increased up to 20%

Project Owner Notes: .



Blacklist Check

The Project Owners of Spiky Inu does not have a blacklist function their contract.

The Project allow owners to transfer their tokens without any restrictions.

Token owner cannot blacklist the contract: Malicious or compromised owners can trap contracts relying on tokens with a blacklist.

Blacklist Notes:

Auditor Notes: Contract does not have a blacklist function presented, auditor reviewed the contract and this was not found.

Project Owner Notes: .

MaxTx Check

The Project Owners of Spiky Inu can set max tx amount.

The ability to set MaxTx can be used as bad actor, this can limit the ability of investors to sale their tokens at any given time if is set too low..

We recommend the project to set MaxTx to Total Supply or simiar to avoid swap or transfer from failures

MaxTX Notes:

Auditor Notes: Customer have the following limits, maxBuyLimit, maxSaleLimit, maxWalletLimit, 10000000000000000000 uint256

Project Owner Notes:

Pause Trade Check

The Project Onwers of Spiky Inu Owner can pause trading but he can't move tokens (Owner can't pause trading)

The Team has done a great job to avoid stop trading, and investors has the ability to trade at any given time without any problems

Pause Trade Notes:

Auditor Notes: Not found a value to stop, however there is a start trade.

Project Owner Notes:

Contract Ownership

The contract ownership of Spiky Inu is not currently renounced. The ownership of the contract grants special powers to the protocol creators, making them the sole addresses that can call sensible ownable functions that may alter the state of the protocol.

The current owner is the address 0xa7a31554c91ed3ed696c83762f24c43c523d0ee1 which can be viewed from:
HERE

The owner wallet has the power to call the functions displayed on the priviliged functions chart below, if the owner wallet is compromised this privileges could be exploited.

We recommend the team to renounce ownership at the right timing if possible, or gradually migrate to a timelock with governing functionalities in respect of transparency and safety considerations.

We recommend the team to use a Multisignature Wallet if contract is not going to be renounced, this will give the ability to the team to have more control over the contract.

Liquidity Ownership

The token does not have liquidity at the moment of the audit, block 20548845

If liquidity is unlocked, then the token developers can do what is infamously known as 'rugpull'. Once investors start buying token from the exchange, the liquidity pool will accumulate more and more coins of established value (e.g., ETH or BNB or Tether). This is because investors are basically sending these tokens of value to the exchange, to get the new token. Developers can withdraw this liquidity from the exchange, cash in all the value and run off with it. Liquidity is locked by renouncing the ownership of liquidity pool (LP) tokens for a fixed time period, by sending them to a time-lock smart contract. Without ownership of LP tokens, developers cannot get liquidity pool funds back. This provides confidence to the investors that the token developers will not run away with the liquidity money. It is now a standard practice that all token developers follow, and this is what really differentiates a scam coin from a real one.

Read More

KYC Information

The Project Onwers of Spiky Inu has provided KYC Documentation.

KYC Certificated can be found on the Following:
KYC Data

KYC Information Notes:

Auditor Notes: Asked project owner about KYC, Project owner passed KYC with PinkSale.

Project Owner Notes:

Smart Contract Vulnerability Checks

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	SpikyInu.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	SpikyInu.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	SpikyInu.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	SpikyInu.sol	L: 6 C: 0
SWC-104	Pass	Unchecked Call Return Value.	SpikyInu.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	SpikyInu.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	SpikyInu.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	SpikyInu.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	SpikyInu.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	SpikyInu.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	SpikyInu.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	SpikyInu.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	SpikyInu.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-113	Pass	Multiple calls are executed in the same transaction.	SpikyInu.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	SpikyInu.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	SpikyInu.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	SpikyInu.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	SpikyInu.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	SpikyInu.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	SpikyInu.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randonmness.	SpikyInu.sol	L: 335 C: 24,L: 693 C: 12
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	SpikyInu.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	SpikyInu.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	SpikyInu.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	SpikyInu.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	SpikyInu.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	SpikyInu.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	SpikyInu.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-128	Pass	DoS With Block Gas Limit.	SpikyInu.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	SpikyInu.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	SpikyInu.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	SpikyInu.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	SpikyInu.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	SpikyInu.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	SpikyInu.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	SpikyInu.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	SpikyInu.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry standard security scanning tool

Smart Contract Vulnerability Details

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Smart Contract Vulnerability Details

SWC-120 - Weak Sources of Randomness from Chain Attributes

CWE-330: Use of Insufficiently Random Values

Description:

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

Remediation:

Using commitment scheme, e.g. RANDAO. Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles. Using Bitcoin block hashes, as they are more expensive to mine.

References:

How can I securely generate a random number in my smart contract?)

When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.

