



Cyberscope

# Audit Report

## **Starship**

February 2023

Type	BEP20
Network	BSC
Address	0xe4b6db3241347de559bfa6013224a87f97a294f7
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>BC - Blacklists Addresses</b>	<b>5</b>
<b>Description</b>	<b>5</b>
<b>Recommendation</b>	<b>5</b>
<b>Diagnostics</b>	<b>6</b>
<b>CR - Code Repetition</b>	<b>8</b>
<b>Description</b>	<b>8</b>
<b>Recommendation</b>	<b>8</b>
<b>MVN - Misleading Variables Naming</b>	<b>10</b>
<b>Description</b>	<b>10</b>
<b>Recommendation</b>	<b>10</b>
<b>PVC - Price Volatility Concern</b>	<b>11</b>
<b>Description</b>	<b>11</b>
<b>Recommendation</b>	<b>11</b>
<b>RSML - Redundant SafeMath Library</b>	<b>12</b>
<b>Description</b>	<b>12</b>
<b>Recommendation</b>	<b>12</b>
<b>L04 - Conformance to Solidity Naming Conventions</b>	<b>13</b>
<b>Description</b>	<b>13</b>
<b>Recommendation</b>	<b>14</b>
<b>L05 - Unused State Variable</b>	<b>15</b>
<b>Description</b>	<b>15</b>
<b>Recommendation</b>	<b>15</b>
<b>L07 - Missing Events Arithmetic</b>	<b>16</b>
<b>Description</b>	<b>16</b>
<b>Recommendation</b>	<b>16</b>
<b>L09 - Dead Code Elimination</b>	<b>17</b>
<b>Description</b>	<b>17</b>

<b>Recommendation</b>	<b>17</b>
<b>L12 - Using Variables before Declaration</b>	<b>18</b>
<b>Description</b>	<b>18</b>
<b>Recommendation</b>	<b>18</b>
<b>L14 - Uninitialized Variables in Local Scope</b>	<b>19</b>
<b>Description</b>	<b>19</b>
<b>Recommendation</b>	<b>19</b>
<b>L15 - Local Scope Variable Shadowing</b>	<b>20</b>
<b>Description</b>	<b>20</b>
<b>Recommendation</b>	<b>20</b>
<b>L16 - Validate Variable Setters</b>	<b>21</b>
<b>Description</b>	<b>21</b>
<b>Recommendation</b>	<b>21</b>
<b>L17 - Usage of Solidity Assembly</b>	<b>22</b>
<b>Description</b>	<b>22</b>
<b>Recommendation</b>	<b>22</b>
<b>L19 - Stable Compiler Version</b>	<b>23</b>
<b>Description</b>	<b>23</b>
<b>Recommendation</b>	<b>23</b>
<b>L20 - Succeeded Transfer Check</b>	<b>24</b>
<b>Description</b>	<b>24</b>
<b>Recommendation</b>	<b>24</b>
<b>Functions Analysis</b>	<b>25</b>
<b>Inheritance Graph</b>	<b>36</b>
<b>Flow Graph</b>	<b>37</b>
<b>Summary</b>	<b>38</b>
<b>Disclaimer</b>	<b>39</b>
<b>About Cyberscope</b>	<b>40</b>

## Review

Contract Name	BABYTOKEN
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	<a href="https://bscscan.com/address/0xe4b6db3241347de559bfa6013224a87f97a294f7">https://bscscan.com/address/0xe4b6db3241347de559bfa6013224a87f97a294f7</a>
Address	0xe4b6db3241347de559bfa6013224a87f97a294f7
Network	BSC
Symbol	Starship
Decimals	18
Total Supply	21,000,000

## Audit Updates

Initial Audit	18 Feb 2023
---------------	-------------

## Source Files

Filename	SHA256
BABYTOKEN.sol	6f7ecbd72a8300db89c95ece7582bafe7d dda9162eaca8b749f301f05657e03a

# 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	Unresolved

## BC - Blacklists Addresses

<b>Criticality</b>	Critical
<b>Location</b>	BABYTOKEN.sol#L2047
<b>Status</b>	Unresolved

