QuillAudits

Audit Report
April, 2023

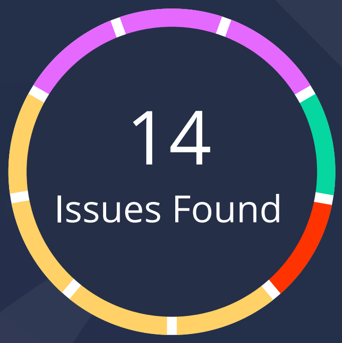
For

web3AiTools

Table of Content

Executive Summary

Project Name	Web3AiTools
Overview	Web3Ai Tools is a set of nine innovative AI tools that cater to the challenges faced in decentralized development ecosystems. These tools employ AI to automate tasks, enhance productivity, and reduce the time and energy required. Moreover, Web3Ai Tools includes the BEP20 upgradable contract, specifically designed for smart contract development. Incorporating Web3Ai Tools can help developers simplify their development process by leveraging cutting-edge AI technology.
Timeline	4th April, 2023 to 10th April, 2023
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyze Web3AiTools codebase for quality, security, and correctness. Contracts in Scope: Web3AiTools.sol *https://drive.google.com/drive/folders/1bI4WHMraHesJkTDNQVT5WlWrsEujlZUQ?usp=sharing*
Fixed In	*https://drive.google.com/drive/folders/1v-G5dyADcFMsMqxH-Z_NPUOPyTeWl0Mg?usp=sharing*



🟥 High 🟨 Medium

🟩 Low 🟪 Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	**1**	**5**	**1**	**5**
Partially Resolved Issues	0	0	0	0
Resolved Issues	**1**	0	**1**	0

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved
These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - Web3AiTools.sol

High Severity Issues

1. Founder and Team tokens - possible early unlocks

Description

This issue deals with how the contract can lose 10% of its supply if the owner gets compromised.

Scenario 1: Via early token unlock
The releaseTime is checked to confirm it's not zero but does not check to see if releaseTime is greater than contractCreatedTime.

```
function setRealeaseTime(uint256 _time) external onlyOwner{
    require(_time != 0, "Should not be Zero"); ...
}
```

This means the claim function can be manipulated to dispense tokens much earlier than expected if the owner forgets to set releaseTime or uses a value too little.

```
function claim() external payable onlyLockedWallets {
    uint256 quartersPassed = (block.timestamp - releaseTime)/quarterTime;
    if(quartersPassed > 10) {
        quartersPassed = 10;
}

uint256 claimableQuarters =  quartersPassed - FounderquartersClaimed;
```

For small values of releaseTime (which could be zero before initialization) less than (contractCreatedTime + one quarter), quartersPassed can get greater than 10 and as such would make claimableQuarters to be equal to 10, thus allowing to withdraw all tokens allocated (10% of total supply) in the very first quarter.

Scenario 2: Malicious/Compromised owner
The erc20 withdraw function in the contract allows any address to be passed in, including the token's contract address. If that happens, withdraw() would check for the ether balance of the contract (another low severity issue described below) and then transfer the available balance of the contract - 10% of the totalSupply - to the owner which could have been compromised. The tokens can then be sold on the open market without any restrictions.

Remediation

- releaseTime can be set in the initializer.
- Ensure that all arithmetic checks are not easily manipulated to avoid privileged users gaming the system.
- Include a check for the Web3AiTools address in the withdraw function
- Implement the remediation specified below for centralization risks.

Auditor's Comment: The contract code now checks for releaseTime to be greater than contractCreatedTime. This way minimal values cannot be passed in and thus game the claim() function.

Status

Resolved

2. Centralization Risk

Description

The contract owner can adjust the tax to as high as 100% on transactions. The owner account should be taken care to not be mismanaged else the privileges associated with the account can slip into the hands of the hacker. Also, if renouncing ownership is not done appropriately, the current contract owner can renounce ownership, and lose access to owner privileges. Functions like setting tax, and wallet addresses will be uncallable and any updates needed to be made will be inaccessible.

Remediation

Having a variable handle the maxTaxLimit will prevent the contract owner adjusting tax to a high value. Carefully manage the owner account by using a multi-signature wallet or implementing community governance to ensure that updates to the project happen by some form of consensus. A timelock can be implemented between changes as well.
The renounceOwnership() and transferOwnership() can be set up for 2-step verification to prevent mistaken privilege transfer or renouncing.

References

- Link 1
- Link 2

Status

Acknowledged

Medium Severity Issues

3. Frontrunning

Description

Upon deployment, some functions can be limited in functionality if the state variables they interact with are not properly set.

