# CFG NINJA AUDITS

## Security Assessment

# Snek Inu Token

October 4, 2022

# Table of Contents





# Audit Summary

This report has been prepared for Snek 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 | 0x002C0d606aA9671137cAD6253b05F28123697384 |
| Name | Snek Inu |
| Token Tracker | Snek Inu (SKINU) |
| Decimals | 18 |
| Supply | 1,000,000,000,000,000 |
| Platform | Binance Smart Chain |
| compiler | v0.8.17+commit.8df45f5f |
| Contract Name | SnekInu |
| Optimization | Yes with 200 runs |
| LicenseType | MIT |
| Language | Solidity |
| Codebase | https://bscscan.com/address/0x002c0d606aa9671137cad6253b05f28123697384#code |
| Payment Tx | 0x2b8a1aa8fa38036c8a5dac33880c16690a9496b641a6de3fde9b7c6630499140 |



# Project Overview

## Risk Analysis Summary

| Parameter | Result |
|---|---|
| Buy Tax | 5% |
| Sale Tax | 5% |
| Is honeypot? | Clean |
| Can edit tax? | Yes |
| Is anti whale? | No |
| Is blacklisted? | Yes |
| Is whitelisted? | No |
| Holders | Clean |
| Security Score | 98/100 |
| Auditor Score | 98/100 |
| 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 |
|---|---|---|
| Snek Inu | 0x002C0d606aA9671137cAD6253b05F28123697384 | Yes |

## TestNet Contract Assessed Contract Name

| Name | Contract | Live |
|---|---|---|
| Snek Inu | 0x53f03B4C4CD583fF9ad2D03beA40dA66dD67D8C2 | Yes |

## Solidity Code Provided

| SolID | File Sha-1 | FileName |
|---|---|---|
| Snekinu | 5289510eece43e623b05acca9cdbcbc4bb032bce | Snekinu.sol |



# Mint Check

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

. .

## The Project has a Total Supply of 1,000,000,000,000,000 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

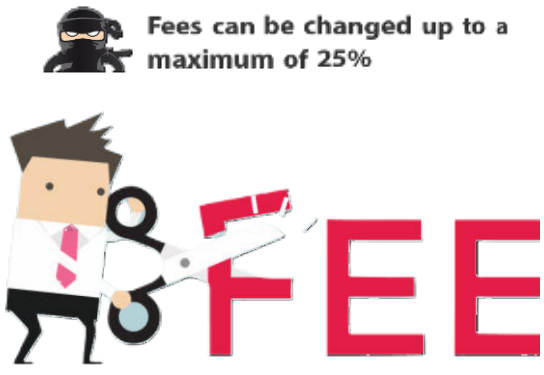
## The Project Owners of Snek Inu does not have the ability to set fees higher than 25% .

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

**Tax Fee Notes:**

**Auditor Notes: Taxes are private**

**Project Owner Notes: .**



# Blacklist Check

## The Project Onwers of Snek Inu has the ability to Blacklist holders from transferring their tokens.

## We Recommend the team to be careful with a blacklist function as this can basically prevent ah holder from buying/selling/transferring their assets. Malicious or compromised owners can trap contracts relying on tokens with a blacklist.

**Blacklist Notes:**

**Auditor Notes:**

**Project Owner Notes: .**



# MaxTx Check

## The Project Onwers of Snek Inu does not has the ability to set max tx amount

## The Team allow any investors to swap, transfer or sale their total amount if needed.

**MaxTX Notes:**

**Auditor Notes: '**

**Project Owner Notes:**

Project Has No MaxTX



# Pause Trade Check

## The Project Owners of Snek Inu don't have the ability to stop or 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 Snek 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 0x8d8a46de5baa4c7f9d531c63412c9af8e2d3ae1a 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 21872825

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 Owners of Snek Inu is not KYC. .

