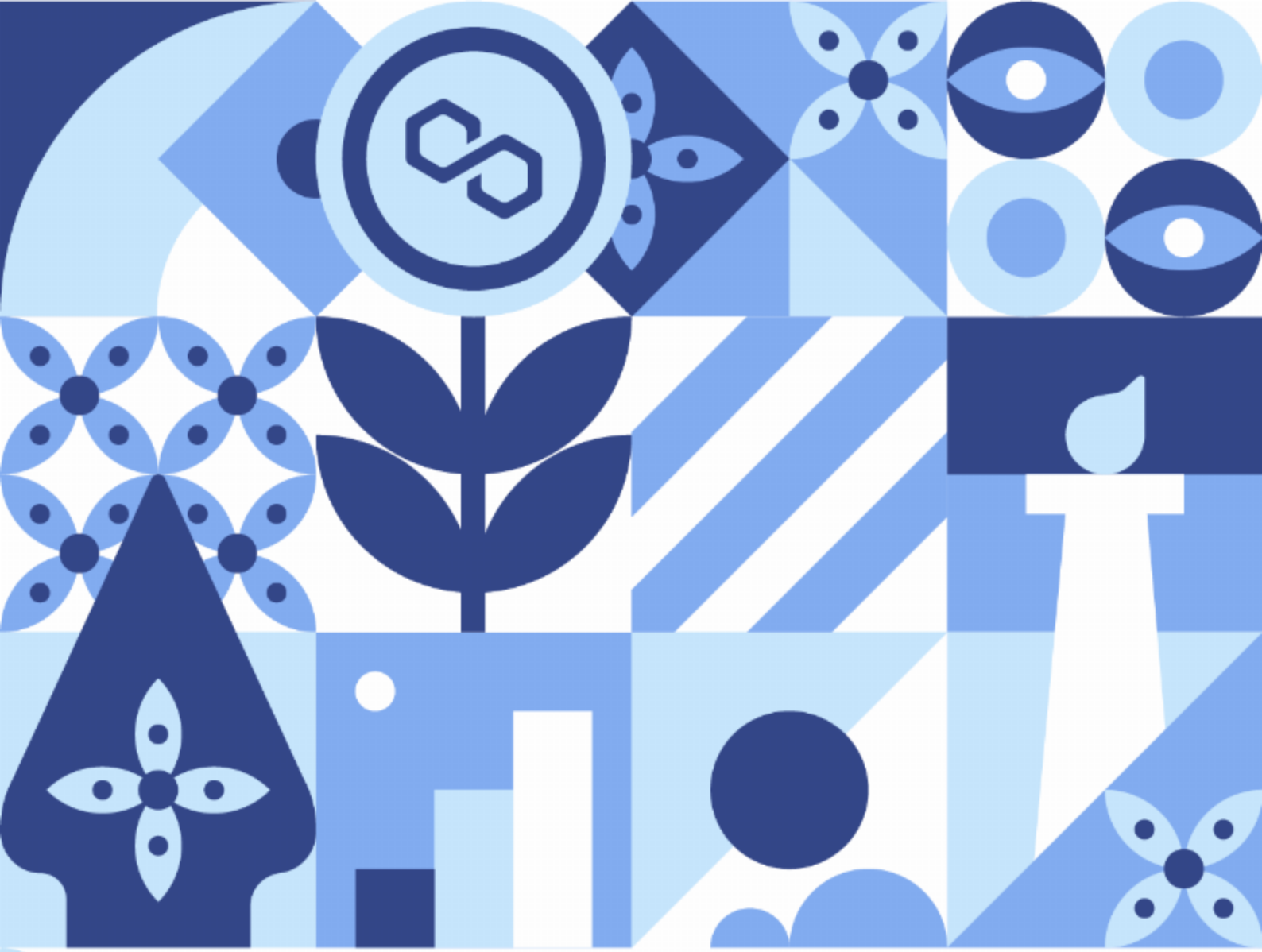

CBomber Token

Smart Contract Audit Report

SYSFIXED

17 Feb 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
CBomber Token	CBOMBER	Polygon Matic

Addresses

Contract address	0xcf74ae52ae2c848387e6cd0048e1ec5a93ee2c66
Contract deployer address	0x89d4c05848811155ce16a447c762421eaC93d927