Remediation

With the current contract structure, it is recommended to call the functions in this order to mitigate risk:

- deploy contract and call initialize( )
- set all wallets, releaseTime, etc to ensure funds are sent to the right address and proper timing for release time follows.
- call distribute( )

The contract deployer has to be aware of safe deployment practices, they can refer to OpenZeppelin's guide linked: *Writing Upgradeable Contracts - OpenZeppelin Docs*

Status

Acknowledged

Web3Tool Team Remarks: The contract, when deployed via the proxy avoids this issue of frontrunning by a malicious actor.

4. Missing test cases

Description

The codebase lacks unit test coverage. It is advisable to have test coverage greater than 95% of the codebase to reduce unexpected functionality and help fuzz test as many invariants as possible.

Remediation

Include unit tests for the codebase.

Status

Acknowledged

5. Inaccurate accounting

Description

For founders and team advisors, the arithmetic calculations carried out determine how tokens will be released to users per quarter. Although the checks are implemented the transfers do not happen. When claim() is called, an update to the balances mapping occurs which causes an overproduction of the tokens; the number of tokens held by all addresses after distribution and claim, exceeds the total supply value noted by the totalSupply function. This is problematic as it leaves the tokens in the custody of the contract and not the intended recipients.

Remediation

Implement transfers using "_transfer" from the contract address which holds the allocation of the Founders and Advisors, instead of updating the balances mapping.

Status

Acknowledged

6. Locked tokens possibility

Description

Due to the logic utilized in the claim function, the Founder and Team Advisors addresses should not be the same. If the Founders and Team Advisors are the same address it would lead to 5% of the token total supply being locked in the contract until the address is updated again. With the check below, if founders == teamAdvisors only the first code block will get executed.

```
function claim() external payable onlyLockedWallets {
            ...
    if(msg.sender == founders) {
```

Remediation

Ensure the Founders address and Team Advisors address are unique.

Status

Acknowledged

7. Input sanitization / Parity checks

Description

Although underflows and overflows are checked internally in solidity versions >0.8.0, multiple logical arithmetic checks would be needed in the codebase dependent on the client. setReleaseTime(), setSupply_Team_Allocation(), setBuyandSellTax(), setTokenPrice() are likely pivots to exploit this vulnerability.

Remediation

- setReleaseTime() has been described in the exploit scenario above
- setSupply_Team_Allocation() can be done with hardcoded values because the whitepaper clearly specifies how much will be allocated to Founder, Marketing, Ecosystem, and Staking,
- setBuyandSellTax() if implemented properly could lead to users swapping and losing all their tokens because there is currently no cap to how much tax would be deducted per transaction,
- setTokenPrice() can be updated to any value without prior notice by the owner but with the values corresponding to the epoch stated in the whitepaper a struct can be implemented to avoid mistaken allocations of price by the owner.
- setProcessingFee() and setNetworkingFee() also should be checked, updated or removed if not in use.
- Add checks for address existence before updating

Status

Acknowledged

Low Severity Issues

8. Unused code

Description

The contract declares IPancakeFactory, IPancakeRouter, and IPancakePair but the current version of the codebase does not put this functionality to use.

Remediation

Consider cleaning up the contract by removing the unused interfaces, implementing said functionality, and deleting dead code to help reduce the contract size and aid deployment.

Status

Acknowledged

9. Wrong checks

Description

The erc20 withdraw function checks for an ether balance in the contract using

```
require(address(this).balance > 0, 'Pre-Sale: Contract has no money');
```

instead of checking for the ERC20 token balance to be withdrawn.

Remediation

Check for erc20 token balances instead and exclude the contract address from the function as well.

Auditor's Comment: ERC20 withdrawals no longer check if the contract has an ether balance before erc20 token withdrawals are made.

Status

Resolved

Informational Issues

10. Event emission

Description

Changes to state in deployed contracts can be easily tracked using events, especially when logs would need to be kept. The contract allows multiple key variables to be reset and it would be advisable to emit events when this occurs.

Remediation

Emit events and index them (where needed) to keep track of critical changes and aid searches for functions that update state.

Status

Acknowledged

11. Unlocked pragma

Description

The solidity pragma version in this codebase is unlocked.

Remediation

It is advised to use a specific Solidity version when deploying to production to reduce the surface of attacks with future releases which were not accounted for when the contracts originally went live.