**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.**

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



| ID | Severity | Name | File | location |
|---|---|---|---|---|
| SWC-113 | Pass | Multiple calls are executed in the same transaction. | Snekinu.sol | L: 0 C: 0 |
| SWC-114 | Pass | Transaction Order Dependence. | Snekinu.sol | L: 0 C: 0 |
| SWC-115 | Pass | Authorization through tx.origin. | Snekinu.sol | L: 0 C: 0 |
| SWC-116 | Pass | A control flow decision is made based on The block.timestamp environment variable. | Snekinu.sol | L: 0 C: 0 |
| SWC-117 | Pass | Signature Malleability. | Snekinu.sol | L: 0 C: 0 |
| SWC-118 | Pass | Incorrect Constructor Name. | Snekinu.sol | L: 0 C: 0 |
| SWC-119 | Pass | Shadowing State Variables. | Snekinu.sol | L: 0 C: 0 |
| SWC-120 | Pass | Potential use of block.number as source of randonmness. | Snekinu.sol | L: 607 C: 12,L: 774 C: 24 |
| SWC-121 | Pass | Missing Protection against Signature Replay Attacks. | Snekinu.sol | L: 0 C: 0 |
| SWC-122 | Pass | Lack of Proper Signature Verification. | Snekinu.sol | L: 0 C: 0 |
| SWC-123 | Pass | Requirement Violation. | Snekinu.sol | L: 0 C: 0 |
| SWC-124 | Pass | Write to Arbitrary Storage Location. | Snekinu.sol | L: 0 C: 0 |
| SWC-125 | Pass | Incorrect Inheritance Order. | Snekinu.sol | L: 0 C: 0 |
| SWC-126 | Pass | Insufficient Gas Griefing. | Snekinu.sol | L: 0 C: 0 |
| SWC-127 | Pass | Arbitrary Jump with Function Type Variable. | Snekinu.sol | L: 0 C: 0 |



| ID | Severity | Name | File | location |
|---|---|---|---|---|
| SWC-128 | Pass | DoS With Block Gas Limit. | Snekinu.sol | L: 0 C: 0 |
| SWC-129 | Pass | Typographical Error. | Snekinu.sol | L: 0 C: 0 |
| SWC-130 | Pass | Right-To-Left-Override control character (U +202E). | Snekinu.sol | L: 0 C: 0 |
| SWC-131 | Pass | Presence of unused variables. | Snekinu.sol | L: 0 C: 0 |
| SWC-132 | Pass | Unexpected Ether balance. | Snekinu.sol | L: 0 C: 0 |
| SWC-133 | Pass | Hash Collisions with Multiple Variable Length Arguments. | Snekinu.sol | L: 0 C: 0 |
| SWC-134 | Pass | Message call with hardcoded gas amount. | Snekinu.sol | L: 0 C: 0 |
| SWC-135 | Pass | Code With No Effects (Irrelevant/Dead Code). | Snekinu.sol | L: 0 C: 0 |
| SWC-136 | Pass | Unencrypted Private Data On-Chain. | Snekinu.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-108 - State Variable Default Visibility

