Pi King

Smart Contract Audit Report

SYSFIXED

11 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Pi King	Pi King	BSC

Addresses

Contract address	0x29a34Db19070a9188a2E45e7cd2D9E92A328589C
Contract deployer address	0xe559EA0fBf86EEDCcB80ee572Cbeb0B89c4119D3

Project Website

https://piking.info/#/

Codebase

https://bscscan.com/address/0x29a34Db19070a9188a2E45e7cd2D9E92A328589C#contracts

SUMMARY

Pi King By the original Pi Network DeFi team to create. Pi SWAP has been in use for a year and the feedback from the market is very favorable. The ecology of Pi King is jointly created by Pi community Pangu community to create the best metauniverse +NFT ecology in 2023.

Contract Summary

Documentation Quality

Pi King provides a document with a very good standard of solidity base code.

- The technical description is provided clearly and structured and also don't have any high risk issue.

Code Quality

The Overall quality of the basecode is GOOD

- Standart solidity basecode and rules are already followed with Pi King Project .

Test Coverage

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

Audit Findings Summary

- SWC-101 | Arithmetic operation Issues discovered on lines 540, 556, 569, 570, 585, 601, 613, 617, 629, 636, 645, 1073, 1102, 1186, 1348, 1382, 1453, 1481, 1671, 1725, 1725, 1744, 1750, 1808, 1810, 1813, 1818, 1820, 1823, 1896, 2011, 2033, 2200, 2210, 2214, and 2285.
- SWC-103 | A floating pragma is set on lines 6. 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.
- SWC-115 | Use of "tx.origin" as a part of authorization control on lines 1595 and 1871. The index access expression can cause an exception in case an invalid array index value is used.
- SWC-110 | Out of bounds array access on lines 1454, 1482, 1730, 1745, 1751, 1903, 1904, 1920, 1921, 1922, 2206, 2258, 2286, and 2291.
- SWC-120 | OPotential use of "block.number" as source of randonmness on lines 1665.

CONCLUSION

We have audited the Pi King Coin which has released on January 2023 to discover issues and identify potential security vulnerabilities in Pi King Project. This process is used to find bugs, technical issues, and security loopholes that find some common issues in the code.

The security audit report produced satisfactory results with a low risk issue on the contract project.

The most common issue found in writing code on contracts that do not pose a big risk, writing on contracts is close to the standard of writing contracts in general. Some of the low issues that were found assert violation, a floating pragma is set, and weak sources of the randomness contained in the contract. We recommend to don't using any of those environment variables as sources of randomness and being aware that the use of these variables introduces a certain level of trust into miners.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	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
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.	ISSUE FOUND
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique Id	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
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	Tue Jan 10 2023 09:13:30 GMT+0000 (Coordinated Universal Time)
Finished	Wed Jan 21 2023 10:13:30 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	PiKing.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 540

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
539   function add(uint256 a, uint256 b) internal pure returns (uint256) {
540   uint256 c = a + b;
541   require(c >= a, "SafeMath: addition overflow");
542
543   return c;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 556

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
555   require(b <= a, errorMessage);
556   uint256 c = a - b;
557
558   return c;
559   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 569

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
568
569   uint256 c = a * b;
570   require(c / a == b, "SafeMath: multiplication overflow");
571
572   return c;
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 570

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
569   uint256 c = a * b;
570   require(c / a == b, "SafeMath: multiplication overflow");
571
572   return c;
573   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 585

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
584   require(b > 0, errorMessage);
585   uint256 c = a / b;
586   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
587
588   return c;
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 601

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
600   require(b != 0, errorMessage);
601   return a % b;
602   }
603   }
604
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 613

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
612   function mul(int256 a, int256 b) internal pure returns (int256) {
613   int256 c = a * b;
614
615   // Detect overflow when multiplying MIN_INT256 with -1
616   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 617

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
616   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
617   require((b == 0) || (c / b == a));
618   return c;
619   }
620
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 629

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
628   // Solidity already throws when dividing by 0.
629   return a / b;
630   }
631
632   /**
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 636

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
635   function sub(int256 a, int256 b) internal pure returns (int256) {
636   int256 c = a - b;
637   require((b >= 0 && c <= a) || (b < 0 && c > a));
638   return c;
639   }
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 645

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
644   function add(int256 a, int256 b) internal pure returns (int256) {
645   int256 c = a + b;
646   require((b >= 0 && c >= a) || (b < 0 && c < a));
647   return c;
648   }
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1073

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1072   //  see https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
1073   uint256 internal constant magnitude = 2**128;
1074
1075   uint256 internal magnifiedDividendPerShare;
1076
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1102

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1101   magnifiedDividendPerShare = magnifiedDividendPerShare.add(
1102   (amount).mul(magnitude) / totalSupply()
1103   );
1104   emit DividendsDistributed(msg.sender, amount);
1105
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1186

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1185   return
1186   magnifiedDividendPerShare
1187   .mul(balanceOf(_owner))
1188   .toInt256Safe()
1189   .add(magnifiedDividendCorrections[_owner])
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1348

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1347   constructor() ERC20("Pi King", "Pi King") {
1348   _totalSupply = 10000000000 * (10**18);
1349
1350   dividendTracker = new dDividendTracker();
1351
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1382

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1381
1382   swapTokensAtAmount = 10000000 * (10**18);
1383
1384   /*
1385   _mint is an internal function in ERC20.sol that is only called here,
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1453

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1452   ) public onlyOwner {
1453   for (uint256 i = 0; i < accounts.length; i++) {
1454   _isExcludedFromFees[accounts[i]] = excluded;
1455   }
1456
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1481