Status

Acknowledged

12. Unnecessary access control

Description

The internal "_mint()" function is only called by the distribute() function which has the onlyOwner modifier attached to it as well. Calling it again internally will lead to extra costs spent on gas which can be avoided.

Remediation

Consider removing the onlyOwner modifier in the "_mint" function.

Status

Acknowledged

13. Low code and documentation readability

Description

The docs say tax is 0 but it's possible to set tax to any arbitrary value, even 100 which would make users lose all their tokens when they perform a transfer/swap if the PancakeSwap interface gets implemented as expected. The codebase contains a few spelling errors, and does not conform to the recommended Solidity standard of code formatting.

Remediation

Ensure that the contract code matches the documentation provided. To improve the ease of reading the codebase, it can be passed through a spellchecker, and the tool “Prettier” can be used on the contract to aid formatting.

Status

Acknowledged

14. General Recommendation

The privileges of the contract owner stretches towards burning tokens in the accounts of token holders. This is unsafe when attackers overrule contract owners and use this privilege to burn users' tokens. It is recommended to let users have the right to burn tokens in their possession.

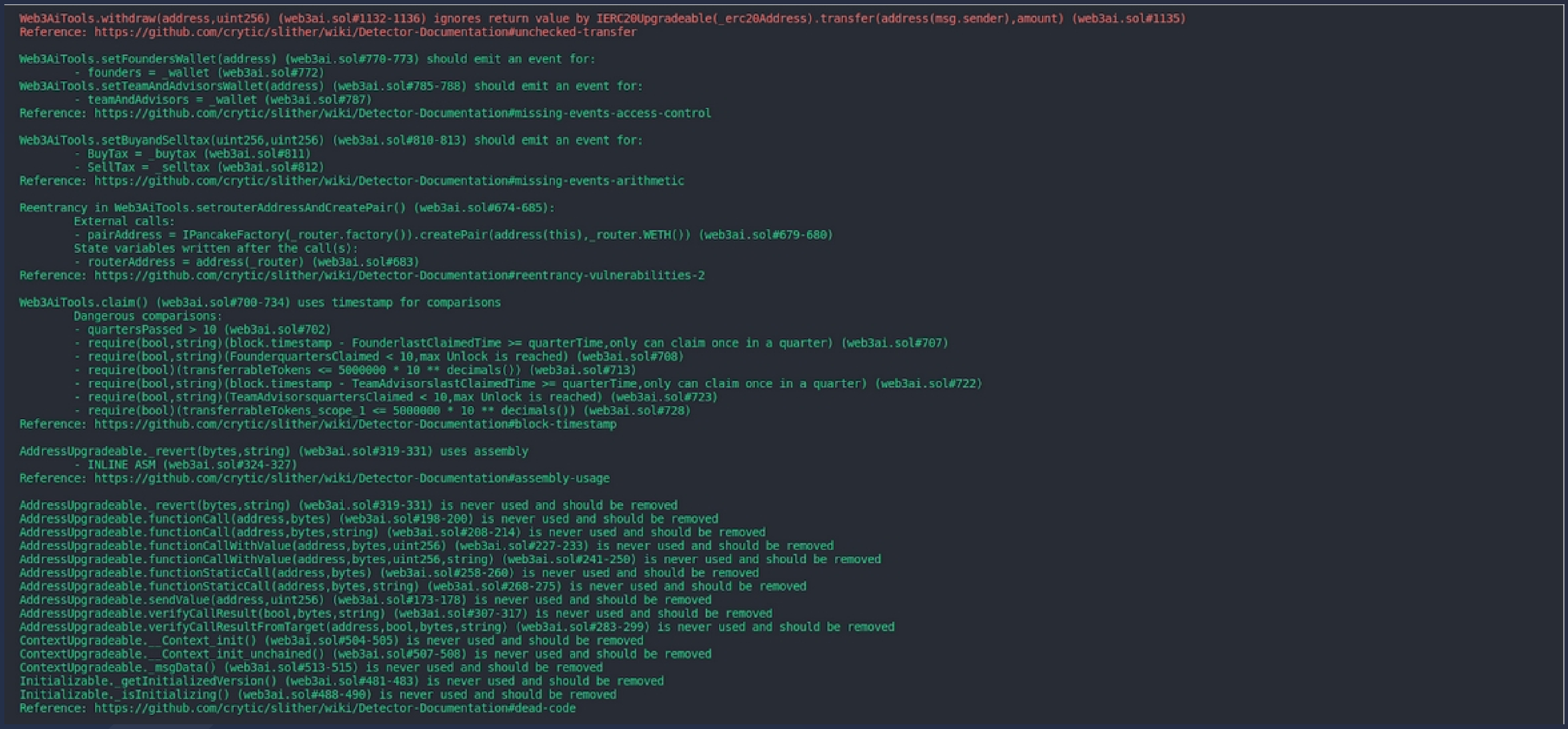
Functional Testing

Some of the tests performed are mentioned below:

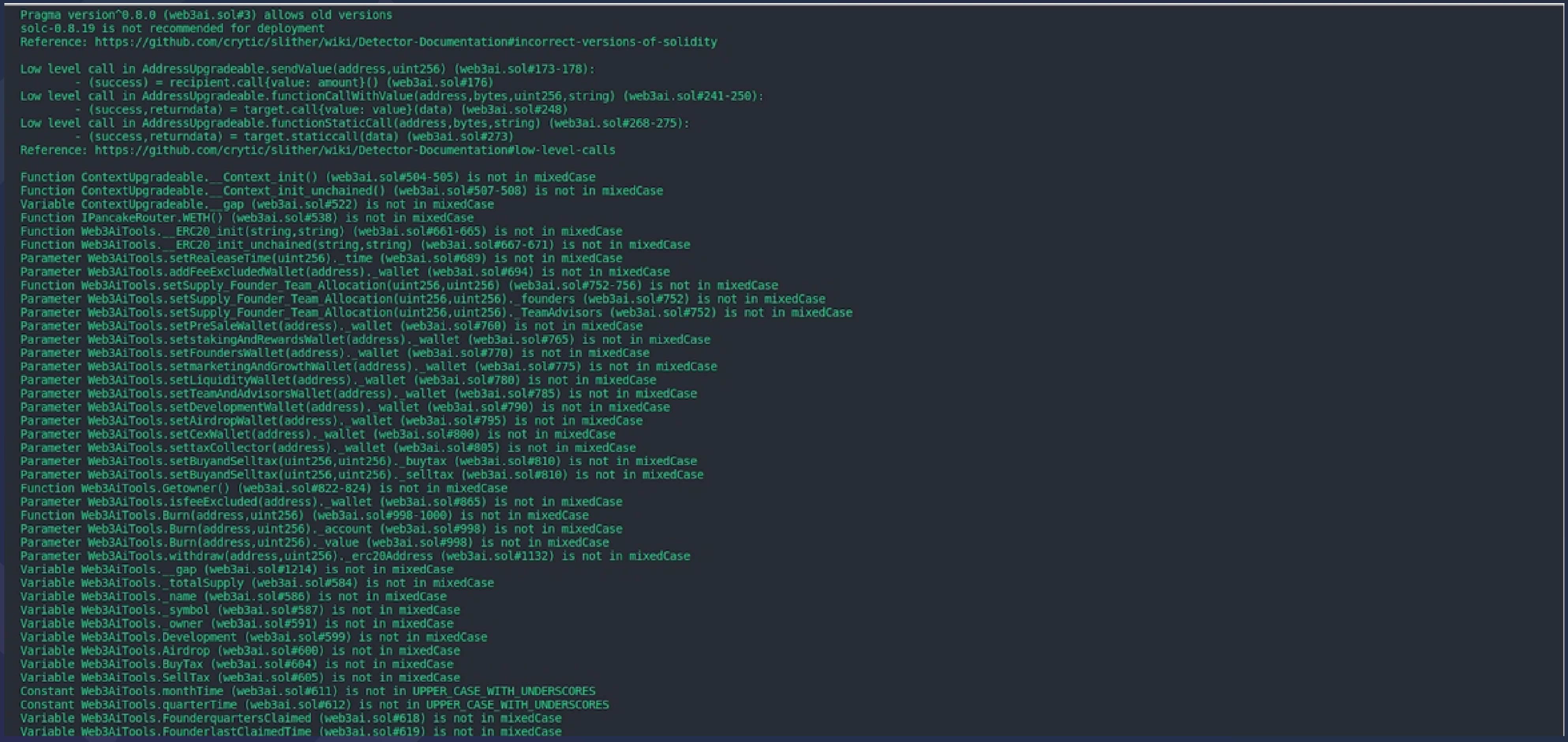
- ✓ Should get the name of the token
- ✓ Should get the symbol of the token
- ✓ Should verify all assigned addresses
- ✓ Should confirm balances of key holders after distribution
- ✓ Should allow the contract owner to successfully distribute all tokens and send 10% of the 100 million to the contract.
- ✓ Should revert when unapproved addresses call the claim function.
- ✓ Should allow founder and team advisors claim tokens
- ✓ Should allow contract owner withdraw ethers in contract
- ✓ Should successfully transfer all tokens to recipient when no sell tax is set
- ✓ Should successfully transfer all tokens to recipient when the sell tax is set
- ✓ Should successfully transfer all tokens to the pair address when no buy tax is set
- ✓ Should successfully approve tokens to spenders before they call the transferFrom function
- ✓ Should transfer tokens without a tax when added to the list of addresses to exclude
- ✓ Should successfully transfer all tokens to the pair address when the buy tax is set
- ✓ Should transfer tokens without a tax when added to the list of addresses to exclude
- ✓ Should transfer tokens to the DEX and increase tax collector balance
- ✓ Should allow spender transfer tokens to the DEX and increase tax collector balance
- ✓ Should allow DEX send to users and increase tax collector balance

