



Rabbit Year Smart Contract Audit Report

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
Rabbit Year	RabbitYear	Binance Smart Chain

Addresses

Contract address	0xA8Baa6Ce72c137A22441b033C5F9FA5A3c60ADDC
Contract deployer address	0x2857417abBcE3C5fce73d14b71dDaF26E7E7e71c

Project Website

<https://www.rabbityear2023.net/>

Codebase

<https://bscscan.com/address/0xA8Baa6Ce72c137A22441b033C5F9FA5A3c60ADDC#code>

SUMMARY

Rabbityear Token is a powerful MEME coin, and its goal is to become a decentralized community ecological project with a real purpose. The mission of RabbitYear Token is to bring the interesting new concept of cryptocurrency meme to mainstream investors, and raise RabbitYear Token to a new level of investment due to the buff of the large-scale Chinese New Year performances.

Contract Summary

Documentation Quality

Rabbit Year 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 Rabbit Year 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 202, 214, 227, 228, 239, 251, 263, 267, 279, 286, 295, 931, 1221, 1240, 1262, 1295, 1297, 1318, 1319, 1344, 1346, 1441, 1476, 1563, 1848, 1858, 1861, 1991, 1991, 1992, 2040, 2071, 2267, 2269, 2271, 2277, 2279, 2281, 2312, 2330, 2386 and 931.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 10.
- 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 900, 932, 937, 1854, 1974, 1975, 1976, 1978, 1979, 1980, 1981, 1983, 1984, 1985, 1986, 1998, 2006, 2041, 2072, 2337, 2338, 2355, 2356 and 2357.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 2165 and 2299.

CONCLUSION

We have audited the Rabbit Year project released on January 2023 to discover issues and identify potential security vulnerabilities in Rabbit Year 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 Rabbit Year 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, tx.origin as a part of authorization control, 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. We recommend avoiding "tx.origin" Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

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.	ISSUE FOUND
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 grieving 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	Sunday Jan 08 2023 07:14:13 GMT+0000 (Coordinated Universal Time)
Finished	Monday Jan 09 2023 03:46:27 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	BABYTOKEN.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	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

[illegible]

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 202

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
201     function add(uint256 a, uint256 b) internal pure returns (uint256) {  
202         uint256 c = a + b;  
203         require(c >= a, "SafeMath: addition overflow");  
204  
205         return c;  
206     }
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 214

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
213     require(b <= a, errorMessage);  
214     uint256 c = a - b;  
215  
216     return c;  
217 }  
218
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 227

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
226
227  uint256 c = a * b;
228  require(c / a == b, "SafeMath: multiplication overflow");
229
230  return c;
231
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 228

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
227     uint256 c = a * b;  
228     require(c / a == b, "SafeMath: multiplication overflow");  
229  
230     return c;  
231 }  
232
```


SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 239

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
238     require(b > 0, errorMessage);  
239     uint256 c = a / b;  
240     // assert(a == b * c + a % b); // There is no case in which this doesn't hold  
241  
242     return c;  
243
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 251

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
250     require(b != 0, errorMessage);
251     return a % b;
252 }
253 }
254
255
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 263

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
262 function mul(int256 a, int256 b) internal pure returns (int256) {  
263   int256 c = a * b;  
264  
265   // Detect overflow when multiplying MIN_INT256 with -1  
266   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));  
267 }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 267

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
266   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
267   require((b == 0) || (c / b == a));
268   return c;
269 }
270
271
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 279

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
278 // Solidity already throws when dividing by 0.  
279 return a / b;  
280 }  
281  
282 /**  
283
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 286

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
285     function sub(int256 a, int256 b) internal pure returns (int256) {  
286         int256 c = a - b;  
287         require((b >= 0 && c <= a) || (b < 0 && c > a));  
288         return c;  
289     }  
290
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 295

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
294 function add(int256 a, int256 b) internal pure returns (int256) {  
295     int256 c = a + b;  
296     require((b >= 0 && c >= a) || (b < 0 && c < a));  
297     return c;  
298 }  
299
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 931

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
930     uint256 index = map.indexOf[key];
931     uint256 lastIndex = map.keys.length - 1;
932     address lastKey = map.keys[lastIndex];
933
934     map.indexOf[lastKey] = index;
935
```


SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1221

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1220 unchecked {  
1221   _approve(sender, _msgSender(), currentAllowance - amount);  
1222 }  
1223  
1224 return true;  
1225
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1240

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1239     function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
1240     _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
1241     return true;
1242 }
1243
1244
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1262

