



Cyberscope

Audit Report

RabitiAI

January 2025

Network ETH

Address 0xC61ac0627f0d4ecc78a9fee05C4290c29B59bA60

Audited by © cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

| Severity | Code | Description | Status |
|----------|------|-------------------------|------------|
| ● | ST | Stops Transactions | Unresolved |
| ● | OTUT | Transfers User's Tokens | Passed |
| ● | ELFM | Exceeds Fees Limit | Passed |
| ● | MT | Mints Tokens | Passed |
| ● | BT | Burns Tokens | Passed |
| ● | BC | Blacklists Addresses | Passed |

Diagnostics

● Critical ● Medium ● Minor / Informative

| Severity | Code | Description | Status |
|----------|------|--|------------|
| ● | MEE | Missing Events Emission | Unresolved |
| ● | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| ● | L09 | Dead Code Elimination | Unresolved |
| ● | L11 | Unnecessary Boolean equality | Unresolved |
| ● | L19 | Stable Compiler Version | Unresolved |

Table of Contents

| | |
|--|-----------|
| Analysis | 1 |
| Diagnostics | 2 |
| Table of Contents | 3 |
| Risk Classification | 4 |
| Review | 5 |
| Audit Updates | 5 |
| Source Files | 5 |
| Findings Breakdown | 6 |
| ST - Stops Transactions | 7 |
| Description | 7 |
| Recommendation | 7 |
| MEE - Missing Events Emission | 8 |
| Description | 8 |
| Recommendation | 8 |
| L04 - Conformance to Solidity Naming Conventions | 9 |
| Description | 9 |
| Recommendation | 9 |
| L09 - Dead Code Elimination | 10 |
| Description | 10 |
| Recommendation | 11 |
| L11 - Unnecessary Boolean equality | 12 |
| Description | 12 |
| Recommendation | 12 |
| L19 - Stable Compiler Version | 13 |
| Description | 13 |
| Recommendation | 13 |
| Functions Analysis | 14 |
| Inheritance Graph | 16 |
| Flow Graph | 17 |
| Summary | 18 |
| Disclaimer | 19 |
| About Cyberscope | 20 |

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation:** This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation:** This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical:** Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium:** Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor:** Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative:** Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

| Severity | Likelihood / Impact of Exploitation |
|-----------------------|--|
| ● Critical | Highly Likely / High Impact |
| ● Medium | Less Likely / High Impact or Highly Likely/ Lower Impact |
| ● Minor / Informative | Unlikely / Low to no Impact |

Review

| | |
|-------------------|---|
| Contract Name | RabitiAI |
| Compiler Version | v0.8.26+commit.8a97fa7a |
| Optimization | 200 runs |
| Explorer | https://etherscan.io/address/0xc61ac0627f0d4ecc78a9fee05c4290c29b59ba60 |
| Address | 0xc61ac0627f0d4ecc78a9fee05c4290c29b59ba60 |
| Network | ETH |
| Symbol | RIAI |
| Decimals | 18 |
| Total Supply | 10,000,000,000 |
| Badge Eligibility | Yes |

Audit Updates

| | |
|---------------|-------------|
| Initial Audit | 15 Jan 2025 |
|---------------|-------------|

Source Files

| | |
|--------------|--|
| Filename | SHA256 |
| RabitiAI.sol | bac7494389f638f57491331cfc2fd017a70ac7c8eb38cb25e662213e24828a40 |

Findings Breakdown



| | |
|---------------------|---|
| Critical | 1 |
| Medium | 0 |
| Minor / Informative | 5 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|---------------------|------------|--------------|----------|-------|
| Critical | 1 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 |
| Minor / Informative | 5 | 0 | 0 | 0 |

ST - Stops Transactions

| | |
|-------------|-------------------|
| Criticality | Critical |
| Location | RabitiAI.sol#L585 |
| Status | Unresolved |

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enabled the owner will not be able to disable them again.

```
require(  
    trading == true ||  
    whitelist[sender] ||  
    whitelist[recipient] ||  
    whitelist[msg.sender],  
    "Trading not enabled"  
);
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

MEE - Missing Events Emission

| | |
|--------------------|-----------------------|
| Criticality | Minor / Informative |
| Location | RabitiAI.sol#L569,577 |
| Status | Unresolved |

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
trading = _trading;  
whitelist[_address] = _status;
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

L04 - Conformance to Solidity Naming Conventions

| | |
|--------------------|---------------------------|
| Criticality | Minor / Informative |
| Location | RabitiAI.sol#L567,574,575 |
| Status | Unresolved |

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
bool _trading
address _address
bool _status
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/stable/style-guide.html#naming-conventions>.

L09 - Dead Code Elimination

| | |
|-------------|---------------------|
| Criticality | Minor / Informative |
| Location | RabitiAI.sol#L499 |
| Status | Unresolved |

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _burn(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: burn from the zero address");

    _beforeTokenTransfer(account, address(0), amount);

    uint256 accountBalance = _balances[account];
    require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
    _balances[account] = accountBalance - amount;
    _totalSupply -= amount;

    emit Transfer(account, address(0), amount);
}
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L11 - Unnecessary Boolean equality

| | |
|-------------|-----------------------|
| Criticality | Minor / Informative |
| Location | RabitiAI.sol#L568,585 |
| Status | Unresolved |

Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
require(trading == false, "Trading already enabled")

require(
    trading == true ||
    whitelist[sender] ||
    whitelist[recipient] ||
    whitelist[msg.sender],
    "Trading not enabled"
)
```

Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

L19 - Stable Compiler Version

| | |
|--------------------|---------------------|
| Criticality | Minor / Informative |
| Location | RabitiAI.sol#L3 |
| Status | Unresolved |

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

Recommendation

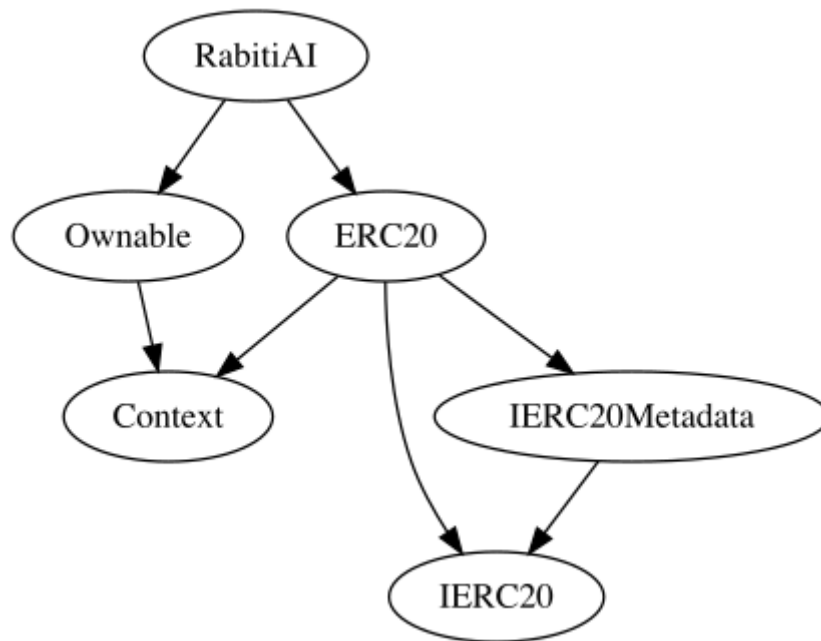
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

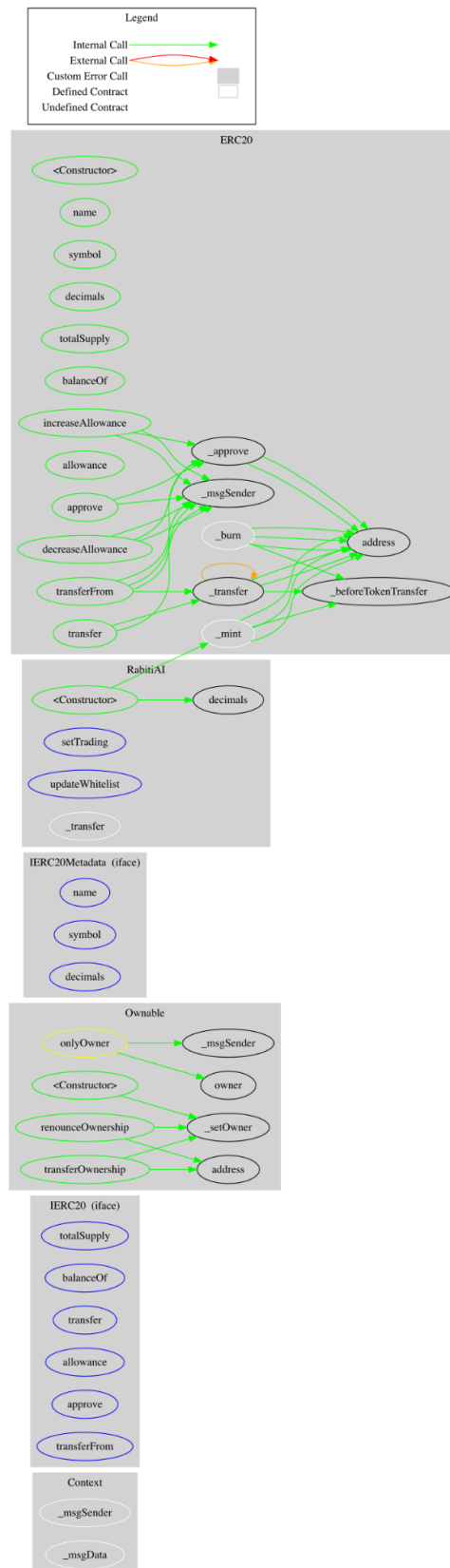
| Contract | Type | Bases | | |
|----------|----------------------|---|------------|-----------|
| | Function Name | Visibility | Mutability | Modifiers |
| | | | | |
| ERC20 | Implementation | Context, IERC20, IERC20Meta data | | |
| | | Public | ✓ | - |
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | Public | | - |
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | _transfer | Internal | ✓ | |
| | _mint | Internal | ✓ | |
| | _burn | Internal | ✓ | |
| | _approve | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |
| | | | | |

| RabitiAI | Implementation | ERC20, Ownable | | |
|----------|-----------------|-------------------|---|------------------|
| | | Public | ✓ | ERC20 Ownable |
| | setTrading | External | ✓ | onlyOwner |
| | updateWhitelist | External | ✓ | onlyOwner |
| | _transfer | Internal | ✓ | |

Inheritance Graph



Flow Graph



Summary

RabitiAI contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. RabitiAI is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are 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.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

cyberscope.io