Automated Tests

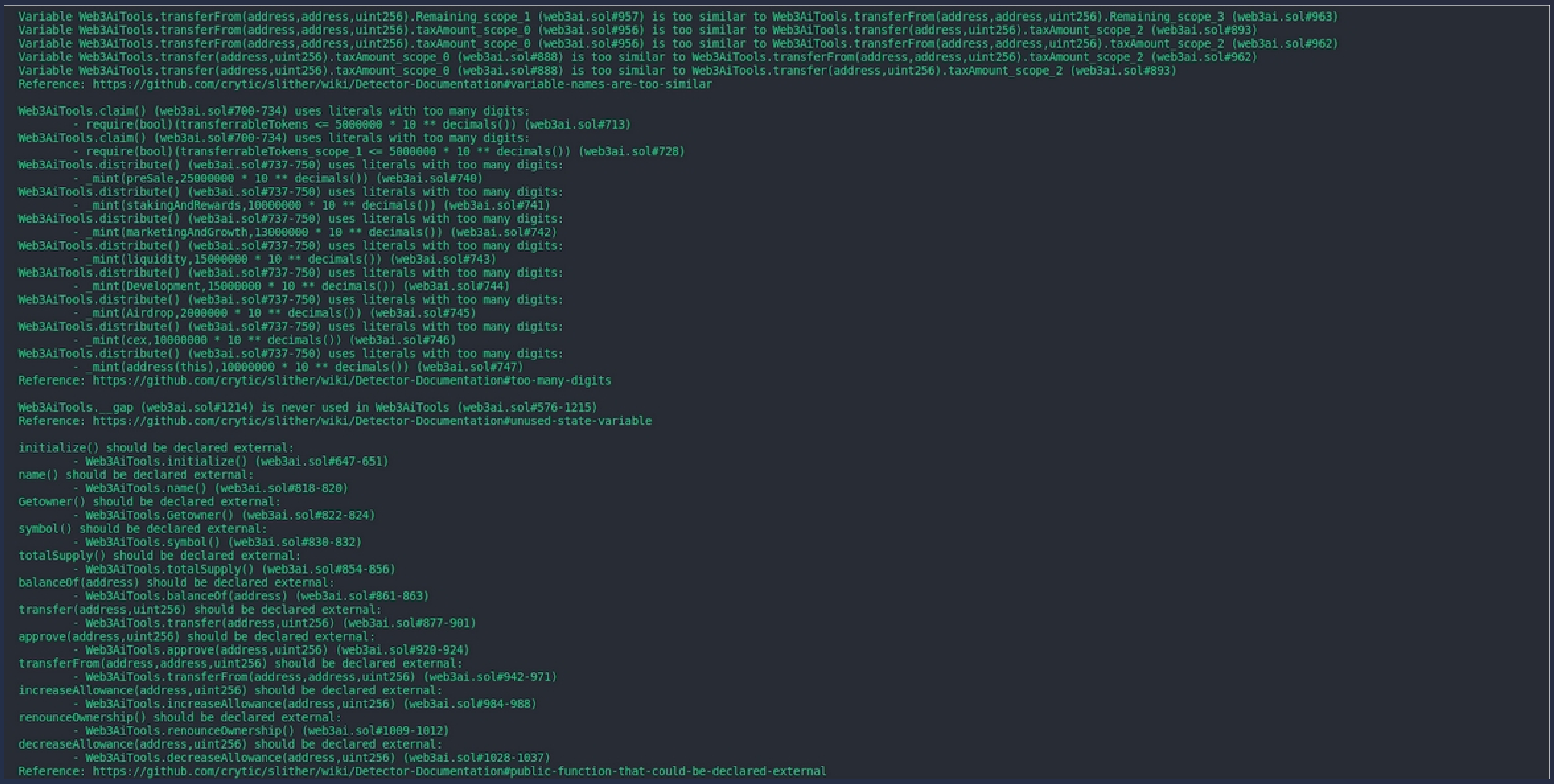
No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.



```
Web3AiTools.withdraw(address,uint256) (web3ai.sol#1132-1136) ignores return value by IERC20Upgradeable(_erc20Address).transfer(address(msg.sender),amount) (web3ai.sol#1135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Web3AiTools.setFoundersWallet(address) (web3ai.sol#770-773) should emit an event for:
        - founders = _wallet (web3ai.sol#772)
Web3AiTools.setTeamAndAdvisorsWallet(address) (web3ai.sol#785-788) should emit an event for:
        - teamAndAdvisors = _wallet (web3ai.sol#787)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

Web3AiTools.setBuyandSelltax(uint256,uint256) (web3ai.sol#810-813) should emit an event for:
        - BuyTax = _buytax (web3ai.sol#811)
        - SellTax = _selltax (web3ai.sol#812)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Reentrancy in Web3AiTools.setrouterAddressAndCreatePair() (web3ai.sol#674-685):
        External calls:
        - pairAddress = IPancakeFactory(_router.factory()).createPair(address(this),_router.WETH()) (web3ai.sol#679-680)
        State variables written after the call(s):
        - routerAddress = address(_router) (web3ai.sol#683)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Web3AiTools.claim() (web3ai.sol#700-734) uses timestamp for comparisons
        Dangerous comparisons:
        - quartersPassed > 10 (web3ai.sol#702)
        - require(bool,string)(block.timestamp - FounderlastClaimedTime >= quarterTime,only can claim once in a quarter) (web3ai.sol#707)
        - require(bool,string)(FounderquartersClaimed < 10,max Unlock is reached) (web3ai.sol#708)
        - require(bool)(transferrableTokens <= 5000000 * 10 ** decimals()) (web3ai.sol#713)
