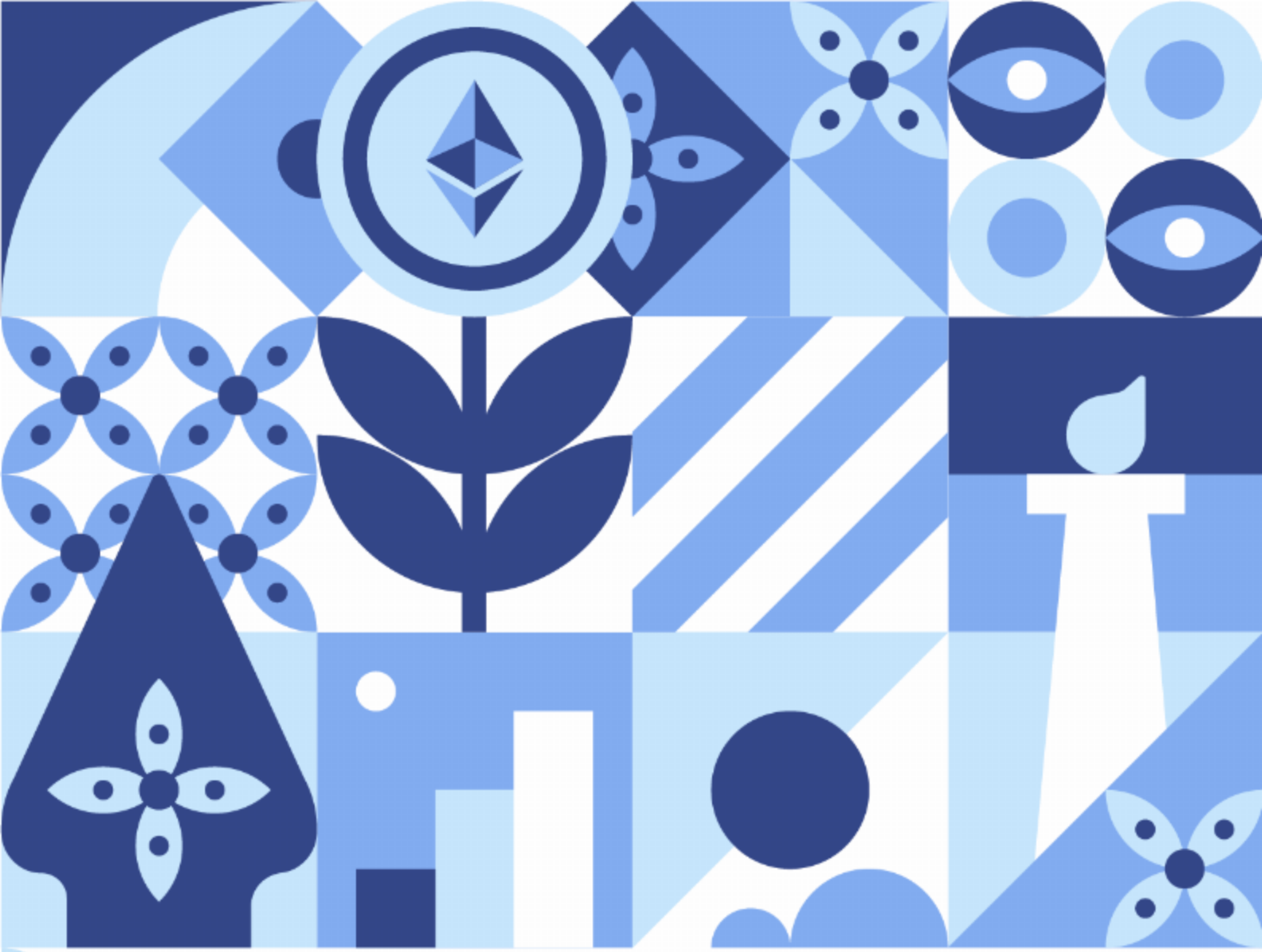

BabyCAW

Smart Contract Audit Report

SYSFIXED

10 Jun 2022

TABLE OF CONTENTS

Audited Details

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

Conclusion

Audit Results

Smart Contract Analysis

- Detected Vulnerabilities

Disclaimer

About Us

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
BabyCAW	BabyCAW	Ethereum

Addresses

Contract address	0x25cd00d22F2255235Ef6823cdA8ad003Dc68d859
Contract deployer address	0x219Ffad28740628653DAa9447f33210267738D52

