CFG NINJA AUDITS

Security Assessment
Snoopy Token

June 28, 2023

Audit Status: Pass

Audit Edition: Advanced

Risk Analysis

Classifications of Manual Risk Results

Classification	Description
🔴 Critical	Danger or Potential Problems.
🟠 High	Be Careful or Fail test.
🟢 Low	Pass, Not-Detected or Safe Item.
ℹ️ Informational	Function Detected

Manual Code Review Risk Results

Contract Priviledge	Description
🟢 Buy Tax	0%
🟢 Sale Tax	0%
🟢 Cannot Sale	Pass
🟢 Cannot Sale	Pass
🟢 Max Tax	0%
🟢 Modify Tax	No
🟢 Fee Check	Pass
🟢 Is Honeypot?	Not Detected
🟢 Trading Cooldown	Not Detected
🔴 Can Pause Trade?	Detected,Owner need to enable trade.

Contract Priviledge	Description
🔴 Pause Transfer?	Detected, Owner need to Enable Trade.
🟠 Max Tx?	Fail
🟠 Is Anti Whale?	Detected
🟠 Is Anti Bot?	Detected
🟢 Is Blacklist?	Not Detected
🟢 Blacklist Check	Pass
🟢 is Whitelist?	Not Detected
🟢 Can Mint?	Pass
🟢 Is Proxy?	Not Detected
🟢 Can Take Ownership?	Not Detected
🟢 Hidden Owner?	Not Detected
🔵 Owner	0x51afdbc5972506d8a6e7455d4e32ec4fdae6e603
🟢 Self Destruct?	Not Detected
🟢 External Call?	Not Detected
🟢 Other?	Not Detected
🟢 Holders	1
🟠 Auditor Confidence	low

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

Project Overview

Token Summary

Parameter	Result
Address	0x42451D82cF0cCd3D1d32b0D9F8F5021AB3c914E8
Name	Snoopy
Token Tracker	Snoopy (S)
Decimals	18
Supply	1,000,000,000,000
Platform	Binance Smart Chain
compiler	v0.8.10+commit.fc410830
Contract Name	Snoopy
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://bscscan.com/token/0x42451D82cF0cCd3D1d32b0D9F8F5021AB3c914E8#code
Payment Tx	Corporate

Project Overview

Simulation Summary

Parameter	Result
Transfer From Owner	Pass
Transfer From Holder	Pass
Add Liquidity	Pass
RemoveLiquidity	Pass
Buy from Owner	Pass
Buy from Holder	Pass
Sale from Owner	Pass
Sale from Holder	Pass
Remove Liquidity	Pass
SwapAndLiquify	Pass
SwapAndSale w/Fee	Pass
SwapAndSale TX	
SwapAndSaleNoFee	Pass
SwapAndSale No/Fee TX	
ExcludeFromFees	Pass
LaunchPad	PinkSale

Parameter	Result
Pool Creation	Pass
Pool Creation TX	
Pool Finalize	Pass
Pool Finalize TX	
Enable	Pass

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

Main Contract Assessed
Contract Name

Name	Contract	Live
Snoopy	0x42451D82cF0cCd3D1d32b0D9F8F5021AB3c914E8	Yes

TestNet Contract Assessed
Contract Name

Name	Contract	Live
Snoopy	0x8f8C3d6F67ce626C63560DCeB928f27Ba087329A	Yes

Solidity Code Provided

SolID	File Sha-1	FileName
Snoopy	b76da9cd983ecd22f15740b4d5bd2253caea601f	Snoopy.sol
Snoopy		
Snoopy		
Snoopy		

