

Poject Dreams Smart Contract Audit Report



07 Jan 2023



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	
Project Dreams	PRO	BSC	

Addresses

Contract address	0x83e575d69397541Cf89f80758eeE63bdA8345Bf6
Contract deployer address	0xaD7f4232371416FdFD8f7E5D1C43E2B90CdB50cA

Project Website

https://projectdreams.net/

Codebase

https://bscscan.com/address/0x83e575d69397541Cf89f80758eeE63bdA8345Bf6#code



SUMMARY

Project Dreams Token (\$PRO) is an auto liquidity and auto BUSD rewarding BEP20 token. A percent of all buys and sells will auto-generate liquidity and also send BUSD rewards to Project Dreams token holders.

Contract Summary

Documentation Quality

The amount of documentation in this project is GOOD.

• The technical description is provided.

Code Quality

The Overall quality of the code is GOOD

• The official Solidity style guide is followed.

Test Coverage

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

Audit Findings Summary

- SWC-101 | Arithmetic operation discovered on lines 23, 32, 42, 43, 51, 58, 63, 67, 71, 75, 79, 85, 92, 99, 266, 269, 353, 354, 359, 401, 402, 421, 469, 575, 581, 655, 701, 718 and 749.
- SWC-101 | Compiler-rewritable " 1" discovered on lines 401 and 402.
- SWC-103 | A floating pragma is set on line 12, The current pragma Solidity directive is ""^0.8.0"".
- SWC-108 | State variable visibility is not set on lines 244, 252, 253, 254, 256, 257, 258, 271, 273, 411, 413, 414, 415, 421, 424, 425, 427, 428, 429, 431, 432, 433, 434, 435, 436, 441, 442, 450, 451, 452, 453, 456, 457, 458, 459, 460, 461, 463, 466 and 470. It is best practice to set the visibility of state variables explicitly to public or private.
- SWC-110 | Out of bounds array access on lines 316, 317, 347, 348, 401, 402, 613, 614, 680 and 681.
- SWC-120 | Potential use of "block.number" as source of randomness on lines 575, 655, 673, 697 and 713.





CONCLUSION

This report has been prepared for Project Dreams to discover issues and vulnerabilities in the source code of the Project Dreams project as well as any contract dependencies that were not part of an officially recognized library.

The security assessment resulted in findings that ranged from critical to informational.

Most issues found were low severity and any critical issue such as High Vulnerability was not found. Except for all other issues that were of negligible importance and mostly referred to coding standards and inefficiencies such as an arithmetic operation, a floating pragma, state variable visibility, and the use of "block.number" as a source of randomness.



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
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	PASS
Assert Violation	SWC-110	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 Caller	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
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
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



SMART CONTRACT ANALYSIS

Started	Mon Jan 30 2023 02:35:23 GMT+0000 (Coordinated Universal Time)	
Finished	Mon Jan 30 2023 02:35:32 GMT+0000 (Coordinated Universal Time)	
Mode	Standard	
Main Source File	pro.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	COMPILER-REWRITABLE " <uint> - 1" DISCOVERED</uint>	low	acknowledged
SWC-101	COMPILER-REWRITABLE " <uint> - 1" DISCOVERED</uint>	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



SYSFIXED

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-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



LINE 71

Iow SEVERITY

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

Source File

- pro.sol

```
70 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
71 return a * b;
72 }
```



LINE 75

Iow SEVERITY

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

Source File

- pro.sol

```
74 function div(uint256 a, uint256 b) internal pure returns (uint256) {
75 return a / b;
76 }
77 |
```



LINE 85

Iow SEVERITY

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

Source File

- pro.sol

```
84 require(b <= a, errorMessage);
85 return a - b;
86 }
87 }
```



LINE 92

Iow SEVERITY

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

Source File

- pro.sol

```
91 require(b > 0, errorMessage);
92 return a / b;
93 }
94 }
```



LINE 99

Iow SEVERITY

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

Source File

- pro.sol

```
98 require(b > 0, errorMessage);
99 return a % b;
100 }
101 }
```



LINE 266

Iow SEVERITY

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

Source File

- pro.sol

```
265 uint256 public dividendsPerShare;
266 uint256 public dividendsPerShareAccuracyFactor = 10 ** 36;
267 uint256 public minPeriod = 1 hours;
268 |
```