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1261     unchecked {  
1262         _approve(_msgSender(), spender, currentAllowance - subtractedValue);  
1263     }  
1264  
1265     return true;  
1266
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1295

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1294     unchecked {  
1295         _balances[sender] = senderBalance - amount;  
1296     }  
1297     _balances[recipient] += amount;  
1298  
1299
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1297

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1296     }  
1297     _balances[recipient] += amount;  
1298  
1299     emit Transfer(sender, recipient, amount);  
1300  
1301
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1318

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1317
1318     _totalSupply += amount;
1319     _balances[account] += amount;
1320     emit Transfer(address(0), account, amount);
1321
1322
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1319

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1318     _totalSupply += amount;  
1319     _balances[account] += amount;  
1320     emit Transfer(address(0), account, amount);  
1321  
1322     _afterTokenTransfer(address(0), account, amount);  
1323
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1344

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1343     unchecked {  
1344         _balances[account] = accountBalance - amount;  
1345     }  
1346     _totalSupply -= amount;  
1347  
1348
```


SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1346

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1345     }  
1346     _totalSupply -= amount;  
1347  
1348     emit Transfer(account, address(0), amount);  
1349  
1350
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1441

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1440 // see https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
1441 uint256 internal constant magnitude = 2**128;
1442
1443 uint256 internal magnifiedDividendPerShare;
1444
1445
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1476

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1475     magnifiedDividendPerShare = magnifiedDividendPerShare.add(  
1476         (amount).mul(magnitude) / totalSupply()  
1477     );  
1478     emit DividendsDistributed(msg.sender, amount);  
1479  
1480
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1563

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1562     return
1563     magnifiedDividendPerShare
1564     .mul(balanceOf(_owner))
1565     .toInt256Safe()
1566     .add(magnifiedDividendCorrections[_owner])
1567
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1848

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1847 while (gasUsed < gas && iterations < numberOfTokenHolders) {  
1848   _lastProcessedIndex++;  
1849  
1850   if (_lastProcessedIndex >= tokenHoldersMap.keys.length) {  
1851     _lastProcessedIndex = 0;  
1852   }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1858

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1857     if (processAccount(payable(account), true)) {  
1858         claims++;  
1859     }  
1860 }  
1861 iterations++;  
1862
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1861

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1860     }  
1861     iterations++;  
1862  
1863     uint256 newGasLeft = gasleft();  
1864  
1865
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1991

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1990
1991     uint256 totalSupply = totalSupply_ * (10**18);
1992     swapTokensAtAmount = totalSupply.mul(2).div(10**6); // 0.002%
1993
1994     // use by default 300,000 gas to process auto-claiming dividends
1995
```


SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1991

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1990
1991     uint256 totalSupply = totalSupply_ * (10**18);
1992     swapTokensAtAmount = totalSupply.mul(2).div(10**6); // 0.002%
1993
1994     // use by default 300,000 gas to process auto-claiming dividends
1995
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1992

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1991  uint256 totalSupply = totalSupply_ * (10**18);
1992  swapTokensAtAmount = totalSupply.mul(2).div(10**6); // 0.002%
1993
1994  // use by default 300,000 gas to process auto-claiming dividends
1995  gasForProcessing = 300000;
1996
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 2040

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2039     function multipleBotlistAddress(address[] calldata accounts, bool excluded) public  
onlyOwner {  
2040     for (uint256 i = 0; i < accounts.length; i++) {  
2041         _isBlacklisted[accounts[i]] = excluded;  
2042     }  
2043 }  
2044
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 2071

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2070     function excludeMultipleAccountsFromFees(address[] calldata accounts, bool
excluded) public onlyOwner {
2071     for(uint256 i = 0; i < accounts.length; i++) {
2072     _isExcludedFromFees[accounts[i]] = excluded;
2073     }
2074
2075
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 2267

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2266   LFee = amount.mul(buyLiquidityFee).div(100);
2267   AmountLiquidityFee += LFee;
2268   RFee = amount.mul(buyTokenRewardsFee).div(100);
2269   AmountTokenRewardsFee += RFee;
2270   MFee = amount.mul(buyMarketingFee).div(100);
2271
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 2269

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2268 RFee = amount.mul(buyTokenRewardsFee).div(100);
2269 AmountTokenRewardsFee += RFee;
2270 MFee = amount.mul(buyMarketingFee).div(100);
2271 AmountMarketingFee += MFee;
2272 DFee = amount.mul(buyDeadFee).div(100);
2273
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 2271

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2270     MFee = amount.mul(buyMarketingFee).div(100);
2271     AmountMarketingFee += MFee;
2272     DFee = amount.mul(buyDeadFee).div(100);
2273     fees = LFee.add(RFee).add(MFee).add(DFee);
2274 }
2275
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 2277

