



Cyberscope

Audit Report

UltronAI

January 2023

SHA256 8a695efe4e441d3ae6d6c8ba5d977fd8658bb31c388ac489c446ad1334c636af

Audited by © cyberscope

Table of Contents

| | |
|---|-----------|
| Table of Contents | 1 |
| Review | 3 |
| Audit Updates | 3 |
| Source Files | 3 |
| Analysis | 4 |
| Diagnostics | 5 |
| L04 - Conformance to Solidity Naming Conventions | 6 |
| Description | 6 |
| Recommendation | 7 |
| L05 - Unused State Variable | 8 |
| Description | 8 |
| Recommendation | 8 |
| L07 - Missing Events Arithmetic | 9 |
| Description | 9 |
| Recommendation | 9 |
| L12 - Using Variables before Declaration | 10 |
| Description | 10 |
| Recommendation | 10 |
| L14 - Uninitialized Variables in Local Scope | 11 |
| Description | 11 |
| Recommendation | 11 |
| L16 - Validate Variable Setters | 12 |
| Description | 12 |
| Recommendation | 12 |
| L20 - Succeeded Transfer Check | 13 |
| Description | 13 |
| Recommendation | 13 |
| Functions Analysis | 14 |
| Inheritance Graph | 18 |
| Flow Graph | 19 |
| Summary | 20 |

| | |
|-------------------|-----------|
| Disclaimer | 21 |
|-------------------|-----------|

| | |
|-------------------------|-----------|
| About Cyberscope | 22 |
|-------------------------|-----------|

Review

| | |
|----------------|---|
| Contract Name | UltronAI |
| Testing Deploy | https://testnet.bscscan.com/address/0x7bf5982bbf07d9547fbc66b24f53ca13b02d4dde |
| Symbol | Ultron AI |
| Decimals | 18 |
| Total Supply | 1,000,000,000 |

Audit Updates

| | |
|---------------|-------------|
| Initial Audit | 04 Feb 2023 |
|---------------|-------------|

Source Files

| | |
|---------------------------------------|--|
| Filename | SHA256 |
| contracts/testingDeploy/Ultron_AI.sol | 8a695efe4e441d3ae6d6c8ba5d977fd8658bb31c388ac489c446ad1334c636af |

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

| Severity | Code | Description | Status |
|----------|------|------------------------------------|--------|
| ● | ST | Stops Transactions | Passed |
| ● | OCTD | Transfers Contract's Tokens | Passed |
| ● | OTUT | Transfers User's Tokens | Passed |
| ● | ELFM | Exceeds Fees Limit | Passed |
| ● | ULTW | Transfers Liquidity to Team Wallet | Passed |
| ● | MT | Mints Tokens | Passed |
| ● | BT | Burns Tokens | Passed |
| ● | BC | Blacklists Addresses | Passed |

Diagnostics

● Critical ● Medium ● Minor / Informative

| Severity | Code | Description | Status |
|----------|------|--|------------|
| ● | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| ● | L05 | Unused State Variable | Unresolved |
| ● | L07 | Missing Events Arithmetic | Unresolved |
| ● | L12 | Using Variables before Declaration | Unresolved |
| ● | L14 | Uninitialized Variables in Local Scope | Unresolved |
| ● | L16 | Validate Variable Setters | Unresolved |
| ● | L20 | Succeeded Transfer Check | Unresolved |

L04 - Conformance to Solidity Naming Conventions

| | |
|--------------------|---|
| Criticality | Minor / Informative |
| Location | contracts/testingDeploy/Ultron_AI.sol#L33,102,134,135,136,137,138,153,160,166,167,168,169,184,410 |
| 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.

```
function WETH() external pure returns (address);
IERC20 IERC20_token
uint256 constant private startingSupply = 1_000_000_000
string constant private _name = "Ultron AI"
string constant private _symbol = "Ultron AI"
uint8 constant private _decimals = 18
uint256 constant private _tTotal = startingSupply * 10**_decimals

Fees public _taxRates = Fees({
    buyFee: 200,
    sellFee: 0,
    transferFee: 0,
    negativeTax: 200
})

...
```

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>.

L05 - Unused State Variable

| | |
|--------------------|--|
| Criticality | Minor / Informative |
| Location | contracts/testingDeploy/Ultron_AI.sol#L130 |
| Status | Unresolved |

Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
mapping (address => bool) private _isExcludedFromLimits
```

Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

L07 - Missing Events Arithmetic

| | |
|--------------------|--|
| Criticality | Minor / Informative |
| Location | contracts/testingDeploy/Ultron_AI.sol#L446,456,494 |
| Status | Unresolved |

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor
piSwapPercent = priceImpactSwapPercent
buybackThreshold = threshold * 10**thresholdMultiplier
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L12 - Using Variables before Declaration

| | |
|--------------------|--|
| Criticality | Minor / Informative |
| Location | contracts/testingDeploy/Ultron_AI.sol#L634,680 |
| Status | Unresolved |

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or if the variable has been declared in a different scope. It is not a good practice to use a local variable before it has been declared.

