

## Security Assessment: **Conan Token**





March 28, 2024

- Audit Status: **Pass**
- Audit Edition: **Standard**
































# 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.
 Informational	Function Detected

## Manual Code Review Risk Results

Contract Privilege	Description
 Buy Tax	0%
 Sale Tax	0%
 Cannot Buy	Pass
 Cannot Sale	Pass
 Max Tax	0%
 Modify Tax	No
 Fee Check	Pass
 Is Honeypot?	Not Detected
 Trading Cooldown	Not Detected
 Can Pause Trade?	Pass
 Pause Transfer?	Not Detected
 Max Tx?	Pass
 Is Anti Whale?	Not Detected
 Is Anti Bot?	Not Detected

Contract Privilege	Description
 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	no
 Self Destruct?	Not Detected
 External Call?	Not Detected
 Other?	Not Detected
 Holders	1
 Auditor Confidence	Low Risk
 KYC Present	No
 KYC URL	

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	0xE6502CEE5B040d813d0A9D8CBA60C096a22E3eC9
Name	Conan
Token Tracker	Conan (vllc)
Decimals	18
Supply	22,101,970,000
Platform	ETHEREUM
compiler	v0.8.9+commit.e5eed63a
Contract Name	Conan
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	<a href="https://etherscan.io/address/0xE6502CEE5B040d813d0A9D8CBA60C096a22E3eC9#code">https://etherscan.io/address/0xE6502CEE5B040d813d0A9D8CBA60C096a22E3eC9#code</a>
Payment Tx	Corporate

## Main Contract Assessed Contract Name

Name	Contract	Live
Conan	0xE6502CEE5B040d813d0A9D8CBA60C096a22E3eC9	Yes

## TestNet Contract was Not Assessed

### Solidity Code Provided

SolID	File Sha-1	FileName
vllc	24e71abde848a5981ed7b77fc92f38d0c540617b	vllc.sol
vllc		
vllc		
vllc		
vllc		
vllc	undefined	

# 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	vllc.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	vllc.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	vllc.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	vllc.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	vllc.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	vllc.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	vllc.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	vllc.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	vllc.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	vllc.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	vllc.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	vllc.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	vllc.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	vllc.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	vllc.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-115	Pass	Authorization through tx.origin.	vllc.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	vllc.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	vllc.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	vllc.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	vllc.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randomness.	vllc.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	vllc.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	vllc.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	vllc.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	vllc.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	vllc.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	vllc.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	vllc.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	vllc.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	vllc.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	vllc.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	vllc.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	vllc.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	vlc.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	vlc.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	vlc.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	vlc.sol	L: 0 C: 0

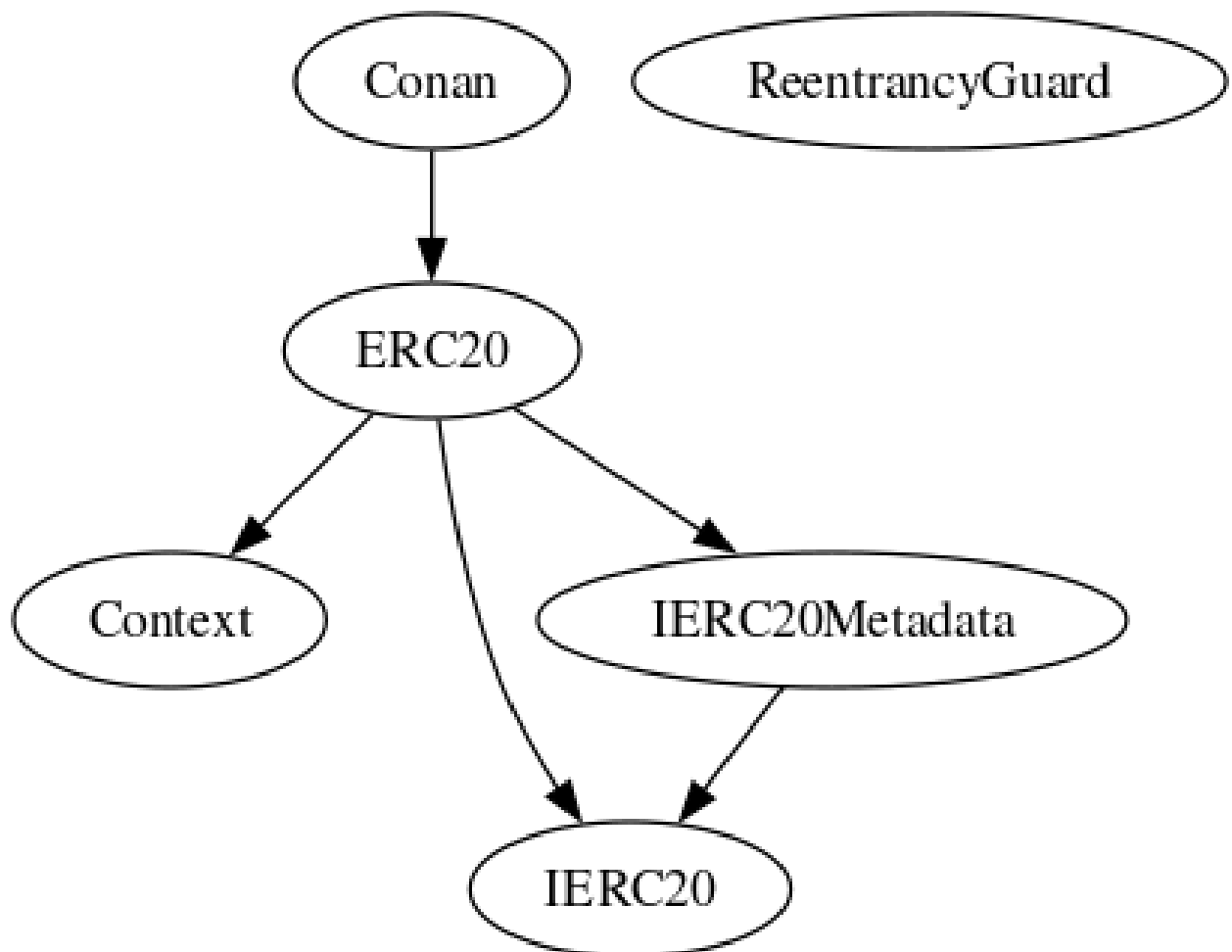
We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.