### Description

The contract owner has the authority to massively stop addresses from transactions. The owner may take advantage of it by calling the `multipleBotlistAddress` function.

```
function multipleBotlistAddress(address[] calldata accounts, bool excluded)
public onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
        _isBlacklisted[accounts[i]] = excluded;
    }
}
```

### 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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

# Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	CR	Code Repetition	Unresolved
●	MVN	Misleading Variables Naming	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L12	Using Variables before Declaration	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved

●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved



## CR - Code Repetition

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L2273
<b>Status</b>	Unresolved

### Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
if(automatedMarketMakerPairs[from]){
    LFee = amount.mul(buyLiquidityFee).div(100);
    AmountLiquidityFee += LFee;
    RFee = amount.mul(buyTokenRewardsFee).div(100);
    AmountTokenRewardsFee += RFee;
    MFee = amount.mul(buyMarketingFee).div(100);
    AmountMarketingFee += MFee;
    DFee = amount.mul(buyDeadFee).div(100);
    fees = LFee.add(RFee).add(MFee).add(DFee);
}
if(automatedMarketMakerPairs[to]){
    LFee = amount.mul(sellLiquidityFee).div(100);
    AmountLiquidityFee += LFee;
    RFee = amount.mul(sellTokenRewardsFee).div(100);
    AmountTokenRewardsFee += RFee;
    MFee = amount.mul(sellMarketingFee).div(100);
    AmountMarketingFee += MFee;
    DFee = amount.mul(sellDeadFee).div(100);
    fees = LFee.add(RFee).add(MFee).add(DFee);
}
```

### Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help to reduce the complexity and size of the

contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

## MVN - Misleading Variables Naming

Criticality	Medium
Location	BABYTOKEN.sol#L2205
Status	Unresolved

### Description

Variables can have misleading names if their names do not accurately reflect the value they contain or the purpose they serve. The contract uses some variable names that are too generic or do not clearly convey the information stored in the variable. Misleading variable names can lead to confusion, making the code more difficult to read and understand.

The `buyDeadFee`, `sellDeadFee` and `deadWallet` intuitively means that the specific tax is burned. On the contrary, the `deadWallet` address can be changed, as a result the tokens will not be burned but moved to the specific address.

```
function setDeadWallet(address addr) public onlyOwner {  
    deadWallet = addr;  
}
```

### Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the readability of the code.

## PVC - Price Volatility Concern

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L2201
<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 amount) public onlyOwner {  
    swapTokensAtAmount = amount;  
}
```

### 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.

## RSML - Redundant SafeMath Library

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L395
<b>Status</b>	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 on 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 unnecessarily the gas consumption.

```
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>.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L92,96,105,116,121,165,670,834,835,852,1110,1115,1417,1441,1461,1462,1463,1464,1523,1530,1542,1556,1731,1915,1916,1917,1919,1920,1922,2194
<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 __Context_init() internal initializer {
    __Context_init_unchained();
}

function __Context_init_unchained() internal initializer {
}
...
    __Ownable_init_unchained();
}

function __Ownable_init_unchained() internal initializer {
    _setOwner(_msgSender());
}
uint256[49] private __gap
...
```

## 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L165,257
<b>Status</b>	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.

```
uint256[49] private __gap  
int256 private constant MAX_INT256 = ~(int256(1) << 255)
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L2198,2208,2217
<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 = amount  
buyTokenRewardsFee = rewardsFee  
sellTokenRewardsFee = rewardsFee
```

### 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>	BABYTOKEN.sol#L92,303,347,361,381,612,1282,1575
<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 __Context_init() internal initializer {
    __Context_init_unchained();
}

function abs(int256 a) internal pure returns (int256) {
    require(a != MIN_INT256);
    return a < 0 ? -a : a;
}

...
```

### 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.

## L12 - Using Variables before Declaration

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L2302
<b>Status</b>	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 iterations
uint256 claims
uint256 lastProcessedIndex
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L2264,2268,2302
<b>Status</b>	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 fees
uint256 DFee
uint256 iterations
uint256 lastProcessedIndex
uint256 claims
```

### Recommendation

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

## L15 - Local Scope Variable Shadowing

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L1463,1464,1523,1530,1542,1556,1995
<b>Status</b>	Unresolved

### Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
string memory _name
string memory _symbol
address _owner
uint256 totalSupply = totalSupply_ * (10**18)
```

### Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

## L16 - Validate Variable Setters

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L2015,2033,2064,2083,2202
<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.