Project Website

https://cryptobomber.io/

Codebase

https://polygonscan.com/address/0xcf74ae52ae2c848387e6cd0048e1ec5a93ee2c66#code

SUMMARY

Crypto Bomber is a Play-to-Earn game with CBOMBER as token reward. This token has an auto staking function that pays USDT for its holders. Crypto Bomber is a Play to Earn NFT RPG developed on the Polygon Mainnet. The game revolves around the acquisition of powerful Hunters and blastering Bombs to detonate them against the Monsters. Players may participate in combat using their assets to earn CBOMBER tokens. Assets are player's owned NFTs minted in the ERC-721 standard which may be traded at our Marketplace or .

Contract Summary

Documentation Quality

CBomber Token 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 CBomber Token with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-100 SWC-108 | Explicitly define visibility for all state variables on lines 194, 202, 203, 204, 206, 207, 208, 221, 223, 360, 361, 362, 363, 369, 374, 375, 377, 378, 379, 381, 382, 383, 384, 385, 386, 391, 392, 399, 400, 401, 402, 406, 407, 408, 409, 410, 412, 413, 417 and 420.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 11, 21, 30, 31, 41, 216, 219, 219, 303, 304, 309, 351, 352, 369, 369, 370, 371, 371, 416, 522, 595, 608, 608, 654, 669, 351 and 352.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 7.
- 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 266, 267, 297, 298, 351, 351, 352, 556, 557, 628 and 629.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 522, 595, 621, 649 and 665.

CONCLUSION

We have audited the CBomber Token project released on February 2022 to discover issues and identify potential security vulnerabilities in CBomber Token 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 satisfactory results with low-risk issues.

The issues found in the CBomber Token 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, a state variable visibility is not set, the potential use of "block.number" as a source of randomness, 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. The current pragma Solidity directive is ""^0.8.5"". Specifying a fixed compiler version is recommended 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. It is best practice to set the visibility of state variables explicitly. The default visibility for "DEAD" is internal. Other possible visibility settings are public and private. The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number, and timestamp are predictable and can be manipulated by a malicious miner. Also, keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness, and be aware that using these variables introduces a certain level of trust into miners.

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.	ISSUE FOUND
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.	ISSUE FOUND
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	Wednesday Feb 16 2022 03:05:53 GMT+0000 (Coordinated Universal Time)
Finished	Thursday Feb 17 2022 10:58:57 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	CBomber.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged

SWC-108	STATE VARIABLE VISIBILITY IS NOT 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-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 11

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
10   function add(uint256 a, uint256 b) internal pure returns (uint256) {
11   uint256 c = a + b;
12   require(c >= a, "SafeMath: addition overflow");
13
14   return c;
15
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 21

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
20   require(b <= a, errorMessage);
21   uint256 c = a - b;
22
23   return c;
24   }
25
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 30

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
29
30   uint256 c = a * b;
31   require(c / a == b, "SafeMath: multiplication overflow");
32
33   return c;
34
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 31

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
30   uint256 c = a * b;
31   require(c / a == b, "SafeMath: multiplication overflow");
32
33   return c;
34   }
35
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 41

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
40   require(b > 0, errorMessage);
41   uint256 c = a / b;
42   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
43
44   return c;
45
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 216

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
215   uint256 public dividendsPerShare;
216   uint256 public dividendsPerShareAccuracyFactor = 10 ** 36;
217
218   uint256 public minPeriod = 1 hours;
219   uint256 public minDistribution = 1 * (10 ** 8);
220
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 219

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
218   uint256 public minPeriod = 1 hours;
219   uint256 public minDistribution = 1 * (10 ** 8);
220
221   uint256 currentIndex;
222
223
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 219

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
218   uint256 public minPeriod = 1 hours;
219   uint256 public minDistribution = 1 * (10 ** 8);
220
221   uint256 currentIndex;
222
223
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 303

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
302   gasLeft = gasleft();
303   currentIndex++;
304   iterations++;
305   }
306   }
307
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 304

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
303   currentIndex++;
304   iterations++;
305   }
306   }
307
308
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 309

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
308   function shouldDistribute(address shareholder) internal view returns (bool) {
309   return shareholderClaims[shareholder] + minPeriod < block.timestamp
310   && getUnpaidEarnings(shareholder) > minDistribution;
311   }
312
313
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 351