        - require(bool,string)(block.timestamp - TeamAdvisorslastClaimedTime >= quarterTime,only can claim once in a quarter) (web3ai.sol#722)
        - require(bool,string)(TeamAdvisorsquartersClaimed < 10,max Unlock is reached) (web3ai.sol#723)
        - require(bool)(transferrableTokens_scope_1 <= 5000000 * 10 ** decimals()) (web3ai.sol#728)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

AddressUpgradeable._revert(bytes,string) (web3ai.sol#319-331) uses assembly
        - INLINE ASM (web3ai.sol#324-327)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

AddressUpgradeable._revert(bytes,string) (web3ai.sol#319-331) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (web3ai.sol#198-200) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (web3ai.sol#208-214) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (web3ai.sol#227-233) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (web3ai.sol#241-250) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (web3ai.sol#258-260) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (web3ai.sol#268-275) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (web3ai.sol#173-178) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (web3ai.sol#307-317) is never used and should be removed
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (web3ai.sol#283-299) is never used and should be removed
ContextUpgradeable.__Context_init() (web3ai.sol#504-505) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (web3ai.sol#507-508) is never used and should be removed
ContextUpgradeable._msgData() (web3ai.sol#513-515) is never used and should be removed
Initializable._getInitializedVersion() (web3ai.sol#481-483) is never used and should be removed
Initializable._isInitializing() (web3ai.sol#488-490) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