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2276 LFee = amount.mul(sellLiquidityFee).div(100);  
2277 AmountLiquidityFee += LFee;  
2278 RFee = amount.mul(sellTokenRewardsFee).div(100);  
2279 AmountTokenRewardsFee += RFee;  
2280 MFee = amount.mul(sellMarketingFee).div(100);  
2281
```


SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 2279

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2278 RFee = amount.mul(sellTokenRewardsFee).div(100);
2279 AmountTokenRewardsFee += RFee;
2280 MFee = amount.mul(sellMarketingFee).div(100);
2281 AmountMarketingFee += MFee;
2282 DFee = amount.mul(sellDeadFee).div(100);
2283
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 2281

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2280     MFee = amount.mul(sellMarketingFee).div(100);
2281     AmountMarketingFee += MFee;
2282     DFee = amount.mul(sellDeadFee).div(100);
2283     fees = LFee.add(RFee).add(MFee).add(DFee);
2284 }
2285
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2312

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2311     IERC20(rewardToken).transfer(_marketingWalletAddress, newBalance);
2312     AmountMarketingFee = AmountMarketingFee - tokens;
2313 }
2314
2315 function swapAndLiquify(uint256 tokens) private {
2316
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2330

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2329     addLiquidity(otherHalf, newBalance);
2330     AmountLiquidityFee = AmountLiquidityFee - tokens;
2331     emit SwapAndLiquify(half, newBalance, otherHalf);
2332 }
2333
2334
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2386

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2385     swapTokensForCake(tokens);
2386     AmountTokenRewardsFee = AmountTokenRewardsFee - tokens;
2387     uint256 dividends = IERC20(rewardToken).balanceOf(address(this));
2388     bool success = IERC20(rewardToken).transfer(address(dividendTracker), dividends);
2389     if (success) {
2390
```

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

LINE 931

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
930     uint256 index = map.indexOf[key];
931     uint256 lastIndex = map.keys.length - 1;
932     address lastKey = map.keys[lastIndex];
933
934     map.indexOf[lastKey] = index;
935
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 10

low SEVERITY

The current pragma Solidity directive is `""^0.8.0""`. 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

- BABYTOKEN.sol

Locations

```
9  // SPDX-License-Identifier: MIT
10 pragma solidity ^0.8.0;
11
12 abstract contract Context {
13     function _msgSender() internal view virtual returns (address) {
14
```

SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 2165

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- BABYTOKEN.sol

Locations

```
2164     (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) =  
dividendTracker.process(gas);  
2165     emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas,  
tx.origin);  
2166     }  
2167  
2168     function claim() external {  
2169
```


SWC-115 | USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.

LINE 2299

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- BABYTOKEN.sol

Locations

```
2298     try dividendTracker.process(gas) returns (uint256 iterations, uint256 claims,
uint256 lastProcessedIndex) {
2299     emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas,
tx.origin);
2300     }
2301     catch {
2302
2303
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 900

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
899  {  
900  return map.keys[index];  
901  }  
902  
903  function size(Map storage map) public view returns (uint256) {  
904
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 932

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
931  uint256 lastIndex = map.keys.length - 1;
932  address lastKey = map.keys[lastIndex];
933
934  map.indexOf[lastKey] = index;
935  delete map.indexOf[key];
936
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 937

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
936
937     map.keys[index] = lastKey;
938     map.keys.pop();
939 }
940 }
941
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1854

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1853
1854     address account = tokenHoldersMap.keys[_lastProcessedIndex];
1855
1856     if (canAutoClaim(lastClaimTimes[account])) {
1857         if (processAccount payable(account), true) {
1858
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1974

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1973     ) payable ERC20(name_, symbol_) {  
1974     rewardToken = addr[0];  
1975     _marketingWalletAddress = addr[2];  
1976     _ContractAddress =addr[4];  
1977  
1978
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1975

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1974   rewardToken = addr[0];  
1975   _marketingWalletAddress = addr[2];  
1976   _ContractAddress =addr[4];  
1977  
1978   buyTokenRewardsFee = buyFeeSetting_[0];  
1979
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1976

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1975     _marketingWalletAddress = addr[2];
1976     _ContractAddress =addr[4];
1977
1978     buyTokenRewardsFee = buyFeeSetting[0];
1979     buyLiquidityFee = buyFeeSetting[1];
1980
```


SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1978

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1977
1978   buyTokenRewardsFee = buyFeeSetting_[0];
1979   buyLiquidityFee = buyFeeSetting_[1];
1980   buyMarketingFee = buyFeeSetting_[2];
1981   buyDeadFee = buyFeeSetting_[3];
1982
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1979

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1978     buyTokenRewardsFee = buyFeeSetting_[0];
1979     buyLiquidityFee = buyFeeSetting_[1];
1980     buyMarketingFee = buyFeeSetting_[2];
1981     buyDeadFee = buyFeeSetting_[3];
1982
1983
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1980

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1979     buyLiquidityFee = buyFeeSetting_[1];
1980     buyMarketingFee = buyFeeSetting_[2];
1981     buyDeadFee = buyFeeSetting_[3];
1982
1983     sellTokenRewardsFee = sellFeeSetting_[0];
1984
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1981

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1980     buyMarketingFee = buyFeeSetting_[2];
1981     buyDeadFee = buyFeeSetting_[3];
1982
1983     sellTokenRewardsFee = sellFeeSetting_[0];
1984     sellLiquidityFee = sellFeeSetting_[1];
1985
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1983

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1982
1983     sellTokenRewardsFee = sellFeeSetting_[0];
1984     sellLiquidityFee = sellFeeSetting_[1];
1985     sellMarketingFee = sellFeeSetting_[2];
1986     sellDeadFee = sellFeeSetting_[3];
1987
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1984

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1983     sellTokenRewardsFee = sellFeeSetting[0];
1984     sellLiquidityFee = sellFeeSetting[1];
1985     sellMarketingFee = sellFeeSetting[2];
1986     sellDeadFee = sellFeeSetting[3];
1987
1988
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1985

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1984     sellLiquidityFee = sellFeeSetting_[1];
1985     sellMarketingFee = sellFeeSetting_[2];
1986     sellDeadFee = sellFeeSetting_[3];
1987
1988
1989     require(buyTokenRewardsFee.add(buyLiquidityFee).add(buyMarketingFee).add(buyDeadFee) <=
25, "Total buy fee is over 25%");
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1986

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1985     sellMarketingFee = sellFeeSetting_[2];
1986     sellDeadFee = sellFeeSetting_[3];
1987
1988
require(buyTokenRewardsFee.add(buyLiquidityFee).add(buyMarketingFee).add(buyDeadFee) <=
25, "Total buy fee is over 25%");
1989
require(sellTokenRewardsFee.add(sellLiquidityFee).add(sellMarketingFee).add(sellDeadFee)
<= 25, "Total sell fee is over 25%");
1990
```


SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1998

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
1997     dividendTracker = BABYTOKENDividendTracker(  
1998     payable(Clones.clone(addr[3]))  
1999     );  
2000  
2001     dividendTracker.initialize(  
2002
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2006

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2005
2006     IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(addr[1]);
2007     address _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
2008         .createPair(address(this), _uniswapV2Router.WETH());
2009
2010
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2041

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2040   for (uint256 i = 0; i < accounts.length; i++) {  
2041     _isBlacklisted[accounts[i]] = excluded;  
2042   }  
2043 }  
2044  
2045
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2072

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2071   for(uint256 i = 0; i < accounts.length; i++) {  
2072       _isExcludedFromFees[accounts[i]] = excluded;  
2073   }  
2074  
2075   emit ExcludeMultipleAccountsFromFees(accounts, excluded);  
2076
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2337

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2336     address[] memory path = new address[](2);
2337     path[0] = address(this);
2338     path[1] = uniswapV2Router.WETH();
2339
2340     _approve(address(this), address(uniswapV2Router), tokenAmount);
2341
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2338

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2337     path[0] = address(this);  
2338     path[1] = uniswapV2Router.WETH();  
2339  
2340     _approve(address(this), address(uniswapV2Router), tokenAmount);  
2341  
2342
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2355

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2354     address[] memory path = new address[](3);
2355     path[0] = address(this);
2356     path[1] = uniswapV2Router.WETH();
2357     path[2] = rewardToken;
2358     _approve(address(this), address(uniswapV2Router), tokenAmount);
2359
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2356

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2355     path[0] = address(this);
2356     path[1] = uniswapV2Router.WETH();
2357     path[2] = rewardToken;
2358     _approve(address(this), address(uniswapV2Router), tokenAmount);
2359     // make the swap
2360
```


SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2357

low SEVERITY

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

Source File

- BABYTOKEN.sol

Locations

```
2356 path[1] = uniswapV2Router.WETH();
2357 path[2] = rewardToken;
2358 _approve(address(this), address(uniswapV2Router), tokenAmount);
2359 // make the swap
2360 uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
2361
```

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.