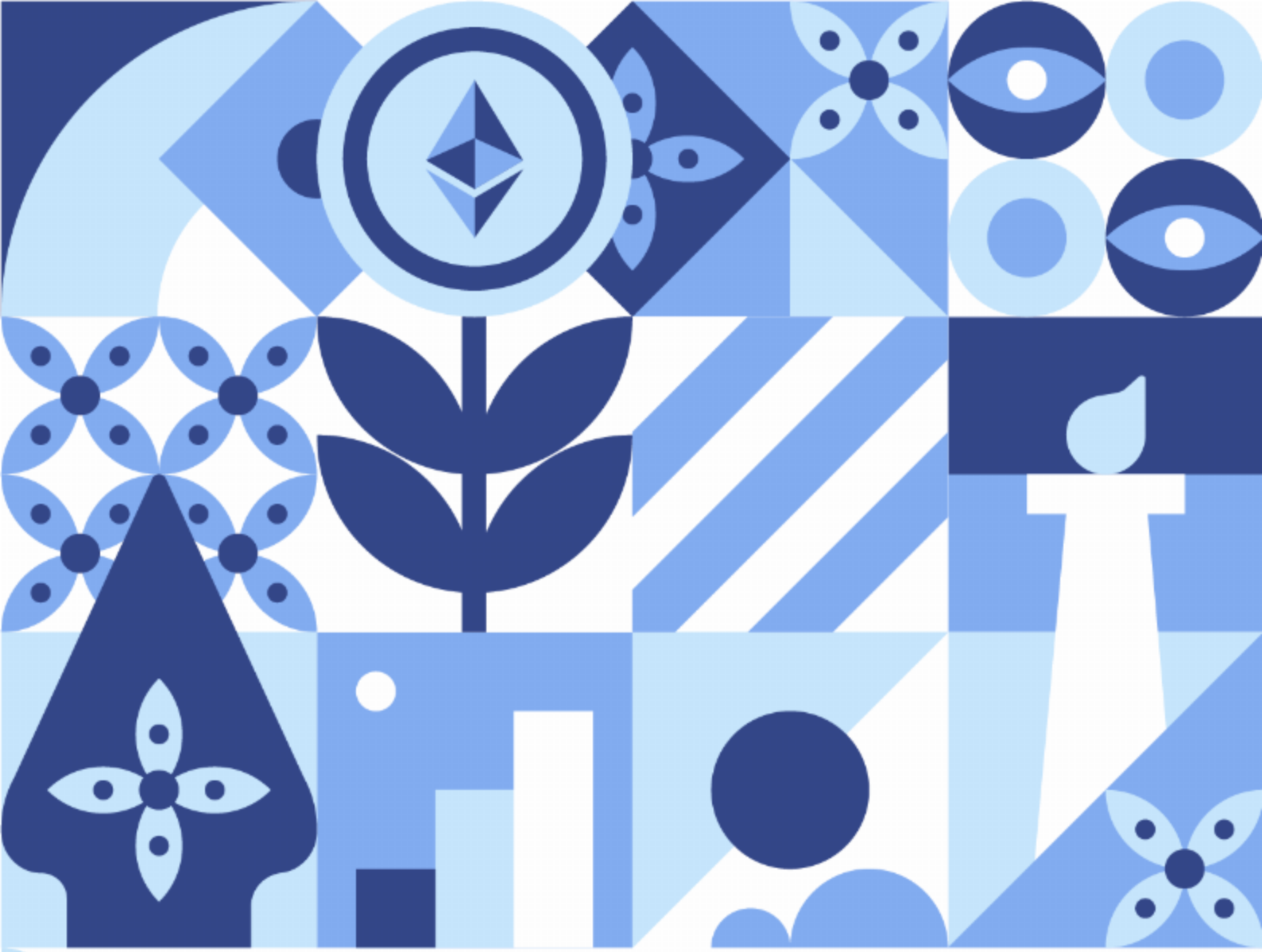

KIKI

Smart Contract Audit Report

SYSFIXED

30 Nov 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
KIKI	KIKI	Ethereum

Addresses

Contract address	0x369b77bbeeee50e6ea206dcf41ee670c47360055
Contract deployer address	0xbb616316B47c91240604A1E17Ac20fb677873302

Project Website

https://www.tabinekokiki.com/

Codebase

https://etherscan.io/address/0x369b77bbeeee50e6ea206dcf41ee670c47360055#code

SUMMARY

Far more than a crypto project, KIKI is a Movement. Combined with NFT, Novel, Charity, and Love, KIKI is one of its kind, a blockchain project with the love of art and charity, an art project with the spirit of blockchain.