```
Pragma version^0.8.0 (web3ai.sol#3) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AddressUpgradeable.sendValue(address,uint256) (web3ai.sol#173-178):
        - (success) = recipient.call{value: amount}() (web3ai.sol#176)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (web3ai.sol#241-250):
        - (success,returndata) = target.call{value: value}(data) (web3ai.sol#248)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (web3ai.sol#268-275):
        - (success,returndata) = target.staticcall(data) (web3ai.sol#273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function ContextUpgradeable.__Context_init() (web3ai.sol#504-505) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (web3ai.sol#507-508) is not in mixedCase
Variable ContextUpgradeable.__gap (web3ai.sol#522) is not in mixedCase
Function IPancakeRouter.WETH() (web3ai.sol#538) is not in mixedCase
Function Web3AiTools.__ERC20_init(string,string) (web3ai.sol#661-665) is not in mixedCase
Function Web3AiTools.__ERC20_init_unchained(string,string) (web3ai.sol#667-671) is not in mixedCase
Parameter Web3AiTools.setRealeaseTime(uint256)._time (web3ai.sol#689) is not in mixedCase
Parameter Web3AiTools.addFeeExcludedWallet(address)._wallet (web3ai.sol#694) is not in mixedCase
Function Web3AiTools.setSupply_Founder_Team_Allocation(uint256,uint256) (web3ai.sol#752-756) is not in mixedCase
Parameter Web3AiTools.setSupply_Founder_Team_Allocation(uint256,uint256)._founders (web3ai.sol#752) is not in mixedCase
Parameter Web3AiTools.setSupply_Founder_Team_Allocation(uint256,uint256)._TeamAdvisors (web3ai.sol#752) is not in mixedCase
Parameter Web3AiTools.setPreSaleWallet(address)._wallet (web3ai.sol#760) is not in mixedCase
Parameter Web3AiTools.setstakingAndRewardsWallet(address)._wallet (web3ai.sol#765) is not in mixedCase
Parameter Web3AiTools.setFoundersWallet(address)._wallet (web3ai.sol#770) is not in mixedCase
Parameter Web3AiTools.setmarketingAndGrowthWallet(address)._wallet (web3ai.sol#775) is not in mixedCase
Parameter Web3AiTools.setLiquidityWallet(address)._wallet (web3ai.sol#780) is not in mixedCase
Parameter Web3AiTools.setTeamAndAdvisorsWallet(address)._wallet (web3ai.sol#785) is not in mixedCase
Parameter Web3AiTools.setDevelopmentWallet(address)._wallet (web3ai.sol#790) is not in mixedCase
Parameter Web3AiTools.setAirdropWallet(address)._wallet (web3ai.sol#795) is not in mixedCase
Parameter Web3AiTools.setCexWallet(address)._wallet (web3ai.sol#800) is not in mixedCase
Parameter Web3AiTools.settaxCollector(address)._wallet (web3ai.sol#805) is not in mixedCase
Parameter Web3AiTools.setBuyandSelltax(uint256,uint256)._buytax (web3ai.sol#810) is not in mixedCase
Parameter Web3AiTools.setBuyandSelltax(uint256,uint256)._selltax (web3ai.sol#810) is not in mixedCase
Function Web3AiTools.Getowner() (web3ai.sol#822-824) is not in mixedCase
Parameter Web3AiTools.isfeeExcluded(address)._wallet (web3ai.sol#865) is not in mixedCase
Function Web3AiTools.Burn(address,uint256) (web3ai.sol#998-1000) is not in mixedCase
Parameter Web3AiTools.Burn(address,uint256)._account (web3ai.sol#998) is not in mixedCase
Parameter Web3AiTools.Burn(address,uint256)._value (web3ai.sol#998) is not in mixedCase
Parameter Web3AiTools.withdraw(address,uint256)._erc20Address (web3ai.sol#1132) is not in mixedCase
Variable Web3AiTools.__gap (web3ai.sol#1214) is not in mixedCase
Variable Web3AiTools._totalSupply (web3ai.sol#584) is not in mixedCase
Variable Web3AiTools._name (web3ai.sol#586) is not in mixedCase
Variable Web3AiTools._symbol (web3ai.sol#587) is not in mixedCase
Variable Web3AiTools._owner (web3ai.sol#591) is not in mixedCase
Variable Web3AiTools.Development (web3ai.sol#599) is not in mixedCase
Variable Web3AiTools.Airdrop (web3ai.sol#600) is not in mixedCase
Variable Web3AiTools.BuyTax (web3ai.sol#604) is not in mixedCase
Variable Web3AiTools.SellTax (web3ai.sol#605) is not in mixedCase
Constant Web3AiTools.monthTime (web3ai.sol#611) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Web3AiTools.quarterTime (web3ai.sol#612) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Web3AiTools.FounderquartersClaimed (web3ai.sol#618) is not in mixedCase
Variable Web3AiTools.FounderlastClaimedTime (web3ai.sol#619) is not in mixedCase
```