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
350   function removeShareholder(address shareholder) internal {
351   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
352   shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
353   shareholders.pop();
354   }
355
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 352

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
351   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
352   shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
353   shareholders.pop();
354   }
355   }
356
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 369

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
368
369   uint256 _totalSupply = 1000000 * (10 ** _decimals);
370   uint256 public _maxTxAmount = _totalSupply / 40; // 2,5%
371   uint256 public maxWalletTokens =  1000000 * (10**9); // Anti-Whale
372   bool public lockTransfer = false;
373
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 369

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
368
369   uint256 _totalSupply = 1000000 * (10 ** _decimals);
370   uint256 public _maxTxAmount = _totalSupply / 40; // 2,5%
371   uint256 public maxWalletTokens =  1000000 * (10**9); // Anti-Whale
372   bool public lockTransfer = false;
373
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 370

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
369   uint256 _totalSupply = 1000000 * (10 ** _decimals);
370   uint256 public _maxTxAmount = _totalSupply / 40; // 2,5%
371   uint256 public maxWalletTokens =  1000000 * (10**9); // Anti-Whale
372   bool public lockTransfer = false;
373
374
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 371

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
370   uint256 public _maxTxAmount = _totalSupply / 40; // 2,5%
371   uint256 public maxWalletTokens =  1000000 * (10**9); // Anti-Whale
372   bool public lockTransfer = false;
373
374   mapping (address => uint256) _balances;
375
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 371

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
370   uint256 public _maxTxAmount = _totalSupply / 40; // 2,5%
371   uint256 public maxWalletTokens =  1000000 * (10**9); // Anti-Whale
372   bool public lockTransfer = false;
373
374   mapping (address => uint256) _balances;
375
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 416

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
415   bool public swapEnabled = true;
416   uint256 public swapThreshold = _totalSupply / 200; // 5%
417   bool inSwap;
418   modifier swapping() { inSwap = true; _; inSwap = false; }
419
420
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 522

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
521   function getTotalFee(bool selling) public view returns (uint256) {
522   if(launchedAt + 1 >= block.number){ return feeDenominator.sub(1); }
523   if(selling && buybackMultiplierTriggeredAt.add(buybackMultiplierLength) >
block.timestamp){ return getMultipliedFee(); }
524   return totalFee;
525   }
526
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 595

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
594   && autoBuybackEnabled
595   && autoBuybackBlockLast + autoBuybackBlockPeriod <= block.number
596   && address(this).balance >= autoBuybackAmount;
597   }
598
599
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 608

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
607   function removeMaxWallet() public onlyOwner {
608   maxWalletTokens = 2000000000 * 10**9;
609   }
610
611   function clearBuybackMultiplier() external authorized {
612
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 608

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
607   function removeMaxWallet() public onlyOwner {
608   maxWalletTokens = 2000000000 * 10**9;
609   }
610
611   function clearBuybackMultiplier() external authorized {
612
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 654

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
653   function setBuybackMultiplierSettings(uint256 numerator, uint256 denominator,
uint256 length) external authorized {
654   require(numerator / denominator <= 2 && numerator > denominator);
655   buybackMultiplierNumerator = numerator;
656   buybackMultiplierDenominator = denominator;
657   buybackMultiplierLength = length;
658
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 669

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
668   function setTxLimit(uint256 amount) external authorized {
669   require(amount >= _totalSupply / 1000);
670   _maxTxAmount = amount;
671   }
672
673
```

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

LINE 351

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
350   function removeShareholder(address shareholder) internal {
351   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
352   shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
353   shareholders.pop();
354   }
355
```

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

LINE 352

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
351   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
352   shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
353   shareholders.pop();
354   }
355   }
356
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.5"". 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