## CWE-710: Improper Adherence to Coding Standards

### Description:

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

### Remediation:

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

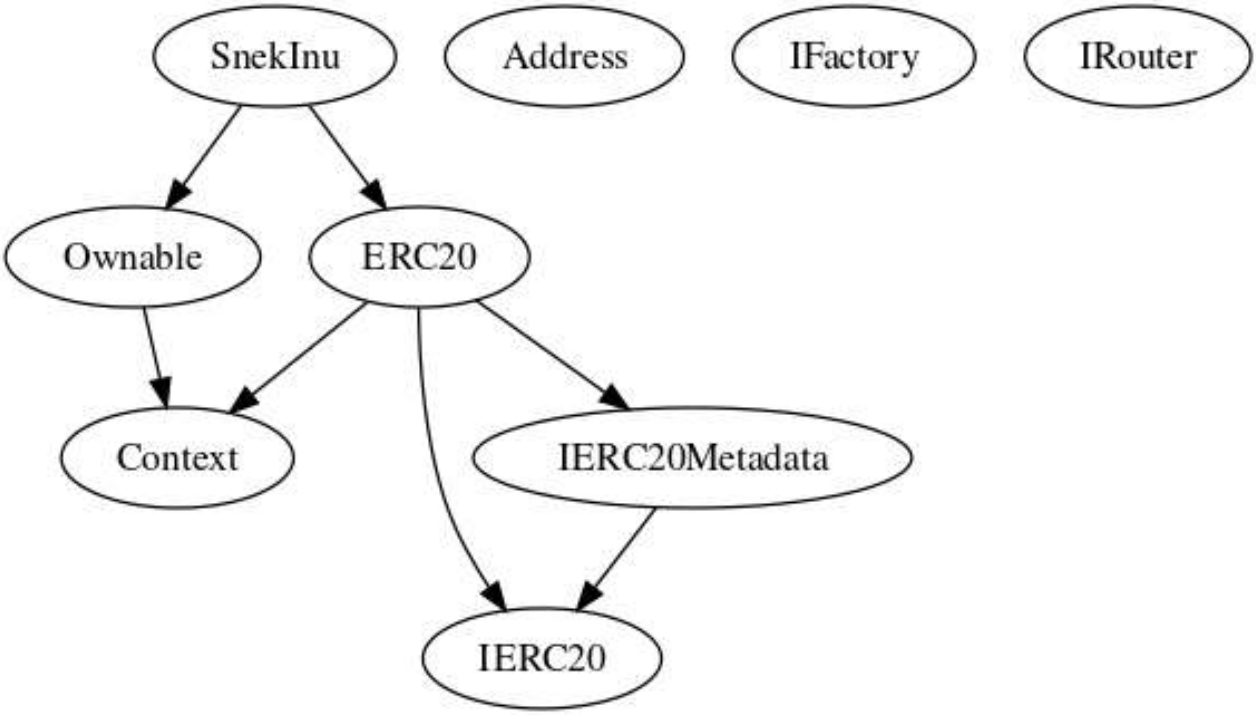
### References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables



# Call Graph and Inheritance

## The contract for Snek Inu has the following call graph structure



# Priviliged Functions (onlyOwner)

| Function Name | Parameters | Visibility |
|---|---|---|
| renounceOwnership | none | public |
| transferOwnership | none | public |
| updateMarketingWallet | | external |
| updateDevWallet | | external |
| updateBuyTaxes | | external |
| updateSellTaxes | | external |
| enableTrading | | external |
| updateLiquidityProvide | | external |
| updateLiquidityTreshhold | | external |
| updateRouterAndPair | | external |
| updatedeadline | | external |

| Function Name | Parameters | Visibility |
| --- | --- | --- |
| updateProjectdevWallet | | external |
| updateCooldown | | external |
| updateIsBlacklisted | | external |
| bulkIsBlacklisted | | external |
| updateExemptFee | | external |
| bulkExemptFee | | external |
| updateMaxBuyTxLimit | | external |
| updateMaxSellTxLimit | | external |
| updateMaxWalletlimit | | external |
| updateMaxTxLimit | | external |
| rescueBNB | | external |
| rescueBSC20 | | external |



## Assessment Results

- Owner can't charge fees up to 25%.
- Owner can't set max tx amount.
- Owner can't pause trading.
- No high-risk Exploits/Vulnerabilities Were Found in the Source Code.
- Contract has been reviewed and tested, there are no sign of issues within the contract and it came back clean.
- Contract has been developed by Anoop and follows the coding best practices, we have fully tested the code and its functionalities.

## Audit Passed

# Social Media Checks

| Social Media | URL | Result |
|---|---|---|
| Twitter | https://twitter.com/SnekInu | Pass |
| Instagram | | Fail |
| Website | https://snekinu.com/ | Pass |
| Telegram | https://t.me/SnekInu | 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 Socials**



# 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 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 |
| Minor | 0 | 0 | 0 |
| Informational | 0 | 0 | 0 |
| Total | 1 | 1 | 0 |



# SKINU-01 | Potential Sandwich Attacks.

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Security | Medium | Snekinu.sol: 1370,20 | 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?.



# SKINU-05 | Missing Event Emission.

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Volatile Code | Major | Snekinu.sol: 0,0 | Pending |

## Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.The linked code does not create an event for the transfer.

## Remediation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.



# 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.