```
uniswapPair = _uniswapV2Pair
payable(serviceFeeReceiver_).transfer(serviceFee_)
_marketingWalletAddress = wallet
deadWallet = addr
```

### 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.

## L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	BABYTOKEN.sol#L330,348,366
Status	Unresolved

## Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {
    let ptr := mload(0x40)
    mstore(ptr,
0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000000000000000000000)
    mstore(add(ptr, 0x14), shl(0x60, implementation))
    mstore(add(ptr, 0x28),
0x5af43d82803e903d91602b57fd5bf300000000000000000000000000000000)
    instance := create(0, ptr, 0x37)
    ...
assembly {
    let ptr := mload(0x40)
    mstore(ptr,
0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000000000000000000000)
    mstore(add(ptr, 0x14), shl(0x60, implementation))
    mstore(add(ptr, 0x28),
0x5af43d82803e903d91602b57fd5bf300000000000000000000000000000000)
    instance := create2(0, ptr, 0x37, salt)
}
    ...
}
```

## Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

## L19 - Stable Compiler Version

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L10
<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 ^0.8.0;
```

### 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.



## L20 - Succeeded Transfer Check

<b>Criticality</b>	Minor / Informative
<b>Location</b>	BABYTOKEN.sol#L1654,2315
<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.

```
IERC20(token).transfer(to, amount)
IERC20(rewardToken).transfer(_marketingWalletAddress, newBalance)
```

### 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>Context</b>	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
<b>Ownable</b>	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	getTime	Public		-
	_transferOwnership	Internal	✓	
<b>Initializable</b>	Implementation			
<b>ContextUpgradeable</b>	Implementation	Initializable		
	__Context_init	Internal	✓	initializer
	__Context_init_unchained	Internal	✓	initializer
	_msgSender	Internal		
	_msgData	Internal		
<b>OwnableUpgradeable</b>	Implementation	Initializable, ContextUpgradeable		
	__Ownable_init	Internal	✓	initializer

	__Ownable_init_unchained	Internal	✓	initializer
