INFINITY

BLOCKCHAIN SOLUTIONS

Smart contract Audit
Full Detailed Report

Website:
https://infinityblockchainsolutions.com

Email:
contact@infinityblockchainsolutions.com

Telegram:
http://t.me/InfinityDev_77

Table of contents

Table of Contents

Introduction ***4***

Project Background ***4***

Audit scope ***4***

Claimed Smart Contract Features ***5***

Audit Summary ***6***

Risk Analysis ***6***

Documentation ***8***

Use of Dependencies ***8***

AS-IS overview ***9***

Severity Definitions ***10***

Audit Findings ***11***

Critical Severity 11

High Severity 11

Medium 11

Low 11

Very Low / Informational / Best practices: 11

Centralization ***11***

Conclusion ***12***

Our Methodology ***12***

Disclaimers ***14***

Infinity Blockchain Solutions.io Disclaimer 14

Technical Disclaimer 14

Appendix ***15***

Slither Results Log **16**

Solidity Static Analysis **17**

Solhint Linter **19**

Email: contact@infinityblockchainsolutions.com

THIS DOCUMENT IS A SECURITY AUDIT REPORT THAT MAY CONTAIN SENSITIVE AND CONFIDENTIAL INFORMATION. THIS REPORT IS FOR INTERNAL USE AND SHOULD ONLY BE DISCLOSED TO THE PUBLIC AFTER THE IDENTIFIED ISSUES HAVE BEEN RESOLVED.

Email: contact@infinityblockchainsolutions.com

Introduction

Infinity Blockchain Solutions was contracted by the ARTEMIS team to perform the Security audit of the ARTEMIS smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 17th, 2024.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

- ARTEMIS is an ERC20-based standard smart contract deployed on the Ethereum blockchain.

Audit scope

Name	**Code Review and Security Analysis Report for Smart Contract for ARTEMIS**
Platform	**Ethereum**
Language	**Solidity**
File	Artemis.sol
Contract Address	0x97D1d74B35b2ef5f0251C3E49361a74e68222122
Audit Date	June 17th, 2024

Email: contact@infinityblockchainsolutions.com

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: • Name: ARTEMIS • Symbol: ARTMS • Decimals: 18 • Total Supply: 100,000,000,000	**YES, this is valid.**
Ownership control: • Current owner can transfer the ownership. • Owner can renounce ownership.	**YES, this is valid. We advise renouncing ownership to make the contract fully decentralized.**

Email: contact@infinityblockchainsolutions.com

Audit Summary

According to the standard audit assessment, Customer`s solidity based smart contracts are **“Secured”**. **Also, these contracts contain owner control, which does not make them fully decentralized.**

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.

All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section.

We found 0 critical, 0 high, 0 medium and 0 low and 1 very low-level/Best-practice issues.

Investors Advice: A technical audit of the smart contract does not ensure the ethical nature of the project. Owner-controlled functions should be executed responsibly by the owner. It is strongly advised that all investors and users conduct thorough due diligence before investing in the project.

Risk Analysis

Category	Result
Buy Tax	None
Sell Tax	None
Cannot Buy	False
Cannot Sell	False
Max Tax	None
Modify Tax	None
Fee Check	N/A

Email: contact@infinityblockchainsolutions.com

Is Honeypot	Not Detected
Trading Cooldown	Not Detected
Can Pause Trade?	No
Pause Transfer?	No
Max Tax?	No
Is it Anti-whale?	No
Is Anti-bot?	Not Detected
Is it a Blacklist?	Not Detected
Blacklist Check	Passed
Can Mint?	No
Is it Proxy?	No
Can Take Ownership?	No
Hidden Owner?	Not Detected
Self Destruction?	Not Detected
Auditor Confidence	High

Overall Audit Result: PASSED

Email: contact@infinityblockchainsolutions.com

Documentation

We were given ARTEMIS smart contract code in the form of an ETHERSCAN web link.

https://etherscan.io/token/0x97D1d74B35b2ef5f0251C3E49361a74e68222122

The logic is straightforward. So, it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are on industry standard libraries like openzeppelin.