- CBomber.sol

Locations

```
6
7   pragma solidity ^0.8.5;
8
9   library SafeMath {
10   function add(uint256 a, uint256 b) internal pure returns (uint256) {
11
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 194

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_token" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
193
194   address _token;
195
196   struct Share {
197   uint256 amount;
198
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 202

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "USDT" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
201
202   IBEP20 USDT = IBEP20(0xc2132D05D31c914a87C6611C10748AEb04B58e8F);
203   address WBNB = 0x0d500B1d8E8eF31E21C99d1Db9A6444d3ADf1270;
204   IDEXRouter router;
205
206
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 203

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "WBNB" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
202   IBEP20 USDT = IBEP20(0xc2132D05D31c914a87C6611C10748AEb04B58e8F);
203   address WBNB = 0x0d500B1d8E8eF31E21C99d1Db9A6444d3ADf1270;
204   IDEXRouter router;
205
206   address[] shareholders;
207
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 204

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "router" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
203   address WBNB = 0x0d500B1d8E8eF31E21C99d1Db9A6444d3ADf1270;
204   IDEXRouter router;
205
206   address[] shareholders;
207   mapping (address => uint256) shareholderIndexes;
208
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 206

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "shareholders" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
205
206   address[] shareholders;
207   mapping (address => uint256) shareholderIndexes;
208   mapping (address => uint256) shareholderClaims;
209
210
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 207

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "shareholderIndexes" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
206   address[] shareholders;
207   mapping (address => uint256) shareholderIndexes;
208   mapping (address => uint256) shareholderClaims;
209
210   mapping (address => Share) public shares;
211
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 208

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "shareholderClaims" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
207   mapping (address => uint256) shareholderIndexes;
208   mapping (address => uint256) shareholderClaims;
209
210   mapping (address => Share) public shares;
211
212
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 221

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "currentIndex" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
220
221   uint256 currentIndex;
222
223   bool initialized;
224   modifier initialization() {
225
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 223

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "initialized" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
222
223   bool initialized;
224   modifier initialization() {
225   require(!initialized);
226   _;
227
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 360

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "USDT" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
359
360   address USDT = 0xc2132D05D31c914a87C6611C10748AEb04B58e8F;
361   address WBNB = 0x0d500B1d8E8eF31E21C99d1Db9A6444d3ADf1270;
362   address DEAD = 0x000000000000000000000000000000000000dEaD;
363   address ZERO = 0x0000000000000000000000000000000000000000;
364
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 361

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "WBNB" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
360   address USDT = 0xc2132D05D31c914a87C6611C10748AEb04B58e8F;
361   address WBNB = 0x0d500B1d8E8eF31E21C99d1Db9A6444d3ADf1270;
362   address DEAD = 0x000000000000000000000000000000000000dEaD;
363   address ZERO = 0x0000000000000000000000000000000000000000;
364
365
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 362

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "DEAD" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
361   address WBNB = 0x0d500B1d8E8eF31E21C99d1Db9A6444d3ADf1270;
362   address DEAD = 0x000000000000000000000000000000000000dEaD;
363   address ZERO = 0x0000000000000000000000000000000000000000;
364
365   string constant _name = "CBomber Token";
366
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 363

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "ZERO" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
362   address DEAD = 0x000000000000000000000000000000000000dEaD;
363   address ZERO = 0x0000000000000000000000000000000000000000;
364
365   string constant _name = "CBomber Token";
366   string constant _symbol = "CBOMBER";
367
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 369

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_totalSupply" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
368
369   uint256 _totalSupply = 1000000 * (10 ** _decimals);
370   uint256 public _maxTxAmount = _totalSupply / 40; // 2,5%
371   uint256 public maxWalletTokens =  1000000 * (10**9); // Anti-Whale
372   bool public lockTransfer = false;
373
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 374

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_balances" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
373
374   mapping (address => uint256) _balances;
375   mapping (address => mapping (address => uint256)) _allowances;
376
377   mapping (address => bool) isFeeExempt;
378
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 375

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "_allowances" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
374   mapping (address => uint256) _balances;
375   mapping (address => mapping (address => uint256)) _allowances;
376
377   mapping (address => bool) isFeeExempt;
378   mapping (address => bool) isTxLimitExempt;
379
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 377

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "isFeeExempt" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
376
377   mapping (address => bool) isFeeExempt;
378   mapping (address => bool) isTxLimitExempt;
379   mapping (address => bool) isDividendExempt;
380
381
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 378

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "isTxLimitExempt" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
377   mapping (address => bool) isFeeExempt;
378   mapping (address => bool) isTxLimitExempt;
379   mapping (address => bool) isDividendExempt;
380
381   uint256 liquidityFee = 300;
382
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 379

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "isDividendExempt" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
378   mapping (address => bool) isTxLimitExempt;
379   mapping (address => bool) isDividendExempt;
380
381   uint256 liquidityFee = 300;
382   uint256 buybackFee = 0;
383
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 381

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "liquidityFee" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