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	
<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>SafeMath</b>	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		

<b>SafeMathInt</b>	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		
	add	Internal		
	abs	Internal		
	toUint256Safe	Internal		
<b>SafeMathUint</b>	Library			
	toInt256Safe	Internal		
<b>Clones</b>	Library			
	clone	Internal	✓	
	cloneDeterministic	Internal	✓	
	predictDeterministicAddress	Internal		
	predictDeterministicAddress	Internal		
<b>ERC20</b>	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	✓	
<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	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
<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>IterableMapping</b>	Library			
	get	Public		-
	getIndexOfKey	Public		-
	getKeyAtIndex	Public		-
	size	Public		-
	set	Public	✓	-
	remove	Public	✓	-
<b>DividendPayingTokenInterface</b>	Interface			

e				
	dividendOf	External		-
	withdrawDividend	External	✓	-
<b>DividendPayingTokenOptionalInterface</b>	Interface			
	withdrawableDividendOf	External		-
	withdrawnDividendOf	External		-
	accumulativeDividendOf	External		-
<b>IERC20Upgradeable</b>	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>IERC20MetadataUpgradeable</b>	Interface	IERC20Upgradeable		
	name	External		-
	symbol	External		-
	decimals	External		-
<b>ERC20Upgradeable</b>	Implementation	Initializable, ContextUpgradeable, IERC20Upgradeable, IERC20MetadataUpgradeable		
	__ERC20_init	Internal	✓	initializer



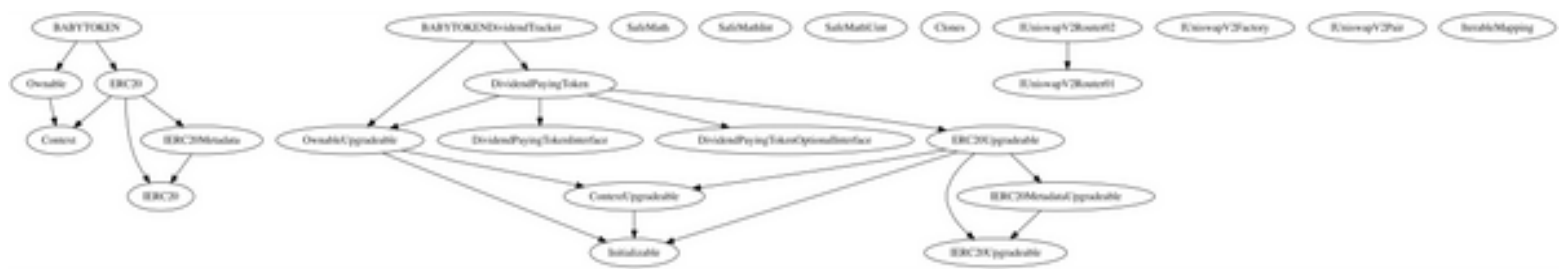
	__ERC20_init_unchained	Internal	✓	initializer
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
<b>DividendPayingToken</b>	Implementation	ERC20Upgradable, OwnableUpgradable, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
	__DividendPayingToken_init	Internal	✓	initializer
	distributeCAKEDividends	Public	✓	onlyOwner
	withdrawDividend	Public	✓	-
	_withdrawDividendOfUser	Internal	✓	

	dividendOf	Public		-
	withdrawableDividendOf	Public		-
	withdrawnDividendOf	Public		-
	accumulativeDividendOf	Public		-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_setBalance	Internal	✓	
<b>BABYTOKENDi videndTracker</b>	Implementation	OwnableUp gradeable, DividendPay ingToken		