LINE 269

Iow SEVERITY

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

Source File

- pro.sol

```
268 uint256 public minPeriod = 1 hours;
269 uint256 public minDistribution = 1 * (10 ** 18);
270 uint256 currentIndex;
271 |
```



LINE 269

Iow SEVERITY

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

Source File

- pro.sol

```
268 uint256 public minPeriod = 1 hours;
269 uint256 public minDistribution = 1 * (10 ** 18);
270 uint256 currentIndex;
271 |
```



LINE 353

Iow SEVERITY

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

Source File

- pro.sol

Locations

352 gasLeft = gasleft(); 353 currentIndex++; 354 iterations++; 355 } 356 |



LINE 354

Iow SEVERITY

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

Source File

- pro.sol

Locations

353 currentIndex++;
354 iterations++;
355 }
356 }



LINE 359

Iow SEVERITY

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

Source File

- pro.sol

Locations

358 function shouldDistribute(address shareholder) internal view returns (bool) {
359 return shareholderClaims[shareholder] + minPeriod < block.timestamp
360 && getUnpaidEarnings(shareholder) > minDistribution;
361 }



LINE 401

Iow SEVERITY

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

Source File

- pro.sol

Locations

400 function removeShareholder(address shareholder) internal {
401 shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length1];
402 shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
403 shareholders.pop();



LINE 402

Iow SEVERITY

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

Source File

- pro.sol

```
401 shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
402 shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
403 shareholders.pop();
404 }
```



LINE 421

Iow SEVERITY

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

Source File

- pro.sol

```
420 uint8 constant _decimals = 18;
421 uint256 _totalSupply = 1_000_000_000_000 * (10 ** _decimals);
422 uint256 public _maxTxAmount = _totalSupply.div(100); // 1%
423 |
```



LINE 421

Iow SEVERITY

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

Source File

- pro.sol

```
420 uint8 constant _decimals = 18;
421 uint256 _totalSupply = 1_000_000_000_000 * (10 ** _decimals);
422 uint256 public _maxTxAmount = _totalSupply.div(100); // 1%
423 |
```



LINE 469

Iow SEVERITY

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

Source File

- pro.sol

Locations

468 bool public swapEnabled = true; 469 uint256 public swapThreshold = _totalSupply / 2000; // 0.005% 470 bool inSwap; 471 modifier swapping() { inSwap = true; _; inSwap = false; }



LINE 575

Iow SEVERITY

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

Source File

- pro.sol

```
574 function getTotalFee(bool selling) public view returns (uint256) {
575 if(launchedAt + 1 >= block.number){ return feeDenominator.sub(1); }
576 if(selling){ return getMultipliedFee(); }
577 return totalFee;
```



LINE 581

Iow SEVERITY

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

Source File

- pro.sol

Locations

580 function getMultipliedFee() public view returns (uint256) {
581 if (launchedAtTimestamp + 1 days > block.timestamp) {
582 return totalFee.mul(18000).div(feeDenominator);
583 } else if (buybackMultiplierTriggeredAt.add(buybackMultiplierLength) >
block.timestamp) {



LINE 655

Iow SEVERITY

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

Source File

- pro.sol

Locations

654 && autoBuybackEnabled 655 && autoBuybackBlockLast + autoBuybackBlockPeriod <= block.number // After N blocks from last buyback 656 && address(this).balance >= autoBuybackAmount; 657 }



LINE 701

Iow SEVERITY

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

Source File

- pro.sol

```
700 function setBuybackMultiplierSettings(uint256 numerator, uint256 denominator,
uint256 length) external authorized {
701 require(numerator / denominator <= 2 && numerator > denominator);
702 buybackMultiplierNumerator = numerator;
703 buybackMultiplierDenominator = denominator;
```



LINE 718

Iow SEVERITY

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

Source File

- pro.sol

Locations

717 function setTxLimit(uint256 amount) external authorized {
718 require(amount >= _totalSupply / 1000);
719 _maxTxAmount = amount;
720 }



