



Cyberscope

Audit Report

HumAI

June 2023

Network BSC

Address 0xeb53a5f8f2de50248bc44f91650de438ce17b5eb

Audited by © cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

| Severity | Code | Description | Status |
|----------|------|-------------------------|--------|
| ● | ST | Stops Transactions | Passed |
| ● | 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 |
|----------|------|--|------------|
| ● | RSML | Redundant SafeMath Library | Unresolved |
| ● | IDI | Immutable Declaration Improvement | Unresolved |
| ● | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| ● | L09 | Dead Code Elimination | Unresolved |
| ● | L16 | Validate Variable Setters | Unresolved |

Table of Contents

| | |
|--|-----------|
| Analysis | 1 |
| Diagnostics | 2 |
| Table of Contents | 3 |
| Review | 4 |
| Audit Updates | 4 |
| Source Files | 4 |
| Findings Breakdown | 5 |
| RSML - Redundant SafeMath Library | 6 |
| Description | 6 |
| Recommendation | 6 |
| IDI - Immutable Declaration Improvement | 7 |
| Description | 7 |
| Recommendation | 7 |
| L04 - Conformance to Solidity Naming Conventions | 8 |
| Description | 8 |
| Recommendation | 8 |
| L09 - Dead Code Elimination | 9 |
| Description | 9 |
| Recommendation | 9 |
| L16 - Validate Variable Setters | 11 |
| Description | 11 |
| Recommendation | 11 |
| Functions Analysis | 12 |
| Inheritance Graph | 15 |
| Flow Graph | 16 |
| Summary | 17 |
| Disclaimer | 18 |
| About Cyberscope | 19 |

Review

| | |
|------------------|---|
| Contract Name | AntiBotStandardToken |
| Compiler Version | v0.8.4+commit.c7e474f2 |
| Optimization | 200 runs |
| Explorer | https://bscscan.com/address/0xeb53a5f8f2de50248bc44f91650de438ce17b5eb |
| Address | 0xeb53a5f8f2de50248bc44f91650de438ce17b5eb |
| Network | BSC |
| Symbol | HumAI |
| Decimals | 18 |
| Total Supply | 500,000,000 |

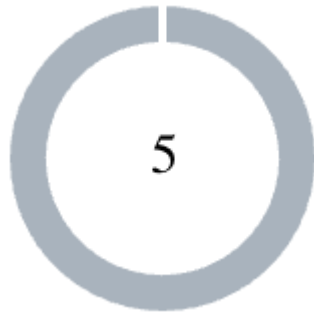
Audit Updates

| | |
|---------------|-------------|
| Initial Audit | 03 Jun 2023 |
|---------------|-------------|

Source Files

| | |
|--------------------------|--|
| Filename | SHA256 |
| AntiBotStandardToken.sol | 018174c70f79aa836fc246f8ee7d0890eb3883c1b40148388f8eaff14c97b391 |

Findings Breakdown



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

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

RSML - Redundant SafeMath Library

| | |
|-------------|--------------------------|
| Criticality | Minor / Informative |
| Location | AntiBotStandardToken.sol |
| Status | Unresolved |

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on

<https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes>.

IDI - Immutable Declaration Improvement

| | |
|--------------------|---------------------------------------|
| Criticality | Minor / Informative |
| Location | AntiBotStandardToken.sol#L490,491,495 |
| Status | Unresolved |

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
_name  
_symbol  
pinkAntiBot
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L04 - Conformance to Solidity Naming Conventions

| | |
|--------------------|-------------------------------|
| Criticality | Minor / Informative |
| Location | AntiBotStandardToken.sol#L509 |
| 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 _enable
```

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/v0.8.17/style-guide.html#naming-convention>.

L09 - Dead Code Elimination

| | |
|-------------|-----------------------------------|
| Criticality | Minor / Informative |
| Location | AntiBotStandardToken.sol#L764,809 |
| 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);

    _balances[account] = _balances[account].sub(
        amount,
        "ERC20: burn amount exceeds balance"
    );
    _totalSupply = _totalSupply.sub(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.

L16 - Validate Variable Setters

| | |
|--------------------|-------------------------------|
| Criticality | Minor / Informative |
| Location | AntiBotStandardToken.sol#L506 |
| Status | Unresolved |

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
payable(serviceFeeReceiver_).transfer(serviceFee_)
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

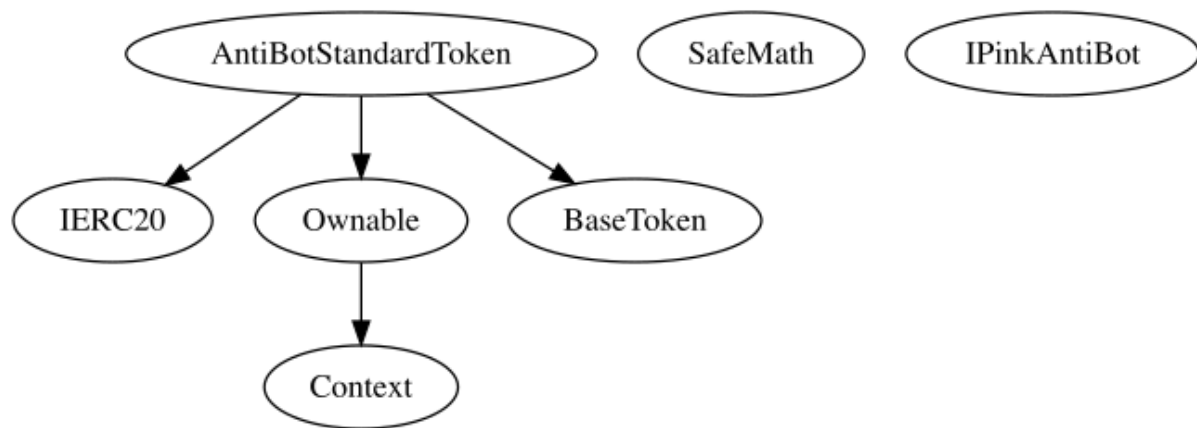
Functions Analysis

| Contract | Type | Bases | | |
|----------|-------------------|------------|------------|-----------|
| | Function Name | Visibility | Mutability | Modifiers |
| IERC20 | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| Context | Implementation | | | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| Ownable | Implementation | Context | | |
| | | Public | ✓ | - |
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | _setOwner | Private | ✓ | |
| | | | | |

| | | | | |
|-----------------------------|--------------------|----------------------------------|---------|---|
| SafeMath | Library | | | |
| | tryAdd | Internal | | |
| | trySub | Internal | | |
| | tryMul | Internal | | |
| | tryDiv | Internal | | |
| | tryMod | Internal | | |
| | add | Internal | | |
| | sub | Internal | | |
| | mul | Internal | | |
| | div | Internal | | |
| | mod | Internal | | |
| | sub | Internal | | |
| | div | Internal | | |
| | mod | Internal | | |
| | | | | |
| IPinkAntiBot | Interface | | | |
| | setTokenOwner | External | ✓ | - |
| | onPreTransferCheck | External | ✓ | - |
| | | | | |
| BaseToken | Implementation | | | |
| | | | | |
| AntiBotStandardToken | Implementation | IERC20, Ownable, BaseToken | | |
| | | Public | Payable | - |

| | | | | |
|--|----------------------|----------|---|-----------|
| | setEnableAntiBot | External | ✓ | onlyOwner |
| | 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 | ✓ | |
| | _setupDecimals | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |

Inheritance Graph



Flow Graph



Summary

HumAI contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. HumAI 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. There is also a limit of max 25% fees.

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

<https://www.cyberscope.io>