380
381   uint256 liquidityFee = 300;
382   uint256 buybackFee = 0;
383   uint256 reflectionFee = 300;
384   uint256 marketingFee = 0;
385
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 382

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "buybackFee" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
381   uint256 liquidityFee = 300;
382   uint256 buybackFee = 0;
383   uint256 reflectionFee = 300;
384   uint256 marketingFee = 0;
385   uint256 totalFee = 600;
386
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 383

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "reflectionFee" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
382   uint256 buybackFee = 0;
383   uint256 reflectionFee = 300;
384   uint256 marketingFee = 0;
385   uint256 totalFee = 600;
386   uint256 feeDenominator = 10000;
387
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 384

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "marketingFee" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
383   uint256 reflectionFee = 300;
384   uint256 marketingFee = 0;
385   uint256 totalFee = 600;
386   uint256 feeDenominator = 10000;
387
388
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 385

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "totalFee" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
384   uint256 marketingFee = 0;
385   uint256 totalFee = 600;
386   uint256 feeDenominator = 10000;
387
388   address public autoLiquidityReceiver;
389
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 386

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "feeDenominator" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
385   uint256 totalFee = 600;
386   uint256 feeDenominator = 10000;
387
388   address public autoLiquidityReceiver;
389   address public marketingFeeReceiver;
390
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 391

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "targetLiquidity" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
390
391   uint256 targetLiquidity = 25;
392   uint256 targetLiquidityDenominator = 100;
393
394   IDEXRouter public router;
395
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 392

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "targetLiquidityDenominator" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
391   uint256 targetLiquidity = 25;
392   uint256 targetLiquidityDenominator = 100;
393
394   IDEXRouter public router;
395   address public pair;
396
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 399

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "buybackMultiplierNumerator" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
398
399   uint256 buybackMultiplierNumerator = 200;
400   uint256 buybackMultiplierDenominator = 100;
401   uint256 buybackMultiplierTriggeredAt;
402   uint256 buybackMultiplierLength = 30 minutes;
403
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 400

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "buybackMultiplierDenominator" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
399   uint256 buybackMultiplierNumerator = 200;
400   uint256 buybackMultiplierDenominator = 100;
401   uint256 buybackMultiplierTriggeredAt;
402   uint256 buybackMultiplierLength = 30 minutes;
403
404
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 401

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "buybackMultiplierTriggeredAt" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
400   uint256 buybackMultiplierDenominator = 100;
401   uint256 buybackMultiplierTriggeredAt;
402   uint256 buybackMultiplierLength = 30 minutes;
403
404   bool public autoBuybackEnabled = false;
405
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 402

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "buybackMultiplierLength" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
401   uint256 buybackMultiplierTriggeredAt;
402   uint256 buybackMultiplierLength = 30 minutes;
403
404   bool public autoBuybackEnabled = false;
405   bool public autoBuybackMultiplier = true;
406
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 406

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "autoBuybackCap" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
405   bool public autoBuybackMultiplier = true;
406   uint256 autoBuybackCap;
407   uint256 autoBuybackAccumulator;
408   uint256 autoBuybackAmount;
409   uint256 autoBuybackBlockPeriod;
410
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 407

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "autoBuybackAccumulator" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
406   uint256 autoBuybackCap;
407   uint256 autoBuybackAccumulator;
408   uint256 autoBuybackAmount;
409   uint256 autoBuybackBlockPeriod;
410   uint256 autoBuybackBlockLast;
411
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 408

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "autoBuybackAmount" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
407   uint256 autoBuybackAccumulator;
408   uint256 autoBuybackAmount;
409   uint256 autoBuybackBlockPeriod;
410   uint256 autoBuybackBlockLast;
411
412
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 409

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "autoBuybackBlockPeriod" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
408   uint256 autoBuybackAmount;
409   uint256 autoBuybackBlockPeriod;
410   uint256 autoBuybackBlockLast;
411
412   DividendDistributor distributor;
413
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 410

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "autoBuybackBlockLast" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
409   uint256 autoBuybackBlockPeriod;
410   uint256 autoBuybackBlockLast;
411
412   DividendDistributor distributor;
413   uint256 distributorGas = 500000;
414
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 412

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "distributor" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
411
412   DividendDistributor distributor;
413   uint256 distributorGas = 500000;
414
415   bool public swapEnabled = true;
416
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 413

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "distributorGas" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
412   DividendDistributor distributor;
413   uint256 distributorGas = 500000;
414
415   bool public swapEnabled = true;
416   uint256 public swapThreshold = _totalSupply / 200; // 5%
417
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 417

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwap" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
416   uint256 public swapThreshold = _totalSupply / 200; // 5%
417   bool inSwap;
418   modifier swapping() { inSwap = true; _; inSwap = false; }
419
420   mapping(address => bool) isBlacklisted;
421
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 420

low SEVERITY

It is best practice to set the visibility of state variables explicitly. The default visibility for "isBlacklisted" is internal. Other possible visibility settings are public and private.

Source File

- CBomber.sol

Locations