Email: contact@infinityblockchainsolutions.com

AS-IS overview

Functions

Sr.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	balanceOf	read	Passed	No Issue
7	transfer	write	Passed	No issue
8	allowance	Read	Passed	No Issue
9	approve	Write	Passed	No Issue
10	transferFrom	Write	Passed	No Issue
11	increaseAllowance	Write	Passed	No Issue
12	decreaseAllowance	Write	Passed	No Issue
13	_transfer	internal	Passed	No Issue
14	_mint	internal	Passed	No Issue
15	_burn	internal	Passed	No Issue
16	_approve	internal	Passed	No Issue
17	_spendAllowance	internal	Passed	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue
19	_afterTokenTransfer	internal	Passed	No Issue
20	onlyOwner	modifier	Passed	No Issue
21	owner	Read	Passed	No Issue
22	_checkOwner	internal	Passed	No Issue
23	renounceOwnership	Write	access only Owner	No Issue
24	transferOwnership	Write	access only Owner	No Issue
25	_transferOwnership	internal	Passed	No Issue

Email: contact@infinityblockchainsolutions.com

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: contact@infinityblockchainsolutions.com

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Informational / Best practices:

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

Ownable.sol

- renounce Ownership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.
- Transfer Ownership: The current owner can transfer ownership of the contract to a new account.

To make the smart contract 100% decentralized, we suggest renouncing ownership of the smart contract once its function is completed.

Email: contact@infinityblockchainsolutions.com

Conclusion

We were given a contract code in the form of Etherscan web link and we have used all possible tests based on given objects as files. We had **no issues** in the smart contract.

Given that potential test cases for such smart contract protocols can be limitless, we cannot guarantee future outcomes. We have utilized the latest static tools and conducted thorough manual reviews to cover as many test cases as possible.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed smart contract, based on standard audit procedure scope, is **"Secured".**

Our Methodology

We prioritize a transparent process and strive to make our reviews a collaborative effort. Our security audits aim to enhance the quality of the systems we review and provide sufficient remediation to protect users. The following outlines the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we search for potential issues related to code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also identify areas where implementing more defensive programming could reduce the risk of future mistakes and expedite future audits. While our primary focus is on the in-scope code, we also examine dependency code and behavior when relevant to our investigation.

Email: contact@infinityblockchainsolutions.com

Vulnerability Analysis:

Our audit techniques include manual code analysis and user interface interaction. We start by reviewing the project's website to gain a high-level understanding of the software's functionality. We then install and use the relevant software, exploring user interactions and roles. During this process, we brainstorm threat models and identify potential attack surfaces. Additionally, we read design documentation, review previous audit results, search for similar projects, and examine source code dependencies.

Documenting Results:

We adhere to a conservative and transparent process for identifying potential security vulnerabilities and ensuring their successful remediation. Whenever we discover a potential issue, we immediately create an entry for it in this document, even if we have not yet verified its feasibility and impact. This conservative approach means that we document our suspicions early, even if they are later found to be non-exploitable. Our process typically involves first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative step, and we strive to provide test code, log captures, or screenshots as evidence of our findings. After confirming an issue, we assess the feasibility of an attack in a live system.

Suggested Solutions:

We actively seek immediate mitigations that can be implemented by live deployments. Additionally, we outline requirements for remediation engineering in future releases. It's crucial that developers and deployment engineers thoroughly review our mitigation and remediation recommendations. Successful implementation of these measures is a collaborative effort that continues after we deliver our report, ensuring that they are in place before any details are made public.

Email: contact@infinityblockchainsolutions.com

Disclaimers

Infinity Blockchain Solutions Disclaimer

The Infinity Blockchain Solutions team has reviewed this smart contract in line with the best industry practices as of the report's date, focusing on: cybersecurity vulnerabilities and issues in the smart contract source code (details provided in this report), the Source Code's compilation, deployment, and functionality (ensuring it performs the intended functions). Given the infinite number of potential test cases, this audit does not make any guarantees about the code's security. It should not be considered a comprehensive assessment of the code's utility, safety, bug-free status, or any other contract statements. Despite our thorough analysis and detailed report, relying solely on this report is not recommended. We also advise implementing a bug bounty program to ensure the smart contract's high level of security.