```
Variable Web3AiTools.transferFrom(address,address,uint256).Remaining_scope_1 (web3ai.sol#957) is too similar to Web3AiTools.transferFrom(address,address,uint256).Remaining_scope_3 (web3ai.sol#963)
Variable Web3AiTools.transferFrom(address,address,uint256).taxAmount_scope_0 (web3ai.sol#956) is too similar to Web3AiTools.transfer(address,uint256).taxAmount_scope_2 (web3ai.sol#893)
Variable Web3AiTools.transferFrom(address,address,uint256).taxAmount_scope_0 (web3ai.sol#956) is too similar to Web3AiTools.transferFrom(address,address,uint256).taxAmount_scope_2 (web3ai.sol#962)
Variable Web3AiTools.transfer(address,uint256).taxAmount_scope_0 (web3ai.sol#888) is too similar to Web3AiTools.transferFrom(address,address,uint256).taxAmount_scope_2 (web3ai.sol#962)
Variable Web3AiTools.transfer(address,uint256).taxAmount_scope_0 (web3ai.sol#888) is too similar to Web3AiTools.transfer(address,uint256).taxAmount_scope_2 (web3ai.sol#893)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

Web3AiTools.claim() (web3ai.sol#700-734) uses literals with too many digits:
        - require(bool)(transferrableTokens <= 5000000 * 10 ** decimals()) (web3ai.sol#713)
Web3AiTools.claim() (web3ai.sol#700-734) uses literals with too many digits:
        - require(bool)(transferrableTokens_scope_1 <= 5000000 * 10 ** decimals()) (web3ai.sol#728)
Web3AiTools.distribute() (web3ai.sol#737-750) uses literals with too many digits:
        - _mint(preSale,25000000 * 10 ** decimals()) (web3ai.sol#740)
Web3AiTools.distribute() (web3ai.sol#737-750) uses literals with too many digits:
        - _mint(stakingAndRewards,10000000 * 10 ** decimals()) (web3ai.sol#741)
Web3AiTools.distribute() (web3ai.sol#737-750) uses literals with too many digits:
        - _mint(marketingAndGrowth,13000000 * 10 ** decimals()) (web3ai.sol#742)
Web3AiTools.distribute() (web3ai.sol#737-750) uses literals with too many digits:
        - _mint(liquidity,15000000 * 10 ** decimals()) (web3ai.sol#743)
Web3AiTools.distribute() (web3ai.sol#737-750) uses literals with too many digits:
        - _mint(Development,15000000 * 10 ** decimals()) (web3ai.sol#744)
Web3AiTools.distribute() (web3ai.sol#737-750) uses literals with too many digits:
        - _mint(Airdrop,2000000 * 10 ** decimals()) (web3ai.sol#745)
Web3AiTools.distribute() (web3ai.sol#737-750) uses literals with too many digits:
        - _mint(cex,10000000 * 10 ** decimals()) (web3ai.sol#746)
Web3AiTools.distribute() (web3ai.sol#737-750) uses literals with too many digits:
        - _mint(address(this),10000000 * 10 ** decimals()) (web3ai.sol#747)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

Web3AiTools.__gap (web3ai.sol#1214) is never used in Web3AiTools (web3ai.sol#576-1215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

initialize() should be declared external:
        - Web3AiTools.initialize() (web3ai.sol#647-651)
name() should be declared external:
        - Web3AiTools.name() (web3ai.sol#818-820)
Getowner() should be declared external:
        - Web3AiTools.Getowner() (web3ai.sol#822-824)
symbol() should be declared external:
        - Web3AiTools.symbol() (web3ai.sol#830-832)
totalSupply() should be declared external:
        - Web3AiTools.totalSupply() (web3ai.sol#854-856)
balanceOf(address) should be declared external:
        - Web3AiTools.balanceOf(address) (web3ai.sol#861-863)
transfer(address,uint256) should be declared external:
        - Web3AiTools.transfer(address,uint256) (web3ai.sol#877-901)
approve(address,uint256) should be declared external:
        - Web3AiTools.approve(address,uint256) (web3ai.sol#920-924)
transferFrom(address,address,uint256) should be declared external:
        - Web3AiTools.transferFrom(address,address,uint256) (web3ai.sol#942-971)
increaseAllowance(address,uint256) should be declared external:
        - Web3AiTools.increaseAllowance(address,uint256) (web3ai.sol#984-988)
renounceOwnership() should be declared external:
        - Web3AiTools.renounceOwnership() (web3ai.sol#1009-1012)
decreaseAllowance(address,uint256) should be declared external:
        - Web3AiTools.decreaseAllowance(address,uint256) (web3ai.sol#1028-1037)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Closing Summary

In this report, we have considered the security of the Web3AiTools codebase. We performed our audit according to the procedure described above.

Some issues of High, Medium, Low and Informational severity were found. Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Web3AiTools Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Web3AiTools Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+
Audits Completed

$16B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
April, 2023

For

web3AiTools

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com