```
419
420   mapping(address => bool) isBlacklisted;
421
422   constructor () Auth(msg.sender) {
423   router = IDEXRouter(0xa5E0829CaCEd8fFDD4De3c43696c57F7D7A678ff);
424
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 266

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
265   address[] memory path = new address[](2);
266   path[0] = WBNB;
267   path[1] = address(USDT);
268
269   router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: msg.value}(
270
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 267

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
266   path[0] = WBNB;
267   path[1] = address(USDT);
268
269   router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: msg.value}(
270   0,
271
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 297

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
296
297   if(shouldDistribute(shareholders[currentIndex])){
298   distributeDividend(shareholders[currentIndex]);
299   }
300
301
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 298

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
297   if(shouldDistribute(shareholders[currentIndex])){
298   distributeDividend(shareholders[currentIndex]);
299   }
300
301   gasUsed = gasUsed.add(gasLeft.sub(gasleft()));
302
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 351

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
350   function removeShareholder(address shareholder) internal {
351   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
352   shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
353   shareholders.pop();
354   }
355
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 351

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
350   function removeShareholder(address shareholder) internal {
351   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
352   shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
353   shareholders.pop();
354   }
355
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 352

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
351   shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
352   shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
353   shareholders.pop();
354   }
355   }
356
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 556

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
555   address[] memory path = new address[](2);
556   path[0] = address(this);
557   path[1] = WBNB;
558
559   uint256 balanceBefore = address(this).balance;
560
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 557

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
556   path[0] = address(this);
557   path[1] = WBNB;
558
559   uint256 balanceBefore = address(this).balance;
560
561
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 628

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
627   address[] memory path = new address[](2);
628   path[0] = WBNB;
629   path[1] = address(this);
630
631   router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
632
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 629

low SEVERITY

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

Source File

- CBomber.sol

Locations

```
628   path[0] = WBNB;
629   path[1] = address(this);
630
631   router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
632   0,
633
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 522

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- CBomber.sol

Locations

```
521   function getTotalFee(bool selling) public view returns (uint256) {
522   if(launchedAt + 1 >= block.number){ return feeDenominator.sub(1); }
523   if(selling && buybackMultiplierTriggeredAt.add(buybackMultiplierLength) >
block.timestamp){ return getMultipliedFee(); }
524   return totalFee;
525   }
526
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 595

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- CBomber.sol

Locations

```
594   && autoBuybackEnabled
595   && autoBuybackBlockLast + autoBuybackBlockPeriod <= block.number
596   && address(this).balance >= autoBuybackAmount;
597   }
598
599
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 621

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- CBomber.sol

Locations

```
620   }
621   autoBuybackBlockLast = block.number;
622   autoBuybackAccumulator = autoBuybackAccumulator.add(autoBuybackAmount);
623   if(autoBuybackAccumulator > autoBuybackCap){ autoBuybackEnabled = false; }
624   }
625
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 649

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- CBomber.sol

Locations

```
648   autoBuybackBlockPeriod = _period;
649   autoBuybackBlockLast = block.number;
650   autoBuybackMultiplier = _autoBuybackMultiplier;
651   }
652
653
```

SWC-120 | POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.

LINE 665

low SEVERITY

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source File

- CBomber.sol

Locations

```
664   function launch() internal {
665   launchedAt = block.number;
666   }
667
668   function setTxLimit(uint256 amount) external authorized {
669
```

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.