LINE 749

Iow SEVERITY

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

Source File

- pro.sol

```
748 require(totalFee <= 2500, "Total fees must be less than or equal to 25%");
749 require(totalFee < feeDenominator/4);
750 }
751 |
```



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

LINE 401

Iow SEVERITY

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

Source File

- pro.sol

Locations

400 function removeShareholder(address shareholder) internal {
401 shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length1];
402 shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
403 shareholders.pop();



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

LINE 402

Iow SEVERITY

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

Source File

- pro.sol

```
401 shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
402 shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
403 shareholders.pop();
404 }
```



SWC-103 | A FLOATING PRAGMA IS SET

LINE 11

Iow 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

- pro.sol

```
10 //SPDX-License-Identifier: MIT
11 pragma solidity ^0.8.0;
12 |
13 |
```


SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 244

Iow 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

- pro.sol

```
243 using SafeMath for uint256;
244 address _token;
245 struct Share {
246 |
```



LINE 252

Iow SEVERITY

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

Source File

- pro.sol

```
251 }
252 IBEP20 BUSD = IBEP20(0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56);
253 address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
254 IDEXRouter router;
```



LINE 253

Iow 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

- pro.sol

```
252 IBEP20 BUSD = IBEP20(0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56);
253 address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
254 IDEXRouter router;
255 |
```



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 254

Iow 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

- pro.sol

Locations

253 address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c; 254 IDEXRouter router; 255 address[] shareholders; 256 |



LINE 256

Iow 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

- pro.sol

Locations

255 IDEXRouter router; 256 address[] shareholders; 257 mapping (address => uint256) shareholderIndexes; 258 mapping (address => uint256) shareholderClaims;



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 257

Iow 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

- pro.sol

```
256 address[] shareholders;
257 mapping (address => uint256) shareholderIndexes;
258 mapping (address => uint256) shareholderClaims;
259 |
```



LINE 258

Iow 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

- pro.sol

```
257 mapping (address => uint256) shareholderIndexes;
258 mapping (address => uint256) shareholderClaims;
259 mapping (address => Share) public shares;
260 |
```



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 271

Iow 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

- pro.sol

```
270 uint256 public minDistribution = 1 * (10 ** 18);
271 uint256 currentIndex;
272 bool initialized;
273 |
```



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 273

Iow 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

- pro.sol

```
272 uint256 currentIndex;
273 bool initialized;
274 modifier initialization() {
275 require(!initialized);
```



LINE 411

Iow SEVERITY

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

Source File

- pro.sol

Locations



LINE 413

Iow 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

- pro.sol

Locations



LINE 414

Iow 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

- pro.sol



LINE 415

Iow 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

- pro.sol

```
414 uint8 constant _decimals = 18;
415 uint256 _totalSupply = 1_000_000_000_000 * (10 ** _decimals);
416 uint256 public _maxTxAmount = _totalSupply.div(100); // 1%
417 |
```



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 424

Iow 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

- pro.sol

```
423 uint256 public _maxTxAmount = _totalSupply.div(100); // 1%
424 mapping (address => uint256) _balances;
425 mapping (address => mapping (address => uint256)) _allowances;
426 |
```



LINE 425

Iow 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

- pro.sol

```
424 mapping (address => uint256) _balances;
425 mapping (address => mapping (address => uint256)) _allowances;
426 mapping (address => bool) isFeeExempt;
427 |
```



LINE 427

Iow 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

- pro.sol

Locations

426 mapping (address => mapping (address => uint256)) _allowances; 427 mapping (address => bool) isFeeExempt; 428 mapping (address => bool) isTxLimitExempt; 429 mapping (address => bool) isDividendExempt;



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 428

Iow 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

- pro.sol

```
427 mapping (address => bool) isFeeExempt;
428 mapping (address => bool) isTxLimitExempt;
429 mapping (address => bool) isDividendExempt;
430 |
```