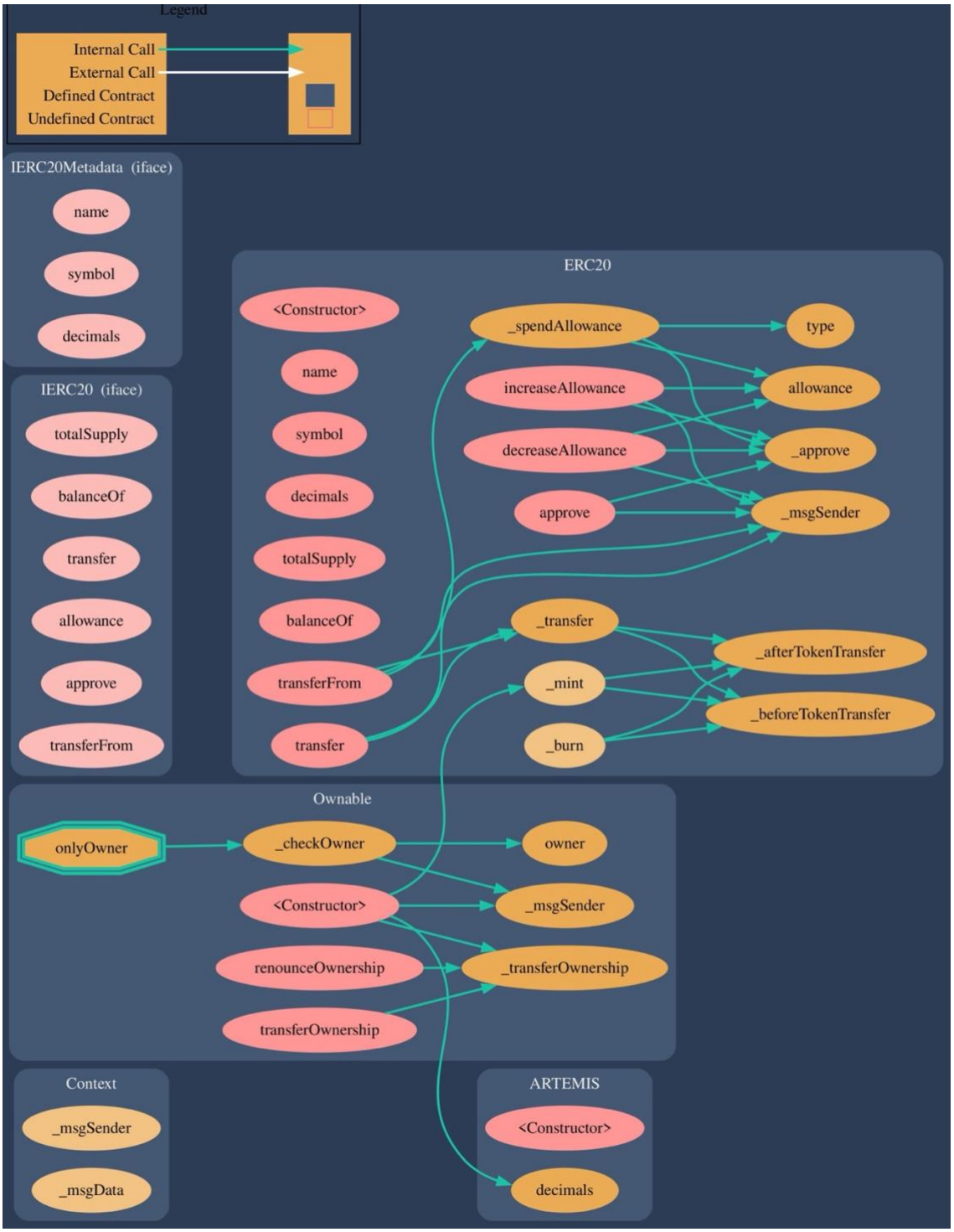
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform, which includes its programming language and other related software. These components can have their own vulnerabilities that may lead to hacks. Therefore, the audit cannot guarantee the absolute security of the audited smart contracts.

