Shibonsu Inu

Smart Contract Audit Report

SYSFIXED

04 Mar 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Shibonsu Inu	Shibonsu	Binance Smart Chain

Addresses

Contract address	0xe093807237362aa30684c3e709c4d113d0eb997b
Contract deployer address	0xA80ee082C2Ea194feC1B0c7E2D117807b04e9B02

Project Website

https://shibonsuinu.com/

Codebase

https://bscscan.com/address/0xe093807237362aa30684c3e709c4d113d0eb997b#code

SUMMARY

We are thrilled to announce the launch of our Instant Usage Rewards program for Shibonsu Inu! From now on, each time a user transacts with Shibonsu Inu, we will reward 5% of the value to holders' decentralized wallets. The more Shibonsu Inu is actively used, the more rewards all holders earn. Our goal with this program is to increase network health and usage and create a more engaged and connected community of users. With this incentive structure, Shibonsu Inu holders have even more reason to utilize and hold their tokens! So join us in rewarding usage and owning Shibonsu Inu today.

Contract Summary

Documentation Quality

Shibonsu Inu 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 Shibonsu Inu with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 137, 137, 137, 137, 138, 138, 140, 140, 237, 243, 253, 286, 301, 303, 325, 326, 331, 334, 336, 364, 364, 365, 365, 367, 367, 388, 394, 395, 397, 397, 405, 411, 414, 415, 417, 473, 477, 480, 481, 519, 531, 531 and 303.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- 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 302, 303, 303, 412, 412, 414, 415, 503, 504 and 520.

CONCLUSION

We have audited the Shibonsu Inu project released on March 2023 to discover issues and identify potential security vulnerabilities in Shibonsu Inu 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 Shibonsu Inu smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, and out-of-bounds array access which the index access expression can cause an exception in case an invalid array index value is used.

AUDIT RESULT

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

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

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

SMART CONTRACT ANALYSIS

Started	Friday Mar 03 2023 13:42:15 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Mar 04 2023 19:27:29 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Shibonsu.sol

Detected Issues

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

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

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

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 137

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
136
137   uint256 private _tTotal = 100 *10**15 * 10**_decimals;
138   uint256 private _rTotal = (MAX - (MAX % _tTotal));
139
140   uint256 public swapTokensAtAmount = 1e14 * 10**_decimals;
141
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 137

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```

uint256 private _tTotal = 100 *10**15 * 10**_decimals;
uint256 private _rTotal = (MAX - (MAX % _tTotal));

uint256 public swapTokensAtAmount = 1e14 * 10**_decimals;

```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 137

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
136
137   uint256 private _tTotal = 100 *10**15 * 10**_decimals;
138   uint256 private _rTotal = (MAX - (MAX % _tTotal));
139
140   uint256 public swapTokensAtAmount = 1e14 * 10**_decimals;
141
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 137

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
136
137   uint256 private _tTotal = 100 *10**15 * 10**_decimals;
138   uint256 private _rTotal = (MAX - (MAX % _tTotal));
139
140   uint256 public swapTokensAtAmount = 1e14 * 10**_decimals;
141
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 138

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
137   uint256 private _tTotal = 100 *10**15 * 10**_decimals;
138   uint256 private _rTotal = (MAX - (MAX % _tTotal));
139
140   uint256 public swapTokensAtAmount = 1e14 * 10**_decimals;
141
142
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 138

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
137   uint256 private _tTotal = 100 *10**15 * 10**_decimals;
138   uint256 private _rTotal = (MAX - (MAX % _tTotal));
139
140   uint256 public swapTokensAtAmount = 1e14 * 10**_decimals;
141
142
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 140

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
139
140   uint256 public swapTokensAtAmount = 1e14 * 10**_decimals;
141
142   address public deadWallet = 0x000000000000000000000000000000000000dEaD;
143   address public marketingWallet = 0xEe650087b95AB37a1d3492595D18dbDae0fE9020;
144
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 140

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
139
140   uint256 public swapTokensAtAmount = 1e14 * 10**_decimals;
141
142   address public deadWallet = 0x000000000000000000000000000000000000dEaD;
143   address public marketingWallet = 0xEe650087b95AB37a1d3492595D18dbDae0fE9020;
144
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 237

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
require(currentAllowance >= amount, "BEP20: transfer amount exceeds allowance");
_approve(sender, _msgSender(), currentAllowance - amount);

return true;
}

