ASSURE DEFI®
THE VERIFICATION GOLD STANDARD

**Security** Assessment:

# Arcane Coin Token

January 21, 2024

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


VERIFIED BY ASSURE DEFI®
INTEGRITY • TRUST • CREDIBILITY

# 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 | 20% |
| Sale Tax | 20% |
| Cannot Sale | Pass |
| Cannot Sale | Pass |
| Max Tax | 20% |
| Modify Tax | Not Detected |
| Fee Check | Pass |
| Is Honeypot? | Not Detected |
| Trading Cooldown | Detected |
| Can Pause Trade? | Pass |
| Pause Transfer? | Not Detected |
| Max Tx? | Detected, Contract has MaxTx function. |
| Is Anti Whale? | Detected |
| Is Anti Bot? | Not Detected |

| Contract Privilege | Description |
| --- | --- |
| Is Blacklist? | Not Detected |
| Blacklist Check | Pass |
| is Whitelist? | Detected |
| Can Mint? | Pass |
| Is Proxy? | Not Detected |
| Can Take Ownership? | Not detected |
| Hidden Owner? | Not detected |
| Owner | 0x4c0B19AA31b20B946bBAD8000d192109F9df769d |
| Self Destruct? | Not Detected |
| External Call? | Not detected |
| Other? | Not detected |
| Holders | 2 |
| Auditor Confidence | Critical 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 | 0xA819FA2cf50FB232dF68f731BCa3E54e88498ae1 |
| Name | Arcane Coin |
| Token Tracker | Arcane Coin (ARCANE) |
| Decimals | 9 |
| Supply | 100,000,000 |
| Platform | Ethereum |
| compiler | v0.8.20+commit.a1b79de6 |
| Contract Name | Arcane |
| Optimization | Yes with 200 runs |
| LicenseType | MIT |
| Language | Solidity |
| Codebase | https://etherscan.io/token/0xA819FA2cf50FB232dF68f731BCa3E54e88498ae1#code |
| Payment Tx | Corporate |

# Main Contract Assessed
# Contract Name

| Name | Contract | Live |
|---|---|---|
| Arcane Coin | 0xA819FA2cf50FB232dF68f731BCa3E54e88498ae1 | No |

# TestNet Contract was Not Assessed

# Solidity Code Provided

| SolID | File Sha-1 | FileName |
|---|---|---|
| Arcade | e8a7e18c75d7c5b1eb831213e6b581cee3e5d7f4 | arcade.sol |
| Arcade | | |
| Arcade | | |
| Arcade | | |
| Arcade | | |
| Arcade | | |

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| SWC-102 | Pass | Outdated Compiler Version file.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

| ID | Severity | Name | File | location |
|---|---|---|---|---|
| SWC-115 | Medium | Authorization through tx.origin. | arcade.sol | L: 241 C: 55, L: 245 C: 51 |
| SWC-116 | Pass | A control flow decision is made based on The block.timestamp environment variable. | arcade.sol | L: 0 C: 0 |
| SWC-117 | Pass | Signature Malleability. | arcade.sol | L: 0 C: 0 |
| SWC-118 | Pass | Incorrect Constructor Name. | arcade.sol | L: 0 C: 0 |
| SWC-119 | Pass | Shadowing State Variables. | arcade.sol | L: 0 C: 0 |
| SWC-120 | Fail | Potential use of block.number as source of randonmness. | arcade.sol | L: 242 C: 30, L: 245 C: 64 |
| SWC-121 | Pass | Missing Protection against Signature Replay Attacks. | arcade.sol | L: 0 C: 0 |
| SWC-122 | Pass | Lack of Proper Signature Verification. | arcade.sol | L: 0 C: 0 |
| SWC-123 | Pass | Requirement Violation. | arcade.sol | L: 0 C: 0 |
| SWC-124 | Pass | Write to Arbitrary Storage Location. | arcade.sol | L: 0 C: 0 |
| SWC-125 | Pass | Incorrect Inheritance Order. | arcade.sol | L: 0 C: 0 |
| SWC-126 | Pass | Insufficient Gas Griefing. | arcade.sol | L: 0 C: 0 |
| SWC-127 | Pass | Arbitrary Jump with Function Type Variable. | arcade.sol | L: 0 C: 0 |
| SWC-128 | Pass | DoS With Block Gas Limit. | arcade.sol | L: 0 C: 0 |
| SWC-129 | Pass | Typographical Error. | arcade.sol | L: 0 C: 0 |
| SWC-130 | Pass | Right-To-Left-Override control character (U +202E). | arcade.sol | L: 0 C: 0 |
| SWC-131 | Pass | Presence of unused variables. | arcade.sol | L: 0 C: 0 |
| SWC-132 | Pass | Unexpected Ether balance. | arcade.sol | L: 0 C: 0 |

| ID | Severity | Name | File | location |
|---|---|---|---|---|
| SWC-133 | Pass | Hash Collisions with Multiple Variable Length Arguments. | arcade.sol | L: 0 C: 0 |
| SWC-134 | Pass | Message call with hardcoded gas amount. | arcade.sol | L: 0 C: 0 |
| SWC-135 | Pass | Code With No Effects (Irrelevant/Dead Code). | arcade.sol | L: 0 C: 0 |
| SWC-136 | Pass | Unencrypted Private Data On-Chain. | arcade.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-115 - Authorization through tx.origin

## CWE-477: Use of Obsolete Function

### Description:

tx.origin is a global variable in Solidity which returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract. A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.