# Inheritance

The contract for Conan has the following inheritance structure.

The Project has a Total Supply of 22,101,970,000



## vllc-08 | Dead Code Elimination.

Category	Severity	Location	Status
Coding Style	 Low	vllc.sol: L: 5 C:14	 Detected

### Description

Functions that are not used in the contract, and make the code s size bigger.



ReentrancyGuard.sol

### Remediation

Remove unused functions. dead-code elimination (also known as DCE, dead-code removal, dead-code stripping, or dead-code strip) is a compiler optimization to remove code which does not affect the program results. Removing such code has several benefits: it shrinks program size, an important consideration in some contexts, and it allows the running program to avoid executing irrelevant operations, which reduces its running time. It can also enable further optimizations by simplifying program structure.

<https://docs.soliditylang.org/en/latest/cheatsheet.html>

## vllc-10 | Initial Token Distribution.

Category	Severity	Location	Status
Centralization / Privilege	 High	vllc.sol: L: 10 C: 14	 Detected

### Description

All of the Conan tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

### Remediation






We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

### Project Action






```
_mint(msg.sender, 22101970000 * 10 ** decimals());
```

# 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.
 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	0		0
 High	1	1	0
 Medium	0	0	0
 Low	1	1	0
 Informational	0	0	0
Total	2	2	0

# Social Media Checks

Social Media	URL	Result
Twitter	<a href="https://twitter.com/conantokenETH">https://twitter.com/conantokenETH</a>	Pass
Other		N/A
Website	<a href="https://www.conantoken.com">https://www.conantoken.com</a>	Pass
Telegram	<a href="https://t.me/conantokenoficial">https://t.me/conantokenoficial</a>	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:**



# Assessment Results

## Score Results

Review	Score
Overall Score	95/100
Auditor Score	85/100
Review by Section	Score
Manual Scan Score	24
SWC Scan Score	37
Advance Check Score	34

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 84 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below.

## Audit Passed



## Assessment Results

### Important Notes:

- **ReentrancyGuard:** Included but not utilized in any function. Consider applying it to functions that may be vulnerable to reentrancy attacks or remove it if unnecessary to save deployment costs.␣
- **Centralization Concern:** The entire token supply is minted to the contract creator. This could pose a risk if the deployer's account is compromised. Consider implementing a decentralized minting process or a time-locked release.␣
- **Initial Supply:** The initial supply is hardcoded and very large. Ensure that this aligns with the project's tokenomics and is not susceptible to manipulation.␣
- **Gas Optimization:** The use of require statements with string reverts is more expensive than using custom errors. Consider replacing them for gas savings.␣
- **Token Supply Management:** The contract lacks public or external functions to mint or burn tokens, which limits the ability to manage the token supply post-deployment.␣
- **Inheritance:** The contract correctly inherits from OpenZeppelin's ERC20, which is a well-audited and secure implementation of the ERC20 standard.␣
- **Compliance:** The contract does not include a permit function for gasless transactions, which could be a feature to consider for user convenience.␣

- Overflow/Underflow Protection: Solidity ^0.8.0 inherently protects against overflow and underflow, so this is not a concern.␣
- Contract Size: The contract size appears manageable, but be cautious of adding more functionality that could approach the maximum contract size limit.␣
- Missing Events: All standard ERC20 events are correctly implemented; no additional events are necessary unless new functionalities are introduced.␣
- Decimals: The contract uses a default of 18 decimals, which is standard for ERC20 tokens. Ensure this is suitable for the token's intended use.␣
- Code Clarity: The code is clear and follows Solidity style guidelines, which is good for maintainability and readability.␣
- Testing: There is no evidence of testing within the provided code. Ensure comprehensive tests are written and passed to cover all functionalities.␣
- Documentation: The contract lacks NatSpec comments for functions. Consider adding detailed comments to improve code understanding and maintainability.␣
- Licensing: The contract correctly specifies the MIT license.␣
- Custom Logic: No custom logic or hooks (`_beforeTokenTransfer` and `_afterTokenTransfer`) are implemented. If the token requires specific behaviors (like a tax or freeze on transfers), these should be added.␣
- External Calls: The contract does not make external calls, which minimizes the risk of interacting with malicious contracts.␣



- State Variables: State variables are appropriately set to private, reducing the risk of unauthorized access.
- In conclusion, the contract is a straightforward implementation of an ERC20 token with a focus on using secure, well-established code

**Auditor Score =85**  
**Audit Passed**



# Appendix

## Finding Categories

### Centralization / Privilege

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

### Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM 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 how `block.timestamp` works.

### Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being 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 may result in a vulnerability.

### Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the 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 requirements on the input variables than a setter function.

### Coding Best Practices

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

# Disclaimer

Assure Defi 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 advocacy 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 Assure Defi is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Assure Defi 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 Assure Defi 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.