```
uint256 initSwapAmount
uint256 initThreshold
bool check
```

Recommendation

By declaring local variables before using them, contract ensures that it operates correctly. It's important to be aware of this rule when working with local variables, as using a variable before it has been declared can lead to unexpected behavior and can be difficult to debug.

L14 - Uninitialized Variables in Local Scope

| | |
|--------------------|--|
| Criticality | Minor / Informative |
| Location | contracts/testingDeploy/Ultron_AI.sol#L634,679,680 |
| Status | Unresolved |

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 initSwapAmount
uint256 initThreshold
bool checked
bool check
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L16 - Validate Variable Setters

| | |
|--------------------|--|
| Criticality | Minor / Informative |
| Location | contracts/testingDeploy/Ultron_AI.sol#L112,292 |
| 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(receiver).transfer(address(this).balance)
operator = newOperator
```

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.

L20 - Succeeded Transfer Check

| | |
|--------------------|--|
| Criticality | Minor / Informative |
| Location | contracts/testingDeploy/Ultron_AI.sol#L117,662 |
| Status | Unresolved |

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
TOKEN.transfer(receiver, TOKEN.balanceOf(address(this)))  
TOKEN.transfer(_owner, TOKEN.balanceOf(address(this)))
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the [Openzeppelin library](#).

Functions Analysis

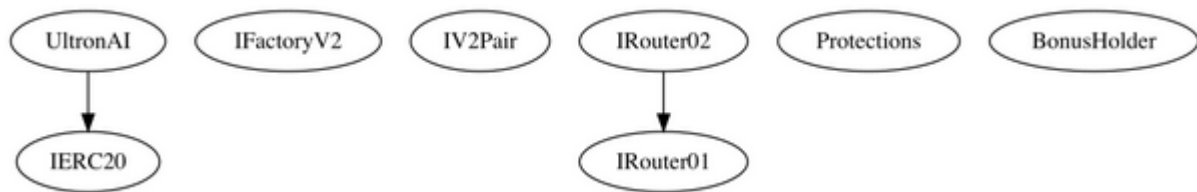
| Contract | Type | Bases | | |
|-------------------|---------------|------------|------------|-----------|
| | Function Name | Visibility | Mutability | Modifiers |
| | | | | |
| IERC20 | Interface | | | |
| | totalSupply | External | | - |
| | decimals | External | | - |
| | symbol | External | | - |
| | name | External | | - |
| | getOwner | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| IFactoryV2 | Interface | | | |
| | getPair | External | | - |
| | createPair | External | ✓ | - |
| | | | | |
| IV2Pair | Interface | | | |
| | factory | External | | - |
| | getReserves | External | | - |
| | sync | External | ✓ | - |
| | | | | |
| IRouter01 | Interface | | | |
| | factory | External | | - |
| | WETH | External | | - |

| | | | | |
|--------------------|---|-----------|---------|-----------|
| | addLiquidityETH | External | Payable | - |
| | addLiquidity | External | ✓ | - |
| | swapExactETHForTokens | External | Payable | - |
| | getAmountsOut | External | | - |
| | getAmountsIn | External | | - |
| | | | | |
| IRouter02 | Interface | IRouter01 | | |
| | swapExactTokensForETHSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactETHForTokensSupportingFeeOnTransferTokens | External | Payable | - |
| | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | ✓ | - |
| | swapExactTokensForTokens | External | ✓ | - |
| | | | | |
| Protections | Interface | | | |
| | checkUser | External | ✓ | - |
| | setLaunch | External | ✓ | - |
| | getInits | External | ✓ | - |
| | setLpPair | External | ✓ | - |
| | setProtections | External | ✓ | - |
| | removeSniper | External | ✓ | - |
| | | | | |
| BonusHolder | Implementation | | | |
| | | Public | ✓ | - |
| | sweep | External | ✓ | onlyOwner |
| | sweepExternalTokens | External | ✓ | onlyOwner |
| | | | | |
| UltronAI | Implementation | IERC20 | | |
| | | Public | Payable | - |
| | transferOwner | External | ✓ | onlyOwner |