SWC Information Notes:

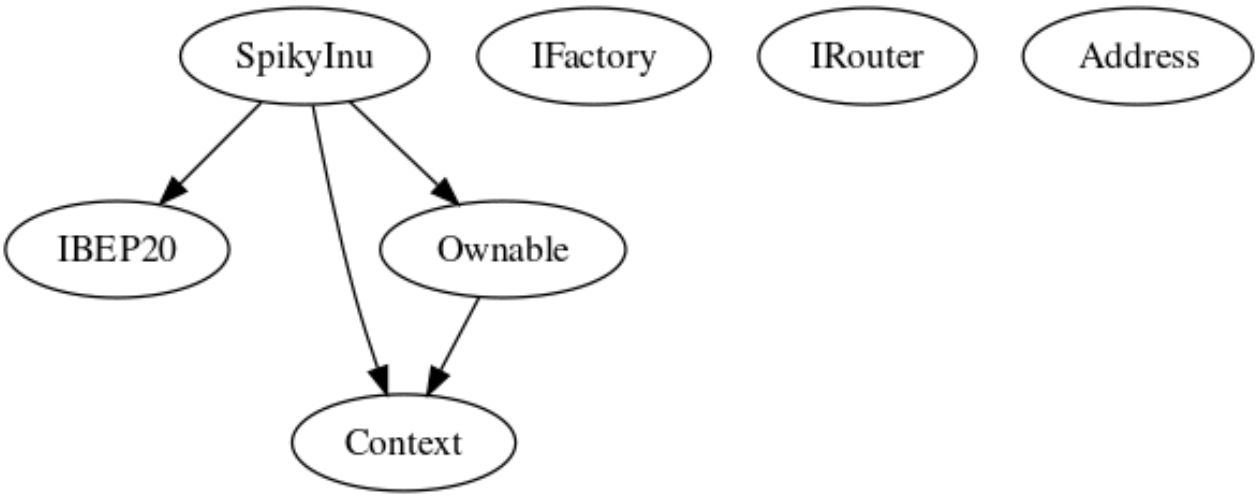
Auditor Notes:

No Vulnerabilities where found during the security scan.

Project Owner Notes:

Call Graph and Inheritance

The contract for Spiky Inu has the following call graph structure



Priviliged Functions (onlyOwner)

Function Name	Parameters	Visibility
renounceOwnership	none	public
transferOwnership	address newOwner	public
EnableTrading		external
updatedeadline		external
excludeFromReward		external
includeInReward		external
includeInFee	account (address)	external
excludeFromFee	account (address)	external
setTaxes		external

Function Name	Parameters	Visibility
setSellTaxes		external
bulkExcludeFee		external
updateMarketingWallet		external
updateDevWallet		external
updateOpsWallet		external
updateCharityWallet		external
updateStakingWallet		external
updateCooldown		external
updateSwapTokensAtAmount		external
updateSwapEnabled		external

Function Name	Parameters	Visibility
updateMaxTxLimit		external
updateRouterAndPair		external
rescueBNB		external
rescueAnyBEP20Tokens		external

Assessment Results

- Deployer Wallet is as follow https://bscscan.com/address /0xa7a31554c91ed3ed696c83762f24c43c523d0ee1
- Initial Deployment money was received by https://bscscan.com/address/0x0160b93b432c8bb74cf11de7f904e27b6fc3eb6e
- Owner can charge fees up to 20%.
- Owner can set MaxTX.
- Owner can't pause trading.
- Reviewed code and everything looks in terms of best practices.

Audit Passed

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/spikyinu	Pass
Instagram	https://instagram.com/spikyinu	Pass
Website	http://www.spikyinu.com	Pass
Telegram	https://t.me/spikyinu	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes: No other social media

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
Major	risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Minor	risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
Informational	errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	0	0	0
Major	2	0	0
Medium	1	0	0
Minor	0	0	0
Informational	0	0	0
Total	3	3	0

SPK-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	Medium	SpikyInu.sol: 806,13	Pending

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset. The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- swapExactTokensForETHSupportingFeeOnTransferTokens()
- addLiquidityETH()

Remediation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Referrences:

What Are Sandwich Attacks in DeFi — and How Can You Avoid Them?.

SPK-06 | Conformance with Solidity Naming Conventions.

Category	Severity	Location	Status
Coding Style	Major	SpikyInu.sol: 338,14	Pending

Description

Solidity defines a naming convention that should be followed. Rule exceptions: Allow constant variable name/symbol/decimals to be lowercase. Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
updatedeadline
```

Remediation

Follow the Solidity naming convention.

https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-convention

SPK-07 | State Variables could be Declared Constant.

Category	Severity	Location	Status
Coding Style	Major	SpikyInu.sol: 155,20	Pending

Description

Constant state variables should be declared constant to save gas.

```
deadWallet
```

Remediation

Add the constant attribute to state variables that never changes.

https://docs.soliditylang.org/en/latest/contracts.html#constant-state-variables

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

CFGNINJA has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and CFGNINJA is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will CFGNINJA or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

The assessment services provided by CFGNINJA is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

CFG NINJA AUDITS