Call Graph

The contract for Snoopy has the following call graph structure.
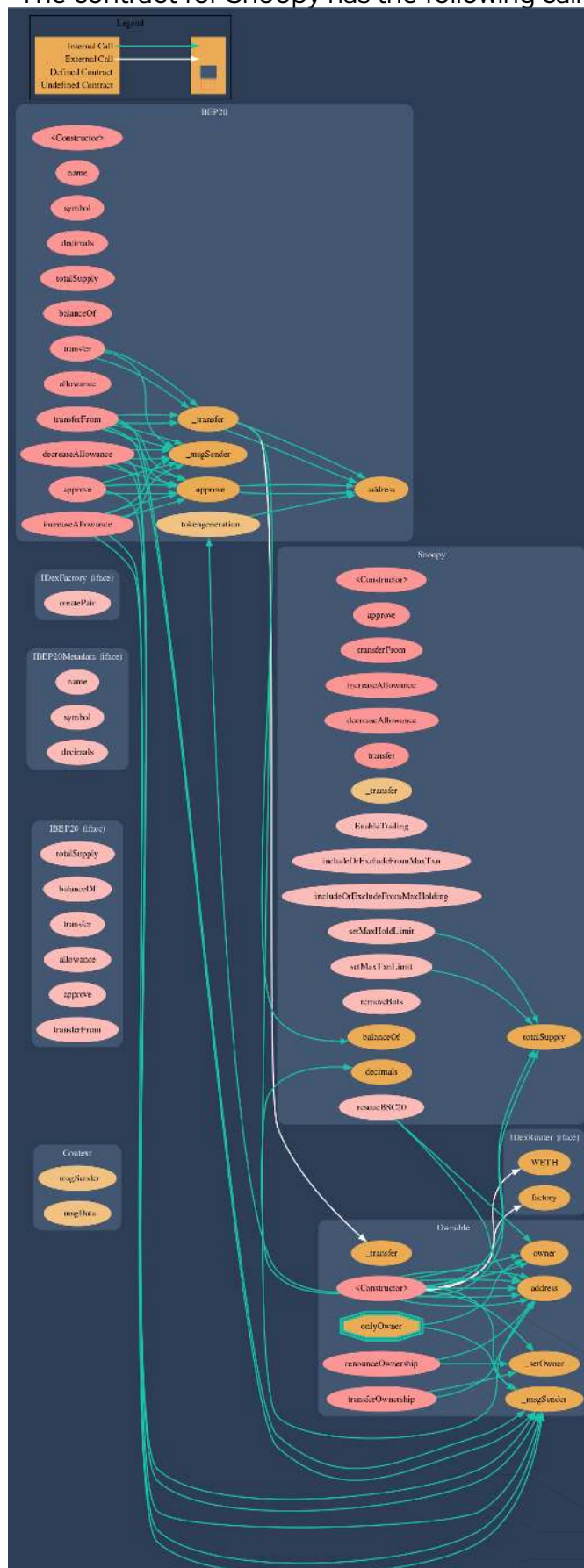
Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	Snoopy.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	Snoopy.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	Snoopy.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	Snoopy.sol	L: 5 C: 0
SWC-104	Pass	Unchecked Call Return Value.	Snoopy.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	Snoopy.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	Snoopy.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	Snoopy.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	Snoopy.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	Snoopy.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	Snoopy.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	Snoopy.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	Snoopy.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	Snoopy.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	Snoopy.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	Snoopy.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	Snoopy.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	Snoopy.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	Snoopy.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	Snoopy.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randonmness.	Snoopy.sol	L: 395 C: 24
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	Snoopy.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	Snoopy.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	Snoopy.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	Snoopy.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	Snoopy.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	Snoopy.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	Snoopy.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	Snoopy.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	Snoopy.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	Snoopy.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	Snoopy.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	Snoopy.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	Snoopy.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	Snoopy.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	Snoopy.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	Snoopy.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

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.

Inheritance

The contract for Snoopy has the following inheritance structure.
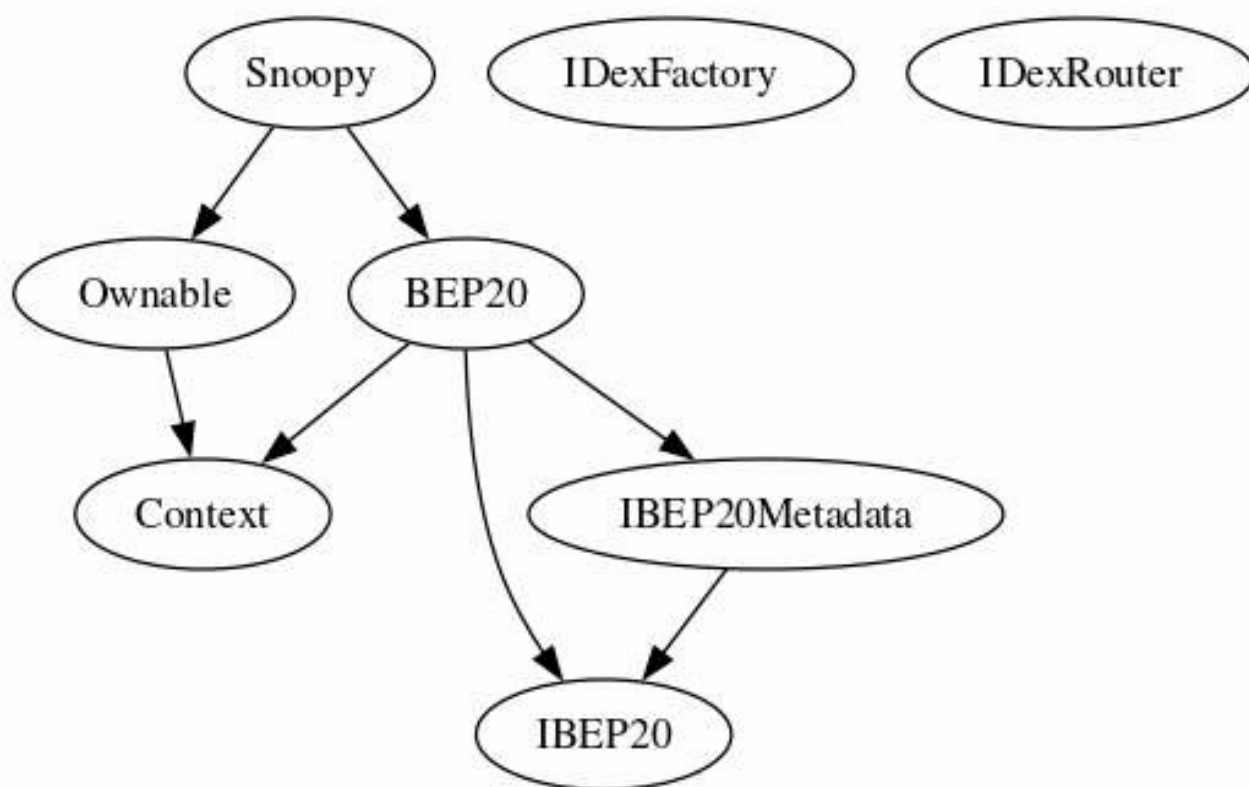
Privileged Functions (onlyOwner)

Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
renounceOwnership		Public
transferOwnership	address newOwner	Public
claimStuckTokens		External
excludeFromFees		External
changeMarketingWallet		Public
disableTax		public
setBuyTax		External
setSellTax		External

Smart Contract Advance Checks

ID	Severity	Name	Result	Status
S-01	Low	Potential Sandwich Attacks.	Pass	Not-Found
S-02	Informational	Function Visibility Optimization	Pass	Not Detected
S-03	Low	Lack of Input Validation.	Fail	Detected
S-04	High	Centralized Risk In addLiquidity.	Pass	Not-Found
S-05	Low	Missing Event Emission.	Fail	Detected
S-06	Low	Conformance with Solidity Naming Conventions.	Pass	Not-Found
S-07	Low	State Variables could be Declared Constant.	Pass	Not-Found
S-08	Low	Dead Code Elimination.	Pass	Not-Found
S-09	High	Third Party Dependencies.	Pass	Not-Found
S-10	High	Initial Token Distribution.	Pass	Not-Found
S-11	High		Pass	Not Detected
S-12	High	Centralization Risks In The X Role	Pass	Not-Found
S-13	Informational	Extra Gas Cost For User..	Pass	Not Detected
S-14	Medium	Unnecessary Use Of SafeMath	Pass	Not Detected
S-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Not-Found
S-16	Medium	Taxes can be up to 100%	Pass	Not Detected

ID	Severity	Name	Result	Status
S-17	Informational	Highly Permissive Role Access.,`	Pass	Not-Found
S-18	Critical	Stop Transactions by using Enable Trade.	Fail	Detected

S-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	🟢 Low	Snoopy.sol: L: 399 C: 14	🗎 Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the .

Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
 require(receiver != address(0), "Receiver is the zero address");
...
...
require(value X limitation, "Your not able to do this function");
...
```

We also recommend customer to review the following function that is missing a required validation. .

S-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	🟢 Low	Snoopy.sol: L: 399 C: 14	📄Detected

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.

S-18 | Stop Transactions by using Enable Trade.

Category	Severity	Location	Status
Logical Issue	🔴 Critical	Snoopy.sol: L: 392 C: 0	🗎 Detected

Description

Enable Trade is presend on the following contract and when combined with Exclude from fees it can be considered a whitelist process, this will allow anyone to trade before others and can represent and issue for the holders.

Remediation

We recommend the project owner to carefully review this function and avoid problems when performing both actions.

Project Action

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.
🟠 High	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
🟢 Low	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.
ℹ️ Informational	Errors are often recommended to improve the code's style 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	1	0	0
🟠 High	0	0	0
🟡 Medium	0	0	0
🟢 Low	2	0	0
ℹ️ Informational	0	0	0
Total	3	0	0

Social Media Checks

Social Media	URL	Result
Twitter		Fail
Other		Fail
Website		Fail
Telegram		Fail

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:

Assessment Results

Score Results

Review	Score
Overall Score	85/100
Auditor Score	80/100
Review by Section	Score
Manual Scan Score	22/53
SWC Scan Score	33 /37
Advance Check Score	30 /19

The Following Score System Has been Added to this page to help understand the value of the audit, the maximun score is 100, however to attain that value the project most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 80 Points, if a project does not attain 80% is an automatic failure. Read our notes and final assessment below.

Audit Passed

Assessment Results

Important Notes:

- No issues or vulnerabilities were found at the time of the assessment.

- The owner needs to enable trade.

- The contract has antibot and anti-whale methods.

- The customer has KYC by AssureDefi.

- Please DYOR on the project.

Auditor Score =80
Audit Passed

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 security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this 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 the Project in question may pursue, and 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 entirely 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. Information is provided 'as is, and CFGNINJA is under no covenant to audited completeness, accuracy, or solidity of the contracts. 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 or actions with regards to the information provided in this audit report.

The assessment services provided by CFGNINJA are subject to dependencies and are under continuing development. You agree that your access 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 with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.