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 429

Iow 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

- pro.sol

```
428 mapping (address => bool) isTxLimitExempt;
429 mapping (address => bool) isDividendExempt;
430 uint256 liquidityFee = 200;
431 |
```



LINE 431

Iow 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

- pro.sol

Locations

430 mapping (address => bool) isDividendExempt; 431 uint256 liquidityFee = 200; 432 uint256 buybackFee = 100; 433 uint256 reflectionFee = 100;



LINE 432

Iow 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

- pro.sol

Locations

431 uint256 liquidityFee = 200; 432 uint256 buybackFee = 100; 433 uint256 reflectionFee = 100; 434 uint256 marketingFee = 200;



LINE 433

Iow 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

- pro.sol

```
432 uint256 buybackFee = 100;
433 uint256 reflectionFee = 100;
434 uint256 marketingFee = 200;
435 uint256 totalFee = 600;
```



LINE 434

Iow 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

- pro.sol

```
433 uint256 reflectionFee = 100;
434 uint256 marketingFee = 200;
435 uint256 totalFee = 600;
436 uint256 feeDenominator = 10000;
```



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 435

Iow 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

- pro.sol

```
434 uint256 marketingFee = 200;
435 uint256 totalFee = 600;
436 uint256 feeDenominator = 10000;
437 |
```



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 436

Iow 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

- pro.sol

```
435 uint256 totalFee = 600;
436 uint256 feeDenominator = 10000;
437 address public autoLiquidityReceiver;
438 |
```



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 441

Iow 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

- pro.sol

```
440 address public marketingFeeReceiver;
441 uint256 targetLiquidity = 25;
442 uint256 targetLiquidityDenominator = 100;
443 |
```



LINE 450

Iow 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

- pro.sol

Locations

449 uint256 public launchedAtTimestamp; 450 uint256 buybackMultiplierNumerator = 200; 451 uint256 buybackMultiplierDenominator = 100; 452 uint256 buybackMultiplierTriggeredAt;



LINE 451

Iow 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

- pro.sol

Locations

450 uint256 buybackMultiplierNumerator = 200; 451 uint256 buybackMultiplierDenominator = 100; 452 uint256 buybackMultiplierTriggeredAt; 453 uint256 buybackMultiplierLength = 30 minutes;



LINE 452

Iow 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

- pro.sol

```
451 uint256 buybackMultiplierDenominator = 100;
452 uint256 buybackMultiplierTriggeredAt;
453 uint256 buybackMultiplierLength = 30 minutes;
454 |
```



LINE 453

Iow 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

- pro.sol

```
452 uint256 buybackMultiplierTriggeredAt;
453 uint256 buybackMultiplierLength = 30 minutes;
454 bool public autoBuybackEnabled = false;
455 |
```



LINE 456

Iow SEVERITY

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

Source File

- pro.sol

Locations

455 bool public autoBuybackEnabled = false; 456 mapping (address => bool) buyBacker; 457 uint256 autoBuybackCap; 458 uint256 autoBuybackAccumulator;



LINE 457

Iow 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

- pro.sol

Locations

456 mapping (address => bool) buyBacker; 457 uint256 autoBuybackCap; 458 uint256 autoBuybackAccumulator; 459 uint256 autoBuybackAmount;





LINE 458

Iow 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

- pro.sol

Locations

457 uint256 autoBuybackCap;
458 uint256 autoBuybackAccumulator;
459 uint256 autoBuybackAmount;
460 uint256 autoBuybackBlockPeriod;



LINE 459

Iow 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

- pro.sol

458	uint256	autoBuybackAccumulator;
459	uint256	autoBuybackAmount;
460	uint256	autoBuybackBlockPeriod;
461	uint256	autoBuybackBlockLast;



LINE 460

Iow 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

- pro.sol

Locations

459 uint256 autoBuybackAmount; 460 uint256 autoBuybackBlockPeriod; 461 uint256 autoBuybackBlockLast; 462 |



LINE 461

Iow 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

- pro.sol

Locations

460 uint256 autoBuybackBlockPeriod;

- 461 uint256 autoBuybackBlockLast;
- 462 DividendDistributor distributor;