Project Website

https://babycawcoin.com/

Codebase

https://etherscan.io/address/0x25cd00d22F2255235Ef6823cdA8ad003Dc68d859#code

SUMMARY

Babycaw is a decentralized DeFi project built on the ethereum blockchain (ERC20). It is inspired by the vision of Ryoshi (Founder of the $CAW & SHIBA) which is to achieve a truly decentralized and fully autonomous cryptocurrency project which is solely of teh people, by teh people and for teh people. This ideology is synonymous to what Abraham Lincoln himself envisioned for democracy which he defined as “a government of the people, by the people and for the people”.

Contract Summary

Documentation Quality

BabyCAW provides a very good documentation with standard of solidity base code.

- The technical description is provided clearly and structured and also dont have any high risk issue.

Code Quality

The Overall quality of the basecode is standard.

- Standard solidity basecode and rules are already followed by BabyCAW with the discovery of several low issues.

Test Coverage

Test coverage of the project is 100% (Through Codebase)

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 62, 74, 84, 85, 97, 109, 216, 466, 466, 467, 467, 498, 498, 510, 510, 624, 659, 661, 803, 804, 805, 806, 884, 901, 974, 974, 974, 974 and 661.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 31.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 625, 660, 661, 661, 750, 751, 885, 885, 886, 887, 1022 and 1023.

CONCLUSION

We have audited the BabyCAW project released on June 2022 to discover issues and identify potential security vulnerabilities in BabyCAW Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

The security audit report provides a satisfactory result with some low-risk issues.

The issues found in the BabyCAW smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
Unprotected Ether Withdrawal	SWC-105	Due to missing or insufficient access controls, malicious parties can withdraw from the contract.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Uninitialized Storage Pointer	SWC-109	Uninitialized local storage variables can point to unexpected storage locations in the contract.	PASS
Assert Violation	SWC-110 SWC-123	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Write to Arbitrary Storage Location	SWC-124	The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS
Insufficient Gas Griefing	SWC-126	Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

Typographical Error	SWC-129	A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable.	PASS
Override control character	SWC-130	Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	PASS
Unused variables	SWC-131 SWC-135	Unused variables are allowed in Solidity and they do not pose a direct security issue.	PASS
Unexpected Ether balance	SWC-132	Contracts can behave erroneously when they strictly assume a specific Ether balance.	PASS
Hash Collisions Variable	SWC-133	Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision.	PASS
Hardcoded gas amount	SWC-134	The transfer() and send() functions forward a fixed amount of 2300 gas.	PASS
Unencrypted Private Data	SWC-136	It is a common misconception that private type variables cannot be read.	PASS

SMART CONTRACT ANALYSIS

Started	Thursday Jun 09 2022 07:43:21 GMT+0000 (Coordinated Universal Time)
Finished	Friday Jun 10 2022 16:36:27 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	BabyCAW.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged

SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 62

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
61   function add(uint256 a, uint256 b) internal pure returns (uint256) {
62   uint256 c = a + b;
63   require(c >= a, "SafeMath: addition overflow");
64
65   return c;
66
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 74

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
73   require(b <= a, errorMessage);
74   uint256 c = a - b;
75
76   return c;
77   }
78
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 84

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
83
84   uint256 c = a * b;
85   require(c / a == b, "SafeMath: multiplication overflow");
86
87   return c;
88
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 85

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
84   uint256 c = a * b;
85   require(c / a == b, "SafeMath: multiplication overflow");
86
87   return c;
88   }
89
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 97

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
96   require(b > 0, errorMessage);
97   uint256 c = a / b;
98   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
99
100   return c;
101
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 109

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
108   require(b != 0, errorMessage);
109   return a % b;
110   }
111   }
112
113
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 216

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
215   _owner = address(0);
216   _lockTime = block.timestamp + time;
217   emit OwnershipTransferred(_owner, address(0));
218   }
219
220
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 466