Contract Summary

Documentation Quality

KIKI 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 KIKI 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 1093.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 114, 132, 151, 152, 169, 185, 200, 214, 228, 242, 258, 281, 308, 334, 689, 1070, 1070, 1071, 1071, 1096, 1096, 1097, 1097, 1307, 1309, 1342, 1450, 1481, 1489, 1493 and 1309.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 1.
- 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 1308, 1309, 1309, 1452, 1453, 1455, 1456, 1614 and 1615.

CONCLUSION

We have audited the KIKI project released on November 2021 to discover issues and identify potential security vulnerabilities in KIKI Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

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

The issues found in the KIKI 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, and out-of-bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	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.	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	Monday Nov 29 2021 21:55:08 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Nov 30 2021 08:53:40 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	KIKI.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	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-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 114

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
113   unchecked {
114   uint256 c = a + b;
115   if (c < a) return (false, 0);
116   return (true, c);
117   }
118
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 132

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
131   if (b > a) return (false, 0);
132   return (true, a - b);
133   }
134   }
135
136
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 151

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
150   if (a == 0) return (true, 0);
151   uint256 c = a * b;
152   if (c / a != b) return (false, 0);
153   return (true, c);
154   }
155
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 152

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
151   uint256 c = a * b;
152   if (c / a != b) return (false, 0);
153   return (true, c);
154   }
155   }
156
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 169

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
168   if (b == 0) return (false, 0);
169   return (true, a / b);
170   }
171   }
172
173
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 185

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
184   if (b == 0) return (false, 0);
185   return (true, a % b);
186   }
187   }
188
189
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 200

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
199   function add(uint256 a, uint256 b) internal pure returns (uint256) {
200   return a + b;
201   }
202
203   /**
204
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 214

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
213   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
214   return a - b;
215   }
216
217   /**
218
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 228

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
227   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
228   return a * b;
229   }
230
231   /**
232
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 242

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
241   function div(uint256 a, uint256 b) internal pure returns (uint256) {
242   return a / b;
243   }
244
245   /**
246
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 258

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
257   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
258   return a % b;
259   }
260
261   /**
262
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 281

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
280   require(b <= a, errorMessage);
281   return a - b;
282   }
283   }
284
285
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 308

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
307   require(b > 0, errorMessage);
308   return a / b;
309   }
310   }
311
312
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 334

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
333   require(b > 0, errorMessage);
334   return a % b;
335   }
336   }
337   }
338
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
688   _owner = address(0);
689   _lockTime = block.timestamp + time;
690   emit OwnershipTransferred(_owner, address(0));
691   }
692
693
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1070

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1069   uint256 private constant MAX = ~uint256(0);
1070   uint256 private _tTotal = 100_000_000 * 10**18;
1071   uint256 private _rTotal = (MAX - (MAX % _tTotal));
1072   uint256 private _tFeeTotal;
1073
1074
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1070

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1069   uint256 private constant MAX = ~uint256(0);
1070   uint256 private _tTotal = 100_000_000 * 10**18;
1071   uint256 private _rTotal = (MAX - (MAX % _tTotal));
1072   uint256 private _tFeeTotal;
1073
1074
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1071

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1070   uint256 private _tTotal = 100_000_000 * 10**18;
1071   uint256 private _rTotal = (MAX - (MAX % _tTotal));
1072   uint256 private _tFeeTotal;
1073
1074   string private _name = "KIKI";
1075
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 1071

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1070   uint256 private _tTotal = 100_000_000 * 10**18;
1071   uint256 private _rTotal = (MAX - (MAX % _tTotal));
1072   uint256 private _tFeeTotal;
1073
1074   string private _name = "KIKI";
1075
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1096

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1095
1096   uint256 public _maxTxAmount = 100_000_000 * 10**18;
1097   uint256 private numTokensSellToAddToLiquidity = 300_000 * 10**18;
1098
1099   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
1100
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1096

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1095
1096   uint256 public _maxTxAmount = 100_000_000 * 10**18;
1097   uint256 private numTokensSellToAddToLiquidity = 300_000 * 10**18;
1098
1099   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
1100
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1096   uint256 public _maxTxAmount = 100_000_000 * 10**18;
1097   uint256 private numTokensSellToAddToLiquidity = 300_000 * 10**18;
1098
1099   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
1100   event SwapAndLiquifyEnabledUpdated(bool enabled);
1101
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1097