```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 243

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
function increaseAllowance(address spender, uint256 addedValue) public returns
(bool) {
_approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
return true;
}


```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 253

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
252   require(currentAllowance >= subtractedValue, "BEP20: decreased allowance below
zero");
253   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
254
255   return true;
256   }
257
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 286

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
285   uint256 currentRate = _getRate();
286   return rAmount / currentRate;
287   }
288
289   //@dev kept original RFI naming -> "reward" as in reflection
290
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 301

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
300   require(_isExcluded[account], "Account is not excluded");
301   for (uint256 i = 0; i < _excluded.length; i++) {
302   if (_excluded[i] == account) {
303   _excluded[i] = _excluded[_excluded.length - 1];
304   _tOwned[account] = 0;
305
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 303

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
302   if (_excluded[i] == account) {
303   _excluded[i] = _excluded[_excluded.length - 1];
304   _tOwned[account] = 0;
305   _isExcluded[account] = false;
306   _excluded.pop();
307
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 325

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
function _reflectRfi(uint256 rRfi, uint256 tRfi) private {
_rTotal -= rRfi;
totFeesPaid.rfi += tRfi;
}
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 326

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
325   _rTotal -= rRfi;
326   totFeesPaid.rfi += tRfi;
327   }
328
329
330
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 331

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
330   function _takeMarketing(uint256 rMarketing, uint256 tMarketing) private {
331   totFeesPaid.marketing += tMarketing;
332
333   if (_isExcluded[address(this)]) {
334   _tOwned[address(this)] += tMarketing;
335
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 334

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
333    if (_isExcluded[address(this)]) {
334    _tOwned[address(this)] += tMarketing;
335    }
336    _rOwned[address(this)] += rMarketing;
337    }
338
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 336

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
335   }
336   _rOwned[address(this)] += rMarketing;
337   }
338
339
340
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 364

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
363
364   s.tRfi = (tAmount * taxes.rfi) / 100;
365   s.tMarketing = (tAmount * taxes.marketing) / 100;
366   s.tTransferAmount =
367   tAmount -
368
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 364

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
363
364   s.tRfi = (tAmount * taxes.rfi) / 100;
365   s.tMarketing = (tAmount * taxes.marketing) / 100;
366   s.tTransferAmount =
367   tAmount -
368
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 365

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
364   s.tRfi = (tAmount * taxes.rfi) / 100;
365   s.tMarketing = (tAmount * taxes.marketing) / 100;
366   s.tTransferAmount =
367   tAmount -
368   s.tRfi -
369
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 365

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
364   s.tRfi = (tAmount * taxes.rfi) / 100;
365   s.tMarketing = (tAmount * taxes.marketing) / 100;
366   s.tTransferAmount =
367   tAmount -
368   s.tRfi -
369
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 367

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
366   s.tTransferAmount =
367   tAmount -
368   s.tRfi -
369   s.tMarketing;
370   return s;
371
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 367

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
366   s.tTransferAmount =
367   tAmount -
368   s.tRfi -
369   s.tMarketing;
370   return s;
371
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 388

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
387   {
388   rAmount = tAmount * currentRate;
389
390   if (!takeFee) {
391   return (rAmount, rAmount, 0, 0);
392
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 394

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
393
394   rRfi = s.tRfi * currentRate;
395   rMarketing = s.tMarketing * currentRate;
396   rTransferAmount =
397   rAmount -
398
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 395

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
394   rRfi = s.tRfi * currentRate;
395   rMarketing = s.tMarketing * currentRate;
396   rTransferAmount =
397   rAmount -
398   rRfi -
399
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 397

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
396   rTransferAmount =
397   rAmount -
398   rRfi -
399   rMarketing;
400   return (rAmount, rTransferAmount, rRfi, rMarketing);
401
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 397

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
396   rTransferAmount =
397   rAmount -
398   rRfi -
399   rMarketing;
400   return (rAmount, rTransferAmount, rRfi, rMarketing);
401
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 405

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
404   (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
405   return rSupply / tSupply;
406   }
407
408   function _getCurrentSupply() private view returns (uint256, uint256) {
409
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 411

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
410   uint256 tSupply = _tTotal;
411   for (uint256 i = 0; i < _excluded.length; i++) {
412   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply)
413   return (_rTotal, _tTotal);
414   rSupply = rSupply - _rOwned[_excluded[i]];
415
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 414

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
413   return (_rTotal, _tTotal);
414   rSupply = rSupply - _rOwned[_excluded[i]];
415   tSupply = tSupply - _tOwned[_excluded[i]];
416   }
417   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
418
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 415

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
414   rSupply = rSupply - _rOwned[_excluded[i]];
415   tSupply = tSupply - _tOwned[_excluded[i]];
416   }
417   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
418   return (rSupply, tSupply);
419
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 417

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
416   }
417   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
418   return (rSupply, tSupply);
419   }
420
421
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 473

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
472   //from excluded
473   _tOwned[sender] = _tOwned[sender] - tAmount;
474   }
475   if (_isExcluded[recipient]) {
476   //to excluded
477
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 477

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
476   //to excluded
477   _tOwned[recipient] = _tOwned[recipient] + s.tTransferAmount;
478   }
479
480   _rOwned[sender] = _rOwned[sender] - s.rAmount;
481
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 480

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
479
480   _rOwned[sender] = _rOwned[sender] - s.rAmount;
481   _rOwned[recipient] = _rOwned[recipient] + s.rTransferAmount;
482
483   if (s.rRfi > 0 || s.tRfi > 0) _reflectRfi(s.rRfi, s.tRfi);
484
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 481

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
480   _rOwned[sender] = _rOwned[sender] - s.rAmount;
481   _rOwned[recipient] = _rOwned[recipient] + s.rTransferAmount;
482
483   if (s.rRfi > 0 || s.tRfi > 0) _reflectRfi(s.rRfi, s.tRfi);
484   if (s.rMarketing > 0 || s.tMarketing > 0) _takeMarketing(s.rMarketing,
s.tMarketing);
485
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 519

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
518   function bulkExcludeFee(address[] memory accounts, bool state) external onlyOwner {
519   for (uint256 i = 0; i < accounts.length; i++) {
520   _isExcludedFromFee[accounts[i]] = state;
521   }
522   }
523
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 531

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
530   require(amount <= 1e15, "Cannot set swap threshold amount higher than 1% of
tokens");
531   swapTokensAtAmount = amount * 10**_decimals;
532   }
533
534   //Use this in case BNB are sent to the contract by mistake
535
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 531

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
530   require(amount <= 1e15, "Cannot set swap threshold amount higher than 1% of
tokens");
531   swapTokensAtAmount = amount * 10**_decimals;
532   }
533
534   //Use this in case BNB are sent to the contract by mistake
535
```