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
465   uint256 private constant MAX = ~uint256(0);
466   uint256 private _tTotal = 333_333_333_333_333 * 10**18;
467   uint256 private _rTotal = (MAX - (MAX % _tTotal));
468   uint256 private _tFeeTotal;
469
470
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 466

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
465   uint256 private constant MAX = ~uint256(0);
466   uint256 private _tTotal = 333_333_333_333_333 * 10**18;
467   uint256 private _rTotal = (MAX - (MAX % _tTotal));
468   uint256 private _tFeeTotal;
469
470
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 467

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
466   uint256 private _tTotal = 333_333_333_333_333 * 10**18;
467   uint256 private _rTotal = (MAX - (MAX % _tTotal));
468   uint256 private _tFeeTotal;
469
470   string private _name = "BabyCAW";
471
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 467

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
466   uint256 private _tTotal = 333_333_333_333_333 * 10**18;
467   uint256 private _rTotal = (MAX - (MAX % _tTotal));
468   uint256 private _tFeeTotal;
469
470   string private _name = "BabyCAW";
471
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 498

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
497
498   uint256 public totalSwapableSaleFee = _saleLiquidityFee +_saleMarketingFee +
_saleBuybackFee;
499
500   bool public blacklistMode = true;
501   mapping (address => bool) public isBlacklisted;
502
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 498

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
497
498   uint256 public totalSwapableSaleFee = _saleLiquidityFee +_saleMarketingFee +
_saleBuybackFee;
499
500   bool public blacklistMode = true;
501   mapping (address => bool) public isBlacklisted;
502
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 510

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
509
510   uint256 private minimumTokensBeforeSwap = 100_000 * 10**18;
511
512   IUniswapV2Router02 public immutable uniswapV2Router;
513   address public immutable uniswapV2Pair;
514
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 510

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
509
510   uint256 private minimumTokensBeforeSwap = 100_000 * 10**18;
511
512   IUniswapV2Router02 public immutable uniswapV2Router;
513   address public immutable uniswapV2Pair;
514
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 624

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
623   function manage_blacklist(address[] calldata addresses, bool status) public
onlyOwner {
624   for (uint256 i; i < addresses.length; ++i) {
625   isBlacklisted[addresses[i]] = status;
626   }
627   }
628
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 659

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
658   require(_isExcluded[account], "Account is already excluded");
659   for (uint256 i = 0; i < _excluded.length; i++) {
660   if (_excluded[i] == account) {
661   _excluded[i] = _excluded[_excluded.length - 1];
662   _tOwned[account] = 0;
663
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 661

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
660   if (_excluded[i] == account) {
661   _excluded[i] = _excluded[_excluded.length - 1];
662   _tOwned[account] = 0;
663   _isExcluded[account] = false;
664   _excluded.pop();
665
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 803

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
802   if(totalSwapableFee==0) { return; }
803   liquidityTokensCollected += amount.mul(_liquidityFee).div(100);
804   devTokensCollected += amount.mul(_devFee).div(100);
805   marketingTokensCollected += amount.mul(_marketingFee).div(100);
806   buybackTokensCollected += amount.mul(_buybackFee).div(100);
807
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 804

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
803   liquidityTokensCollected += amount.mul(_liquidityFee).div(100);
804   devTokensCollected += amount.mul(_devFee).div(100);
805   marketingTokensCollected += amount.mul(_marketingFee).div(100);
806   buybackTokensCollected += amount.mul(_buybackFee).div(100);
807   }
808
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 805

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
804   devTokensCollected += amount.mul(_devFee).div(100);
805   marketingTokensCollected += amount.mul(_marketingFee).div(100);
806   buybackTokensCollected += amount.mul(_buybackFee).div(100);
807   }
808
809
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 806

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
805   marketingTokensCollected += amount.mul(_marketingFee).div(100);
806   buybackTokensCollected += amount.mul(_buybackFee).div(100);
807   }
808
809
810
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 884

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
883   uint256 tSupply = _tTotal;
884   for (uint256 i = 0; i < _excluded.length; i++) {
885   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
886   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
887   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
888
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 901

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
900   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
901   return _amount.mul(_taxFee).div(10**2);
902   }
903
904   function calculateLiquidityFee(uint256 _amount) private view returns (uint256) {
905
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 974