Email: contact@infinityblockchainsolutions.com

Appendix

Code Flow Diagram – Artemis.sol



Email: contact@infinityblockchainsolutions.com

Slither Results Log

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither Log >> Artemis.sol

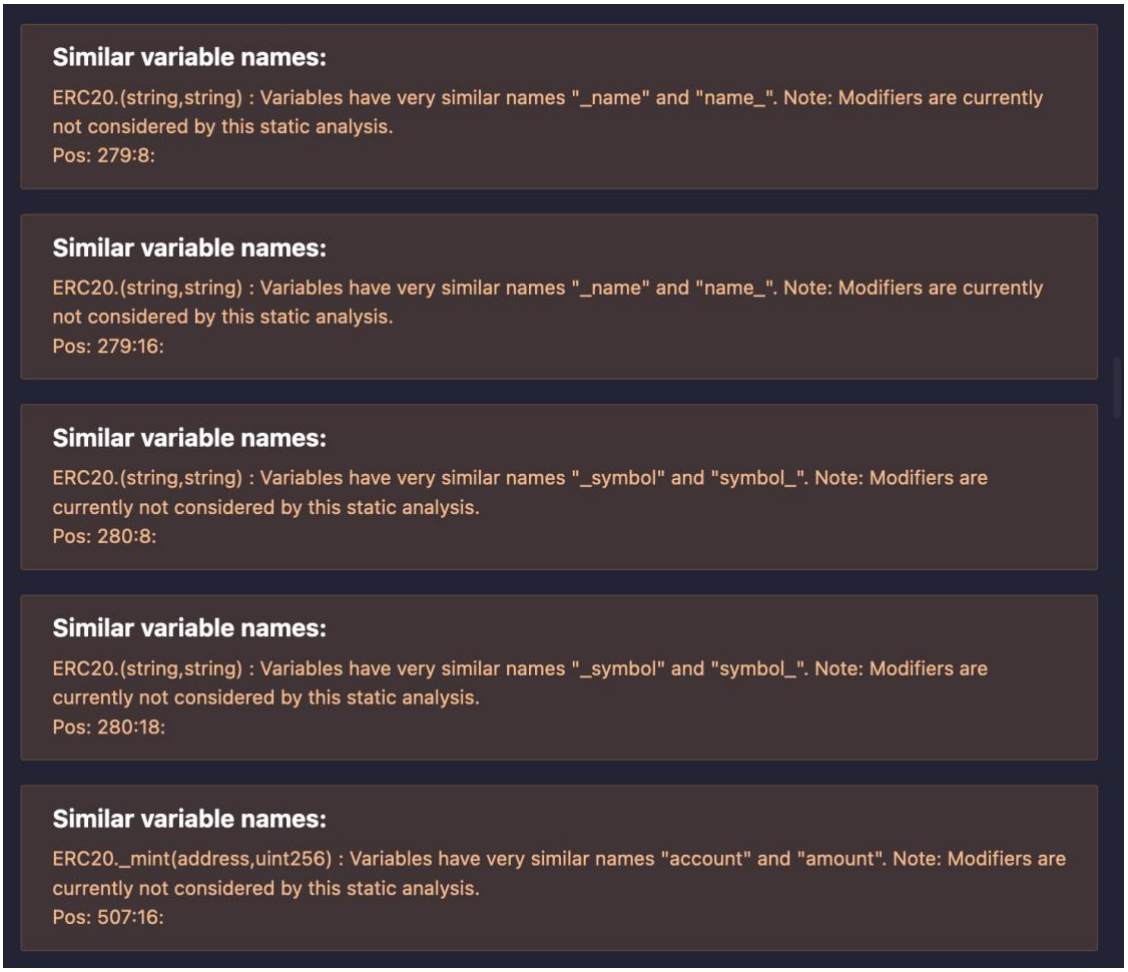
```
Context._msgData() (contracts/Artemis.sol#23-25) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Artemis.sol#532-548) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version0.8.24 (contracts/Artemis.sol#2) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6/0.8.7
solc-0.8.24 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

ARTEMIS.constructor() (contracts/Artemis.sol#642-644) uses literals with too many digits:
        - _mint(msg.sender,100000000000 * 10 ** decimals()) (contracts/Artemis.sol#643)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/Artemis.sol#88-90)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/Artemis.sol#96-102)
name() should be declared external:
        - ERC20.name() (contracts/Artemis.sol#286-288)
symbol() should be declared external:
        - ERC20.symbol() (contracts/Artemis.sol#294-296)
totalSupply() should be declared external:
        - ERC20.totalSupply() (contracts/Artemis.sol#318-320)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (contracts/Artemis.sol#325-329)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (contracts/Artemis.sol#339-346)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (contracts/Artemis.sol#368-375)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (contracts/Artemis.sol#393-402)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (contracts/Artemis.sol#416-423)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (contracts/Artemis.sol#439-454)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```


Email: contact@infinityblockchainsolutions.com

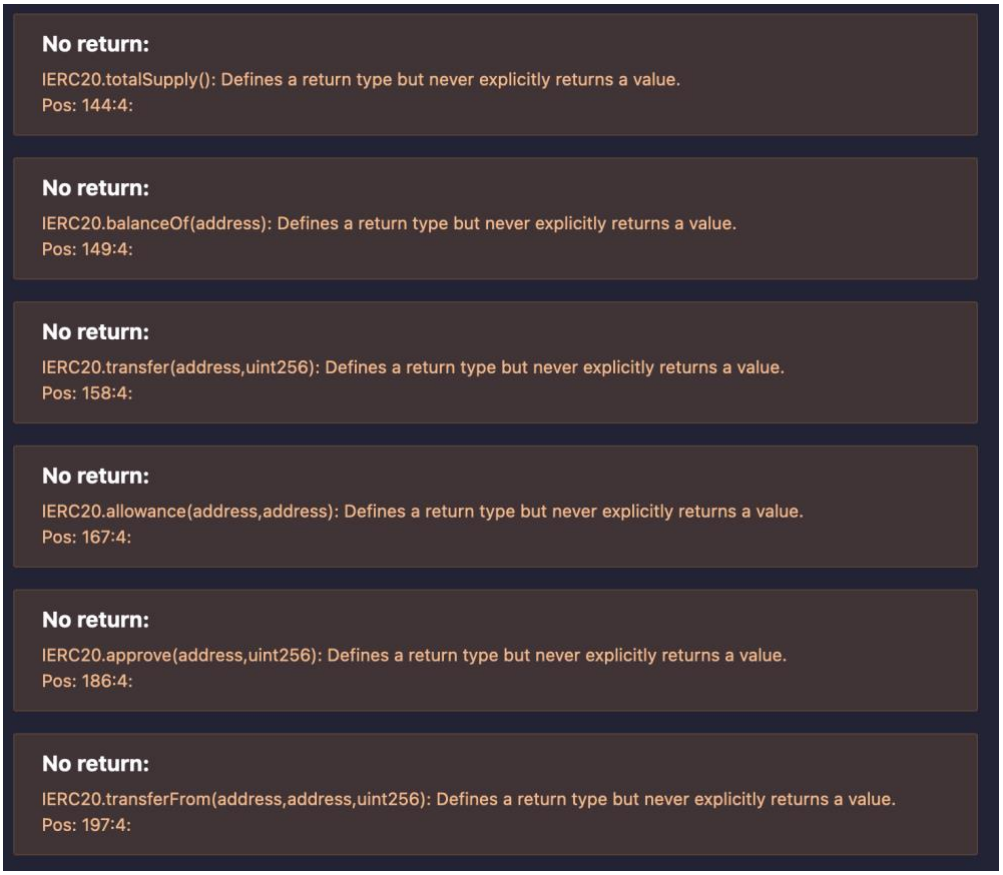
Solidity Static Analysis

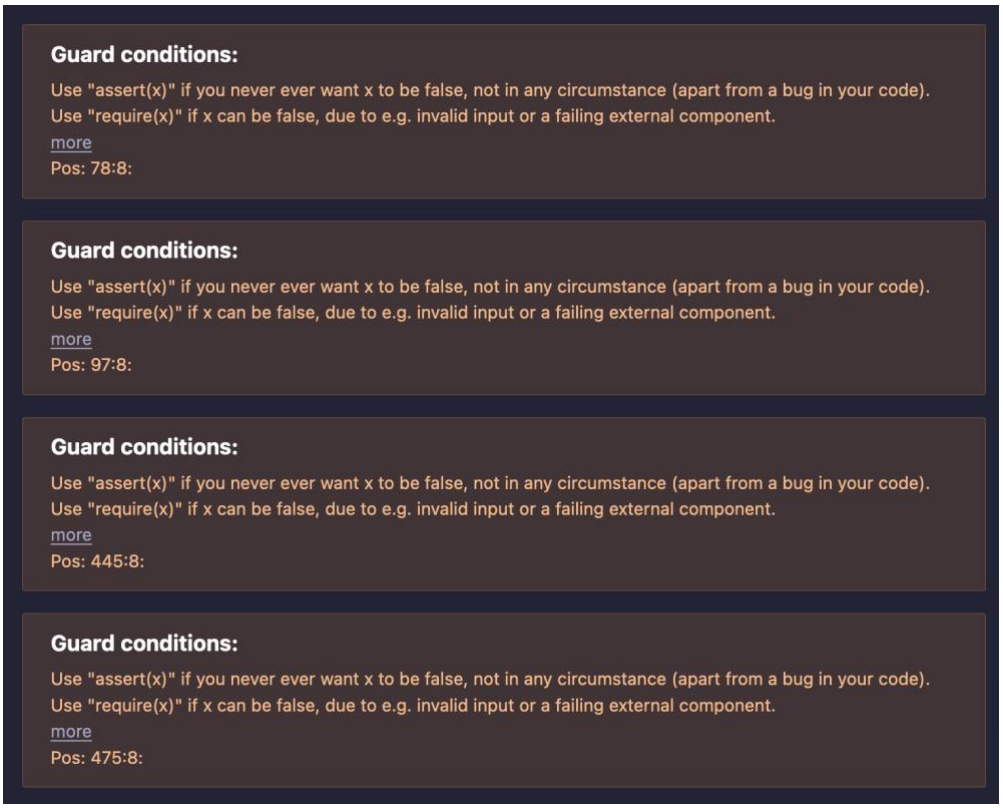
Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

Artemis.sol



Email: contact@infinityblockchainsolutions.com



