



# Cyberscope

## Audit Report

# **AI Baby Doge**

February 2023

Type	BEP20
Network	BSC
Address	0x9183818c7d34820950DB6A836d51d52ebFE04379
Audited by	© cyberscope

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Review</b>	<b>3</b>
<b>Audit Updates</b>	<b>3</b>
<b>Source Files</b>	<b>3</b>
<b>Analysis</b>	<b>4</b>
<b>ULTW - Transfers Liquidity to Team Wallet</b>	<b>5</b>
<b>Description</b>	<b>5</b>
<b>Recommendation</b>	<b>5</b>
<b>Diagnostics</b>	<b>6</b>
<b>PVC - Price Volatility Concern</b>	<b>7</b>
<b>Description</b>	<b>7</b>
<b>Recommendation</b>	<b>7</b>
<b>PTRP - Potential Transfer Revert Propagation</b>	<b>8</b>
<b>Description</b>	<b>8</b>
<b>Recommendation</b>	<b>8</b>
<b>L02 - State Variables could be Declared Constant</b>	<b>9</b>
<b>Description</b>	<b>9</b>
<b>Recommendation</b>	<b>9</b>
<b>L04 - Conformance to Solidity Naming Conventions</b>	<b>10</b>
<b>Description</b>	<b>10</b>
<b>Recommendation</b>	<b>10</b>
<b>L07 - Missing Events Arithmetic</b>	<b>12</b>
<b>Description</b>	<b>12</b>
<b>Recommendation</b>	<b>12</b>
<b>L09 - Dead Code Elimination</b>	<b>13</b>
<b>Description</b>	<b>13</b>
<b>Recommendation</b>	<b>13</b>
<b>L16 - Validate Variable Setters</b>	<b>15</b>
<b>Description</b>	<b>15</b>
<b>Recommendation</b>	<b>15</b>
<b>L19 - Stable Compiler Version</b>	<b>16</b>
<b>Description</b>	<b>16</b>

<b>Recommendation</b>	<b>16</b>
<b>L20 - Succeeded Transfer Check</b>	<b>17</b>
<b>Description</b>	<b>17</b>
<b>Recommendation</b>	<b>17</b>
<b>Functions Analysis</b>	<b>18</b>
<b>Inheritance Graph</b>	<b>23</b>
<b>Flow Graph</b>	<b>24</b>
<b>Summary</b>	<b>25</b>
<b>Disclaimer</b>	<b>26</b>
<b>About Cyberscope</b>	<b>27</b>

## Review

Contract Name	AIBBDOGE
Compiler Version	v0.8.7+commit.e28d00a7
Optimization	200 runs
Explorer	<a href="https://bscscan.com/address/0x9183818c7d34820950db6a836d51d52ebfe04379">https://bscscan.com/address/0x9183818c7d34820950db6a836d51d52ebfe04379</a>
Address	0x9183818c7d34820950db6a836d51d52ebfe04379
Network	BSC
Symbol	AiBabyDoge
Decimals	18
Total Supply	420.000.000.000

## Audit Updates

Initial Audit	06 Feb 2023
---------------	-------------

## Source Files

Filename	SHA256
AIBBDOGE.sol	17356e3c8194a476b00706554c88f958c3bcd3ca215e8218dcb544af352c170a

# 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	Unresolved
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

## ULTW - Transfers Liquidity to Team Wallet

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L738,944
<b>Status</b>	Unresolved

### Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the `claimStuckBNB` and `forceSwap` methods.

```
function claimStuckBNB() external onlyOwner {
    sendBNB(payable(mktWallet), address(this).balance);
}
function forceSwap() external onlyOwner {
    uint256 contractTokenBalance = balanceOf(address(this));
    swapping = true;
    uint256 marketingTokens = contractTokenBalance;
    swapAndSendMarketing(marketingTokens);
    swapping = false;
}
```

### Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped. Since a huge amount may volatile the token's price. 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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

# Diagnostics

● Critical   ● Medium   ● Minor / Informative

Severity	Code	Description	Status
●	PVC	Price Volatility Concern	Unresolved
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

## PVC - Price Volatility Concern

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L956
<b>Status</b>	Unresolved

### Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapTokensAtAmount` sets a threshold where the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function setSwapTokensAtAmount(uint256 newAmount) external onlyOwner {
    require(
        newAmount > totalSupply() / 100000,
        "SwapTokensAtAmount must be greater than 0.001% of total supply"
    );
    swapTokensAtAmount = newAmount;
}
```

### Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the total supply. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.



## PTRP - Potential Transfer Revert Propagation

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L752
<b>Status</b>	Unresolved

### Description

The contract sends funds to a `marketingWallet` as part of the transfer flow. This address can either be a wallet address or a contract. If the address is a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
function sendBNB(address payable recipient, uint256 amount) internal {
    require(
        address(this).balance >= amount,
        "Address: insufficient balance"
    );

    (bool success, ) = recipient.call{value: amount}("");
    require(
        success,
        "Address: unable to send value, recipient may have reverted"
    );
}
```

### Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be achieved by not allowing set contract addresses or by sending the funds in a non-revertable way.

## L02 - State Variables could be Declared Constant

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L645,652
<b>Status</b>	Unresolved

### Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 public tradecooldown  
address private DEAD = 0x00000000000000000000000000000000dEaD
```

### Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L355,357,388,434,652,668,811,830
<b>Status</b>	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 DOMAIN_SEPARATOR() external view returns (bytes32);
function PERMIT_TYPEHASH() external pure returns (bytes32);
function MINIMUM_LIQUIDITY() external pure returns (uint256);
function WETH() external pure returns (address);
address private DEAD = 0x00000000000000000000000000000000dEaD;
event wallet1Changed(address wallet1);
uint256 _buyFee
uint256 _sellFee
address _mktWallet
```

### 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>.

## L07 - Missing Events Arithmetic

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L961
<b>Status</b>	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.

```
swapTokensAtAmount = newAmount
```

### 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.

## L09 - Dead Code Elimination

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L250,742,892
<b>Status</b>	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];
    ...
}
_totalSupply -= amount;

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

_afterTokenTransfer(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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L698,768,835
<b>Status</b>	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.

```
mktWallet = address(_mktWallet)
uniswapV2Pair = _uniswapV2Pair
mktWallet = _mktWallet
```

### 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.



## L19 - Stable Compiler Version

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L2
<b>Status</b>	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 ^0.8.7;
```

### 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.

## L20 - Succeeded Transfer Check

<b>Criticality</b>	Minor / Informative
<b>Location</b>	AIBBDOGE.sol#L735
<b>Status</b>	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.

```
ERC20token.transfer(msg.sender, balance)
```

### 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
<b>IERC20</b>	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>IERC20Metadata</b>	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
<b>Context</b>	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
<b>Ownable</b>	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner

ERC20	Implementation	Context, IERC20, IERC20Met adata		
		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	✓	
	_afterTokenTransfer	Internal	✓	
<b>IUniswapV2Factory</b>	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-

	setFeeToSetter	External	✓	-
<b>IUniswapV2Pair</b>	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-

	initialize	External	✓	-
<b>IUniswapV2Router01</b>	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-
<b>IUniswapV2Router02</b>	Interface	IUniswapV2 Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-

	swapExactETHForTokensSupporting FeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupporting FeeOnTransferTokens	External	✓	-
<b>AIBBDOGE</b>	Implementation	ERC20, Ownable		
		Public	Payable	ERC20
		External	Payable	-
	claimStuckTokens	External	✓	onlyOwner
	claimStuckBNB	External	✓	onlyOwner
	isContract	Internal		
	sendBNB	Internal	✓	
	updateUniswapV2Router	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	excludeFromFees	External	✓	onlyOwner
	isExcludedFromFees	Public		-
	updateFees	External	✓	onlyOwner
	enableWalletToWalletTransferWithout Fee	External	✓	onlyOwner
	changewallets	External	✓	onlyOwner
	_transfer	Internal	✓	
	swapAndLiquify	Private	✓	
	swapAndSendMarketing	Private	✓	
	forceSwap	External	✓	onlyOwner
	setAutoSwap	External	✓	onlyOwner
	setSwapTokensAtAmount	External	✓	onlyOwner

# Inheritance Graph





# Flow Graph



## Summary

There are some functions that can be abused by the owner like transfer funds to the team's wallet. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 15% 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>