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
973   _saleBuybackFee = buybackFee;
974   totalSwapableSaleFee = _saleLiquidityFee +_saleMarketingFee + _saleBuybackFee +
_saleTaxFee + _saleDevFee;
975   require(totalSwapableSaleFee <= 10, "Must be less than 10% total");
976   }
977
978
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 974

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
973   _saleBuybackFee = buybackFee;
974   totalSwapableSaleFee = _saleLiquidityFee +_saleMarketingFee + _saleBuybackFee +
_saleTaxFee + _saleDevFee;
975   require(totalSwapableSaleFee <= 10, "Must be less than 10% total");
976   }
977
978
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 974

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
973   _saleBuybackFee = buybackFee;
974   totalSwapableSaleFee = _saleLiquidityFee +_saleMarketingFee + _saleBuybackFee +
_saleTaxFee + _saleDevFee;
975   require(totalSwapableSaleFee <= 10, "Must be less than 10% total");
976   }
977
978
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 974

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
973   _saleBuybackFee = buybackFee;
974   totalSwapableSaleFee = _saleLiquidityFee +_saleMarketingFee + _saleBuybackFee +
_saleTaxFee + _saleDevFee;
975   require(totalSwapableSaleFee <= 10, "Must be less than 10% total");
976   }
977
978
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 661

low SEVERITY

This plugin produces issues to support false positive discovery within mythril.

Source File

- BabyCAW.sol

Locations

```
660   if (_excluded[i] == account) {
661   _excluded[i] = _excluded[_excluded.length - 1];
662   _tOwned[account] = 0;
663   _isExcluded[account] = false;
664   _excluded.pop();
665
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 31

low SEVERITY

The current pragma Solidity directive is ""^0.8.14"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- BabyCAW.sol

Locations

```
30
31   pragma solidity ^0.8.14;
32
33   abstract contract Context {
34   function _msgSender() internal view virtual returns (address payable) {
35
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 625

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
624   for (uint256 i; i < addresses.length; ++i) {
625   isBlacklisted[addresses[i]] = status;
626   }
627   }
628
629
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 660

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
659   for (uint256 i = 0; i < _excluded.length; i++) {
660   if (_excluded[i] == account) {
661   _excluded[i] = _excluded[_excluded.length - 1];
662   _tOwned[account] = 0;
663   _isExcluded[account] = false;
664
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 661

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
660   if (_excluded[i] == account) {
661   _excluded[i] = _excluded[_excluded.length - 1];
662   _tOwned[account] = 0;
663   _isExcluded[account] = false;
664   _excluded.pop();
665
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 661

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
660   if (_excluded[i] == account) {
661   _excluded[i] = _excluded[_excluded.length - 1];
662   _tOwned[account] = 0;
663   _isExcluded[account] = false;
664   _excluded.pop();
665
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 750

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
749   address[] memory path = new address[](2);
750   path[0] = address(this);
751   path[1] = uniswapV2Router.WETH();
752   _approve(address(this), address(uniswapV2Router), tokenAmount);
753
754
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 751

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
750   path[0] = address(this);
751   path[1] = uniswapV2Router.WETH();
752   _approve(address(this), address(uniswapV2Router), tokenAmount);
753
754   // make the swap
755
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 885

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
884   for (uint256 i = 0; i < _excluded.length; i++) {
885   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
886   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
887   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
888   }
889
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 885

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
884   for (uint256 i = 0; i < _excluded.length; i++) {
885   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
886   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
887   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
888   }
889
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 886

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
885   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply) return
(_rTotal, _tTotal);
886   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
887   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
888   }
889   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
890
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 887

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
886   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
887   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
888   }
889   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
890   return (rSupply, tSupply);
891
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1022

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
1021   address[] memory path = new address[](2);
1022   path[0] = uniswapV2Router.WETH();
1023   path[1] = address(this);
1024   // make the swap
1025   uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
1026
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1023

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- BabyCAW.sol

Locations

```
1022   path[0] = uniswapV2Router.WETH();
1023   path[1] = address(this);
1024   // make the swap
1025   uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
1026   0, // accept any amount of Tokens
1027
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. 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 Sysfixed's prior written consent in each instance.

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 considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report 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 is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.