	claimToken	External	✓	onlyOwner
	initialize	External	✓	initializer
	_transfer	Internal		
	withdrawDividend	Public		-
	excludeFromDividends	External	✓	onlyOwner
	isExcludedFromDividends	Public		-
	updateClaimWait	External	✓	onlyOwner
	updateMinimumTokenBalanceForDividends	External	✓	onlyOwner
	getLastProcessedIndex	External		-
	getNumberOfTokenHolders	External		-
	getAccount	Public		-
	getAccountAtIndex	Public		-
	canAutoClaim	Private		
	setBalance	External	✓	onlyOwner
	process	Public	✓	-
	processAccount	Public	✓	onlyOwner

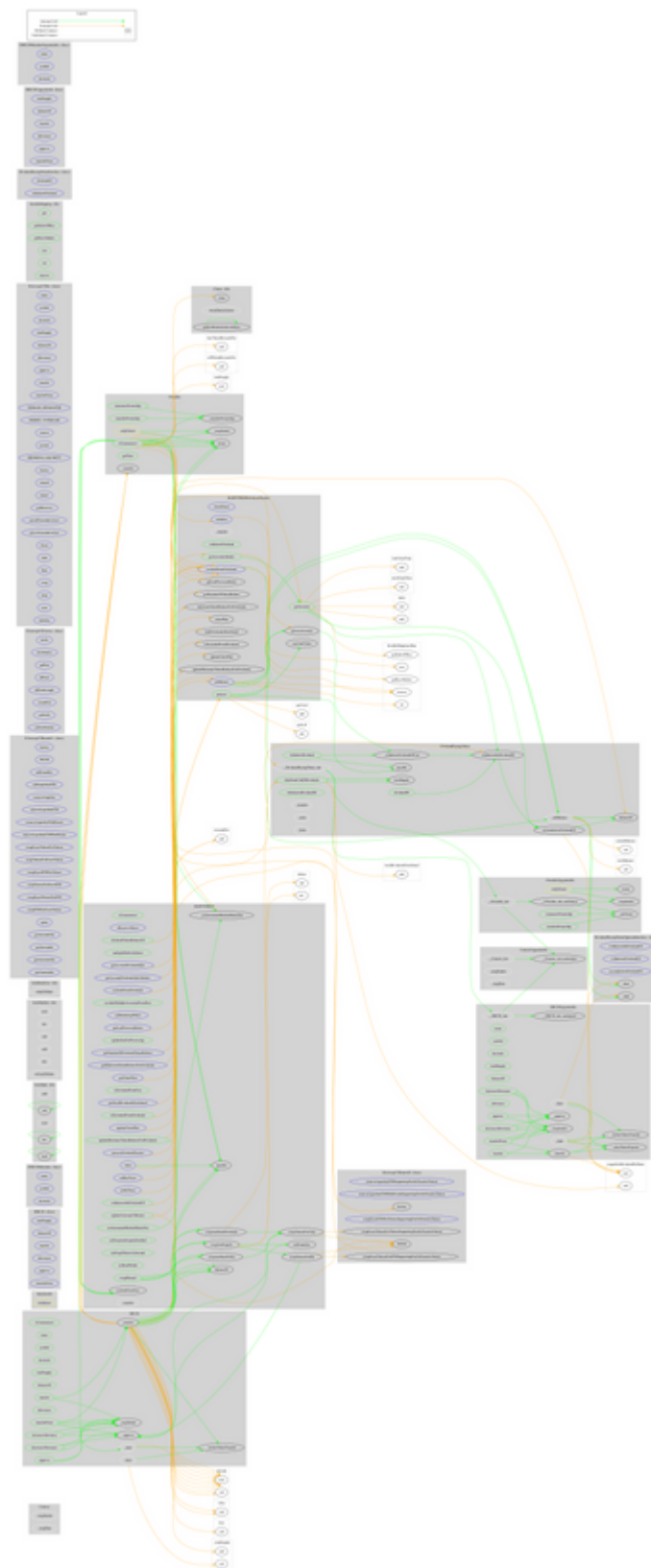
BABYTOKEN	Implementation	ERC20, Ownable		
		Public	Payable	ERC20
		External	Payable	-
	updateMinimumTokenBalanceForDividends	Public	✓	onlyOwner
	multipleBotlistAddress	Public	✓	onlyOwner
	getMinimumTokenBalanceForDividends	External		-
	updateUniswapV2Router	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	excludeMultipleAccountsFromFees	Public	✓	onlyOwner
	setMarketingWallet	External	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	updateGasForProcessing	Public	✓	onlyOwner
	updateClaimWait	External	✓	onlyOwner
	getClaimWait	External		-
	getTotalDividendsDistributed	External		-
	isExcludedFromFees	Public		-
	withdrawableDividendOf	Public		-
	dividendTokenBalanceOf	Public		-
	excludeFromDividends	External	✓	onlyOwner
	isExcludedFromDividends	Public		-
	getAccountDividendsInfo	External		-
	getAccountDividendsInfoAtIndex	External		-
	processDividendTracker	External	✓	-
	claim	External	✓	-
	getLastProcessedIndex	External		-
	getNumberOfDividendTokenHolders	External		-
	swapManual	Public	✓	onlyOwner

	setSwapAndLiquifyEnabled	Public	✓	onlyOwner
	setSwapTokensAtAmount	Public	✓	onlyOwner
	setDeadWallet	Public	✓	onlyOwner
	setBuyTaxes	External	✓	onlyOwner
	setSellTaxes	External	✓	onlyOwner
	_transfer	Internal	✓	
	swapAndSendToFee	Private	✓	
	swapAndLiquify	Private	✓	
	swapTokensForEth	Private	✓	
	swapTokensForCake	Private	✓	
	addLiquidity	Private	✓	
	swapAndSendDividends	Private	✓	

# Inheritance Graph



# Flow Graph



## Summary

There are some functions that can be abused by the owner like massively blacklist addresses. 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 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>