463



SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 463

Iow 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

- pro.sol

Locations

462 uint256 autoBuybackBlockLast;

- 463 DividendDistributor distributor;
- 464 address public distributorAddress;

465


SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 466

Iow 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

- pro.sol

Locations

465 address public distributorAddress; 466 uint256 distributorGas = 500000; 467 bool public swapEnabled = true; 468 |



C

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET

LINE 470

Iow 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

- pro.sol

```
469 uint256 public swapThreshold = _totalSupply / 2000; // 0.005%
470 bool inSwap;
471 modifier swapping() { inSwap = true; _; inSwap = false; }
472 |
```



LINE 316

Iow SEVERITY

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

Source File

- pro.sol

```
315 address[] memory path = new address[](2);
316 path[0] = WBNB;
317 path[1] = address(BUSD);
318 |
```



LINE 317

Iow SEVERITY

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

Source File

- pro.sol

```
316 path[0] = WBNB;
317 path[1] = address(BUSD);
318 router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: msg.value}(
319 |
```



LINE 347

Iow SEVERITY

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

Source File

- pro.sol

Locations

346 }
347 if(shouldDistribute(shareholders[currentIndex])){
348 distributeDividend(shareholders[currentIndex]);
349 }



LINE 348

Iow SEVERITY

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

Source File

- pro.sol

Locations

347 if(shouldDistribute(shareholders[currentIndex])){
348 distributeDividend(shareholders[currentIndex]);
349 }
350 |



LINE 401

Iow SEVERITY

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

Source File

- pro.sol

Locations

400 function removeShareholder(address shareholder) internal {
401 shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length1];
402 shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
403 shareholders.pop();



LINE 402

Iow SEVERITY

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

Source File

- pro.sol

```
401 shareholders[shareholderIndexes[shareholder]] = shareholders[shareholders.length-
1];
402 shareholderIndexes[shareholders[shareholders.length-1]] =
shareholderIndexes[shareholder];
403 shareholders.pop();
404 }
```



LINE 613

Iow SEVERITY

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

Source File

- pro.sol

Locations

612 address[] memory path = new address[](2); 613 path[0] = address(this); 614 path[1] = WBNB; 615 uint256 balanceBefore = address(this).balance;



LINE 614

Iow SEVERITY

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

Source File

- pro.sol

```
613 path[0] = address(this);
614 path[1] = WBNB;
615 uint256 balanceBefore = address(this).balance;
616 |
```



LINE 680

Iow SEVERITY

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

Source File

- pro.sol

```
679 address[] memory path = new address[](2);
680 path[0] = WBNB;
681 path[1] = address(this);
682 |
```



LINE 681

Iow SEVERITY

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

Source File

- pro.sol

```
680 path[0] = WBNB;
681 path[1] = address(this);
682 router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
683 |
```



LINE 575

Iow 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

- pro.sol

Locations

574 function getTotalFee(bool selling) public view returns (uint256) {
575 if(launchedAt + 1 >= block.number){ return feeDenominator.sub(1); }
576 if(selling){ return getMultipliedFee(); }
577 return totalFee;



LINE 655

Iow 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

- pro.sol

```
654 && autoBuybackEnabled
655 && autoBuybackBlockLast + autoBuybackBlockPeriod <= block.number // After N blocks
from last buyback
656 && address(this).balance >= autoBuybackAmount;
657 }
```





LINE 673

Iow 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

- pro.sol

Locations

672 buyTokens(autoBuybackAmount, DEAD); 673 autoBuybackBlockLast = block.number; 674 autoBuybackAccumulator = autoBuybackAccumulator.add(autoBuybackAmount); 675 if(autoBuybackAccumulator > autoBuybackCap){ autoBuybackEnabled = false; }



LINE 697

Iow 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

- pro.sol

```
696 autoBuybackBlockPeriod = _period;
697 autoBuybackBlockLast = block.number;
698 }
699 |
```



LINE 713

Iow 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

- pro.sol

Locations

712 require(launchedAt == 0, "Already launched boi");
713 launchedAt = block.number;
714 launchedAtTimestamp = block.timestamp;
715 }



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.