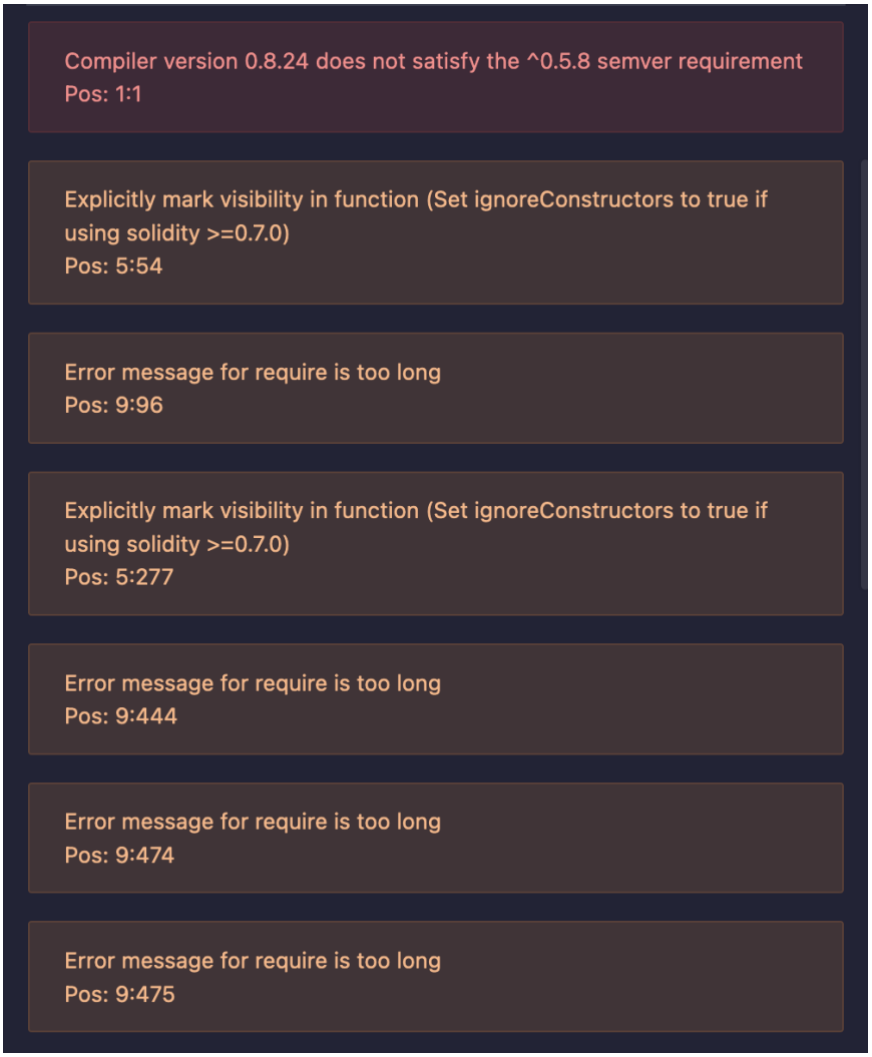


Email: contact@infinityblockchainsolutions.com

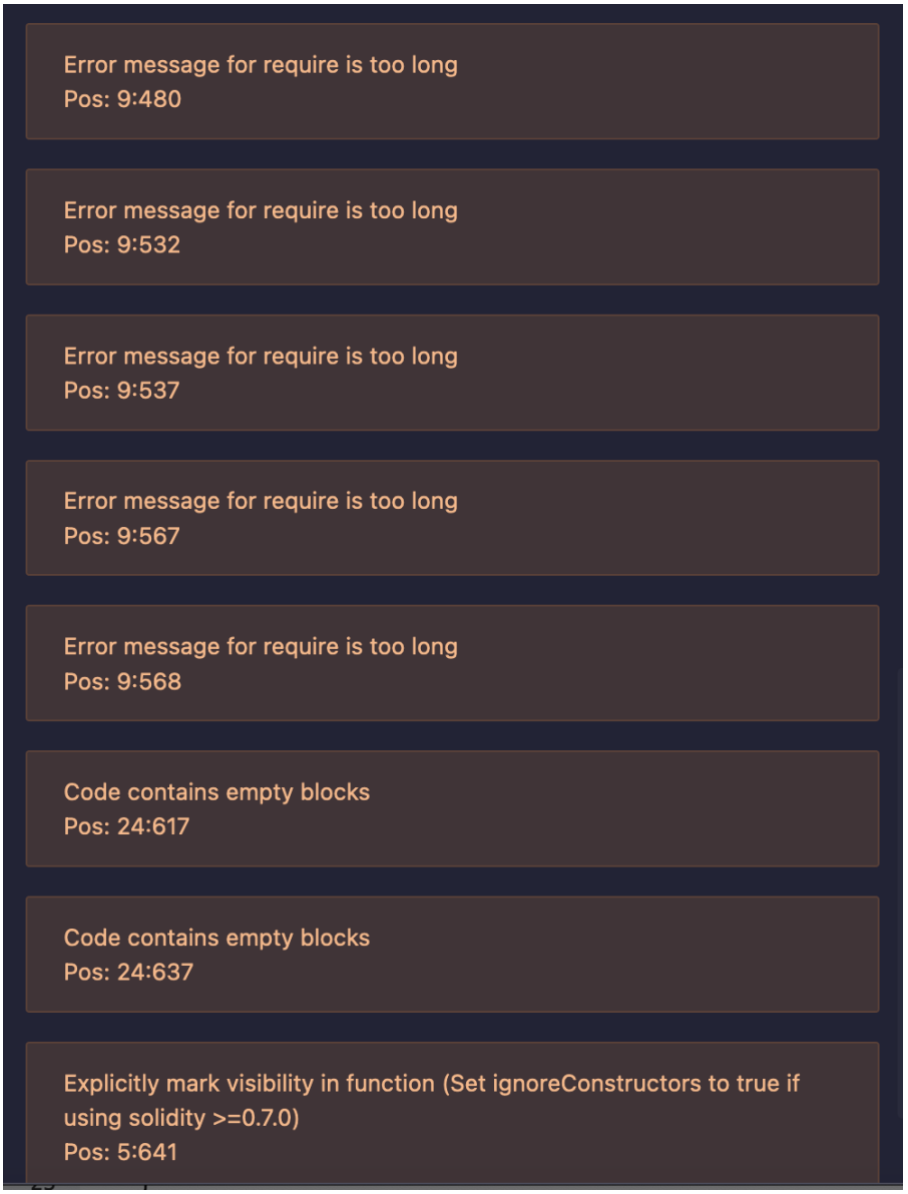
Solhint Linter

Linters are the utility tools that analyze the given source code and report programming errors, bugs, and stylistic errors. For the Solidity language, there are some linter tools available that a developer can use to improve the quality of their Solidity contracts.

Artemis.sol



Email: contact@infinityblockchainsolutions.com



Software analysis result:

These softwares reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Software analysis result: PASSED

Email: contact@infinityblockchainsolutions.com

INFINITY

BLOCKCHAIN SOLUTIONS