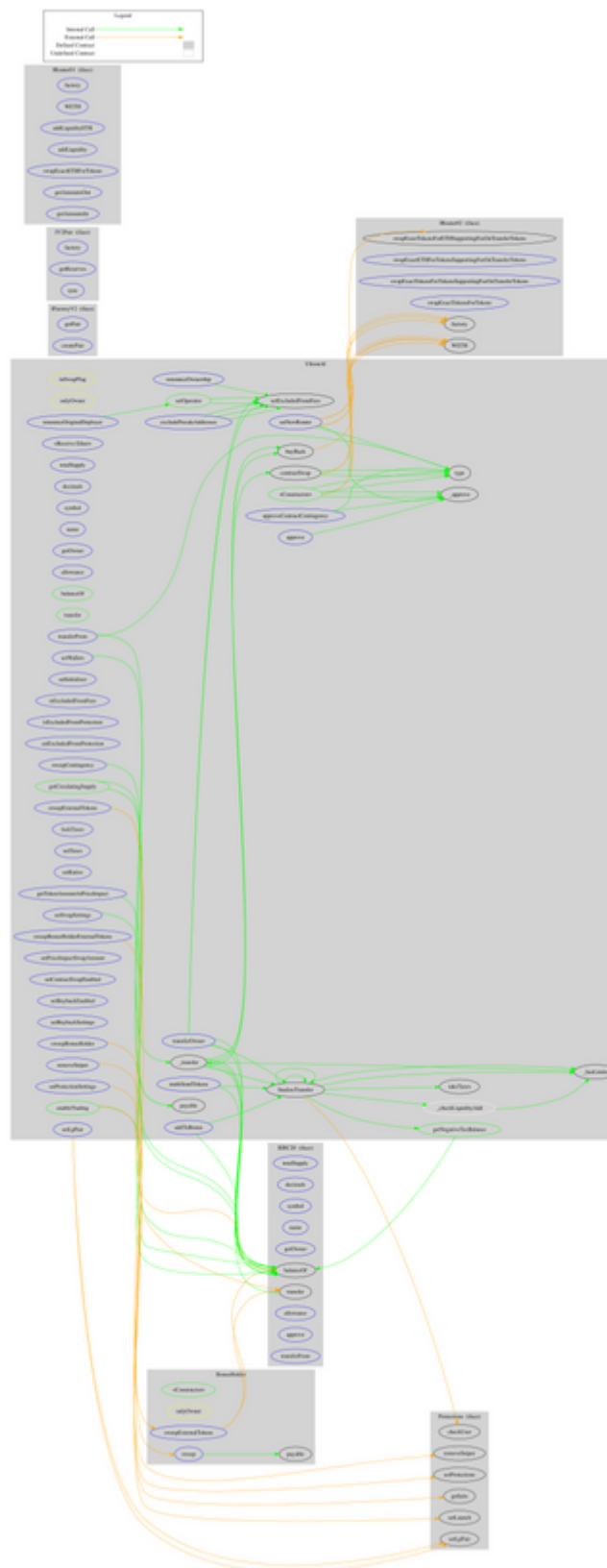
| | | | | |
|--|----------------------------|----------|---------|-----------|
| | renounceOwnership | External | ✓ | onlyOwner |
| | setOperator | Public | ✓ | - |
| | renounceOriginalDeployer | External | ✓ | - |
| | | External | Payable | - |
| | totalSupply | External | | - |
| | decimals | External | | - |
| | symbol | External | | - |
| | name | External | | - |
| | getOwner | External | | - |
| | allowance | External | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | approve | External | ✓ | - |
| | _approve | Internal | ✓ | |
| | approveContractContingency | External | ✓ | onlyOwner |
| | transferFrom | External | ✓ | - |
| | setNewRouter | External | ✓ | onlyOwner |
| | setLpPair | External | ✓ | onlyOwner |
| | setInitializer | External | ✓ | onlyOwner |
| | isExcludedFromFees | External | | - |
| | setExcludedFromFees | Public | ✓ | onlyOwner |
| | isExcludedFromProtection | External | | - |
| | setExcludedFromProtection | External | ✓ | onlyOwner |
| | getCirculatingSupply | Public | | - |
| | removeSniper | External | ✓ | onlyOwner |
| | setProtectionSettings | External | ✓ | onlyOwner |
| | lockTaxes | External | ✓ | onlyOwner |
| | setTaxes | External | ✓ | onlyOwner |
| | setRatios | External | ✓ | onlyOwner |

| | | | | |
|--|--------------------------------|----------|---|------------|
| | setWallets | External | ✓ | onlyOwner |
| | getTokenAmountAtPriceImpact | External | | - |
| | setSwapSettings | External | ✓ | onlyOwner |
| | setPriceImpactSwapAmount | External | ✓ | onlyOwner |
| | setContractSwapEnabled | External | ✓ | onlyOwner |
| | excludePresaleAddresses | External | ✓ | onlyOwner |
| | getNegativeTaxBalance | Public | | - |
| | setBuybackEnabled | External | ✓ | onlyOwner |
| | setBuybackSettings | External | ✓ | onlyOwner |
| | _hasLimits | Internal | | |
| | _transfer | Internal | ✓ | |
| | contractSwap | Internal | ✓ | inSwapFlag |
| | buyBack | Internal | ✓ | |
| | _checkLiquidityAdd | Internal | ✓ | |
| | enableTrading | Public | ✓ | onlyOwner |
| | addToBonus | External | ✓ | - |
| | sweepBonusHolder | External | ✓ | onlyOwner |
| | sweepContingency | External | ✓ | onlyOwner |
| | sweepExternalTokens | External | ✓ | onlyOwner |
| | sweepBonusHolderExternalTokens | External | ✓ | onlyOwner |
| | multiSendTokens | External | ✓ | onlyOwner |
| | finalizeTransfer | Internal | ✓ | |
| | takeTaxes | Internal | ✓ | |

Inheritance Graph



Flow Graph



Summary

UltronAI 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 2% on buy 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>