### Remediation:

tx.origin should not be used for authorization. Use msg.sender instead.

### References:

Solidity Documentation - tx.origin

Ethereum Smart Contract Best Practices - Avoid using tx.origin

SigmaPrime - Visibility.

# 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?)

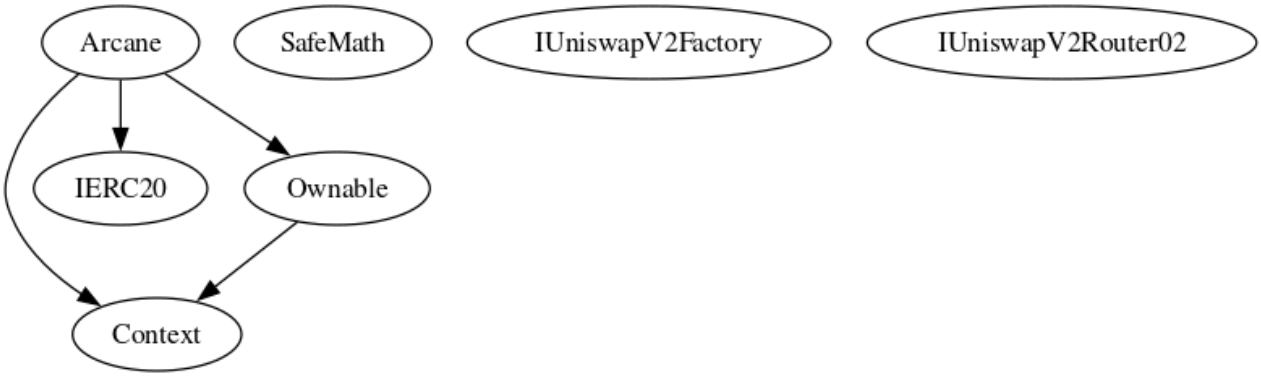
When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.

# Inheritance

## The contract for Arcane Coin has the following inheritance structure.

## The Project has a Total Supply of 100,000,000



# Privileged Functions (onlyOwner)

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

| Function Name | Parameters | Visibility |
| --- | --- | --- |
| openTrading | | public |
| removeLimits | | public |
| renounceOwnership | | public |

# ARCANE-14 | Unnecessary Use Of SafeMath

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Medium | arcade.sol: L: 37 C:14 | Detected |

## Description

The SafeMath library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations
will automatically revert in case of integer overflow or underflow.
library SafeMath {
An implementation of SafeMath library is found.
using SafeMath for uint256;
SafeMath library is used for uint256 type in  contract.

## Remediation

We advise removing the usage of SafeMath library and using the built-in arithmetic operations provided by the
Solidity programming language

## Project Action

# ARCANE-18 | Stop Transactions by using Enable Trade.

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Critical | arcade.sol: L: 322 C: 47 | 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

# ARCANE-20 | Complications with the antiWhale code..

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Optimization | Critical | arcade.sol: L: 230 C: 47 | Detected |

## Description

Inside the transfer there are some required functions that may transform the contract into a honeypot.

## Remediation

Simplify or clean the contract.

## 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. |
| 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 | 2 | 1 | 2 |
| High | 0 | 0 | 0 |
| Medium | 1 | 1 | 0 |
| Low | 0 | 0 | 0 |
| Informational | 0 | 0 | 0 |
| Total | 3 | 2 | 2 |

# Social Media Checks

| Social Media | URL | Result |
|---|---|---|
| Twitter | https://x.com/ArcaneErc20 | Pass |
| Other | https://medium.com/@arcaneofficial | Pass |
| Website | https://Archanemix.tech | Pass |
| Telegram | https://t.me/ArcaneOfficialCoin | 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 | 85/100 |
| Auditor Score | 86/100 |
| **Review by Section** | **Score** |
| Manual Scan Score | 28 |
| SWC Scan Score | 33 |
| Advance Check Score | 24 |

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 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:

- Several items were identified.
- Failed code, and modification of safemath detected.
- Customer Enabled Trade.

## Auditor Score =86
## 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

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 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 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.