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

LINE 303

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
302   if (_excluded[i] == account) {
303   _excluded[i] = _excluded[_excluded.length - 1];
304   _tOwned[account] = 0;
305   _isExcluded[account] = false;
306   _excluded.pop();
307
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""^0.8.17"". 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

- Shibonsu.sol

Locations

```
5    // SPDX-License-Identifier: UNLICENSE
6    pragma solidity ^0.8.17;
7
8    interface IBEP20 {
9    function totalSupply() external view returns (uint256);
10
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 302

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
301   for (uint256 i = 0; i < _excluded.length; i++) {
302   if (_excluded[i] == account) {
303   _excluded[i] = _excluded[_excluded.length - 1];
304   _tOwned[account] = 0;
305   _isExcluded[account] = false;
306
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 303

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
302   if (_excluded[i] == account) {
303   _excluded[i] = _excluded[_excluded.length - 1];
304   _tOwned[account] = 0;
305   _isExcluded[account] = false;
306   _excluded.pop();
307
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 303

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
302   if (_excluded[i] == account) {
303   _excluded[i] = _excluded[_excluded.length - 1];
304   _tOwned[account] = 0;
305   _isExcluded[account] = false;
306   _excluded.pop();
307
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 412

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
411   for (uint256 i = 0; i < _excluded.length; i++) {
412   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply)
413   return (_rTotal, _tTotal);
414   rSupply = rSupply - _rOwned[_excluded[i]];
415   tSupply = tSupply - _tOwned[_excluded[i]];
416
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 412

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
411   for (uint256 i = 0; i < _excluded.length; i++) {
412   if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] > tSupply)
413   return (_rTotal, _tTotal);
414   rSupply = rSupply - _rOwned[_excluded[i]];
415   tSupply = tSupply - _tOwned[_excluded[i]];
416
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 414

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
413   return (_rTotal, _tTotal);
414   rSupply = rSupply - _rOwned[_excluded[i]];
415   tSupply = tSupply - _tOwned[_excluded[i]];
416   }
417   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
418
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 415

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
414   rSupply = rSupply - _rOwned[_excluded[i]];
415   tSupply = tSupply - _tOwned[_excluded[i]];
416   }
417   if (rSupply < _rTotal / _tTotal) return (_rTotal, _tTotal);
418   return (rSupply, tSupply);
419
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 503

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
502   address[] memory path = new address[](2);
503   path[0] = address(this);
504   path[1] = router.WETH();
505
506   _approve(address(this), address(router), tokenAmount);
507
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 504

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
503   path[0] = address(this);
504   path[1] = router.WETH();
505
506   _approve(address(this), address(router), tokenAmount);
507
508
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 520

low SEVERITY

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

Source File

- Shibonsu.sol

Locations

```
519   for (uint256 i = 0; i < accounts.length; i++) {
520   _isExcludedFromFee[accounts[i]] = state;
521   }
522   }
523
524
```

DISCLAIMER

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.