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1096   uint256 public _maxTxAmount = 100_000_000 * 10**18;
1097   uint256 private numTokensSellToAddToLiquidity = 300_000 * 10**18;
1098
1099   event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
1100   event SwapAndLiquifyEnabledUpdated(bool enabled);
1101
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1307

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1306   require(_isExcluded[account], "Account is already included");
1307   for (uint256 i = 0; i < _excluded.length; i++) {
1308   if (_excluded[i] == account) {
1309   _excluded[i] = _excluded[_excluded.length - 1];
1310   _tOwned[account] = 0;
1311
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1309

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1308   if (_excluded[i] == account) {
1309   _excluded[i] = _excluded[_excluded.length - 1];
1310   _tOwned[account] = 0;
1311   _isExcluded[account] = false;
1312   _excluded.pop();
1313
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1342

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1341   function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner {
1342   _maxTxAmount = _tTotal.mul(maxTxPercent).div(10**2);
1343   }
1344
1345   function setMarketingWallet(address payable marketingWallet)
1346
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1450

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1449   uint256 tSupply = _tTotal;
1450   for (uint256 i = 0; i < _excluded.length; i++) {
1451   if (
1452   _rOwned[_excluded[i]] > rSupply ||
1453   _tOwned[_excluded[i]] > tSupply
1454
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1481

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1480   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1481   return _amount.mul(_taxFee).div(10**2);
1482   }
1483
1484   function calculateLiquidityFee(uint256 _amount)
1485
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1489

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1488   {
1489   return _amount.mul(_liquidityFee).div(10**2);
1490   }
1491
1492   function calculateBurnFee(uint256 _amount) private view returns (uint256) {
1493
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1493

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1492   function calculateBurnFee(uint256 _amount) private view returns (uint256) {
1493   return _amount.mul(_burnFee).div(10**2);
1494   }
1495
1496   function removeAllFee() private {
1497
```

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

LINE 1309

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1308   if (_excluded[i] == account) {
1309   _excluded[i] = _excluded[_excluded.length - 1];
1310   _tOwned[account] = 0;
1311   _isExcluded[account] = false;
1312   _excluded.pop();
1313
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1

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

- KIKI.sol

Locations

```
0
1   pragma solidity ^0.8.5;
2
3   // SPDX-License-Identifier: Unlicensed
4
5
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1093

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1092
1093   bool inSwapAndLiquify;
1094   bool public swapAndLiquifyEnabled = true;
1095
1096   uint256 public _maxTxAmount = 100_000_000 * 10**18;
1097
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1308

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1307   for (uint256 i = 0; i < _excluded.length; i++) {
1308   if (_excluded[i] == account) {
1309   _excluded[i] = _excluded[_excluded.length - 1];
1310   _tOwned[account] = 0;
1311   _isExcluded[account] = false;
1312
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1309

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1308   if (_excluded[i] == account) {
1309   _excluded[i] = _excluded[_excluded.length - 1];
1310   _tOwned[account] = 0;
1311   _isExcluded[account] = false;
1312   _excluded.pop();
1313
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1309

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1308   if (_excluded[i] == account) {
1309   _excluded[i] = _excluded[_excluded.length - 1];
1310   _tOwned[account] = 0;
1311   _isExcluded[account] = false;
1312   _excluded.pop();
1313
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1452

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1451   if (
1452   _rOwned[_excluded[i]] > rSupply ||
1453   _tOwned[_excluded[i]] > tSupply
1454   ) return (_rTotal, _tTotal);
1455   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1456
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1453

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1452   _rOwned[_excluded[i]] > rSupply ||
1453   _tOwned[_excluded[i]] > tSupply
1454   ) return (_rTotal, _tTotal);
1455   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1456   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1457
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1455

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1454   ) return (_rTotal, _tTotal);
1455   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1456   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1457   }
1458   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1459
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1456

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1455   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1456   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1457   }
1458   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1459   return (rSupply, tSupply);
1460
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1614

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1613   address[] memory path = new address[](2);
1614   path[0] = address(this);
1615   path[1] = uniswapV2Router.WETH();
1616
1617   _approve(address(this), address(uniswapV2Router), tokenAmount);
1618
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1615

low SEVERITY

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

Source File

- KIKI.sol

Locations

```
1614   path[0] = address(this);
1615   path[1] = uniswapV2Router.WETH();
1616
1617   _approve(address(this), address(uniswapV2Router), tokenAmount);
1618
1619
```

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.