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1480   {
1481   for (uint256 i = 0; i < account.length; i++) {
1482   _isCpalaceed[account[i]] = value;
1483   }
1484   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1671

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1670   function setSwapTokensAtAmount(uint256 amount) public onlyOwner {
1671   swapTokensAtAmount = amount * (10**18);
1672   }
1673
1674   function setLiquidityHolders(address account, bool value) public onlyOwner {
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1725

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1724   {
1725   uint256 SCCC = tokens * addresses.length;
1726
1727   require(balanceOf(_msgSender()) >= SCCC, "Not enough tokens in wallet");
1728
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1729

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1728
1729   for (uint256 i = 0; i < addresses.length; i++) {
1730   _transfer(_msgSender(), addresses[i], tokens);
1731   }
1732   }
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1744

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1743
1744   for (uint256 i = 0; i < addresses.length; i++) {
1745   SCCC = SCCC.add(tokens[i]);
1746   }
1747
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1750

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1749
1750   for (uint256 i = 0; i < addresses.length; i++) {
1751   _transfer(_msgSender(), addresses[i], tokens[i]);
1752   }
1753   }
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1808

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1807   LFee = amount.mul(buyLiquidityFee).div(100);
1808   AmountLiquidityFee += LFee;
1809   CFee = amount.mul(buyRewardsFee).div(100);
1810   AmountRewardsFee += CFee;
1811   DFee = amount.mul(buyDeadFee).div(100);
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1810

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1809   CFee = amount.mul(buyRewardsFee).div(100);
1810   AmountRewardsFee += CFee;
1811   DFee = amount.mul(buyDeadFee).div(100);
1812   MFee = amount.mul(buyMarketFee).div(100);
1813   AmountMarketFee += MFee;
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1813

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1812   MFee = amount.mul(buyMarketFee).div(100);
1813   AmountMarketFee += MFee;
1814   fees = LFee.add(CFee).add(DFee).add(MFee);
1815   }
1816   if (automatedMarketMakerPairs[to]) {
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1818

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1817   LFee = amount.mul(sellLiquidityFee).div(100);
1818   AmountLiquidityFee += LFee;
1819   CFee = amount.mul(sellRewardsFee).div(100);
1820   AmountRewardsFee += CFee;
1821   DFee = amount.mul(sellDeadFee).div(100);
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1820

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1819   CFee = amount.mul(sellRewardsFee).div(100);
1820   AmountRewardsFee += CFee;
1821   DFee = amount.mul(sellDeadFee).div(100);
1822   MFee = amount.mul(sellMarketFee).div(100);
1823   AmountMarketFee += MFee;
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1823

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1822   MFee = amount.mul(sellMarketFee).div(100);
1823   AmountMarketFee += MFee;
1824   fees = LFee.add(CFee).add(DFee).add(MFee);
1825   uint256 balance = balanceOf(from);
1826   if (balance == amount) {
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1896

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1895   addLiquidity(otherHalf, newBalance);
1896   AmountLiquidityFee = AmountLiquidityFee - tokens;
1897   emit SwapAndLiquify(half, newBalance, otherHalf);
1898   }
1899
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 2011

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2010   claimWait = 21600;
2011   minimumTokenBalanceForDividends = 1000000 * (10**18); //must hold tokens for
dividen
2012   }
2013
2014   function _transfer(
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 2033

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2032   {
2033   minimumTokenBalanceForDividends = val * (10**18);
2034   }
2035
2036   function excludeFromDividends(address account) external onlyOwner {
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 2200

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2199   while (gasUsed < gas && iterations < numberOfTokenHolders) {
2200   _lastProcessedIndex++;
2201
2202   if (_lastProcessedIndex >= tokenHoldersMap.keys.length) {
2203   _lastProcessedIndex = 0;
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 2210

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2209   if (processAccount(payable(account), true)) {
2210   claims++;
2211   }
2212   }
2213
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 2214

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2213
2214   iterations++;
2215
2216   uint256 newGasLeft = gasleft();
2217
```

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

LINE 2285

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2284   uint256 index = tokenHoldersMap.indexOf[key];
2285   uint256 lastIndex = tokenHoldersMap.keys.length - 1;
2286   address lastKey = tokenHoldersMap.keys[lastIndex];
2287
2288   tokenHoldersMap.indexOf[lastKey] = index;
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""^0.8.4"". 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

- PiKing.Sol

Locations

```
5   // SPDX-License-Identifier: MIT
6   pragma solidity ^0.8.4;
7
8   interface IERC20 {
9   /**
```

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

LINE 1595

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1594   gas,
1595   tx.origin
1596   );
1597   }
1598
```

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

LINE 1871

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1870   gas,
1871   tx.origin
1872   );
1873   } catch {}
1874   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1454

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1453   for (uint256 i = 0; i < accounts.length; i++) {
1454   _isExcludedFromFees[accounts[i]] = excluded;
1455   }
1456
1457   emit ExcludeMultipleAccountsFromFees(accounts, excluded);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1482

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1481   for (uint256 i = 0; i < account.length; i++) {
1482   _isCpalaceed[account[i]] = value;
1483   }
1484   }
1485
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1730

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1729   for (uint256 i = 0; i < addresses.length; i++) {
1730   _transfer(_msgSender(), addresses[i], tokens);
1731   }
1732   }
1733
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1745

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1744   for (uint256 i = 0; i < addresses.length; i++) {
1745   SCCC = SCCC.add(tokens[i]);
1746   }
1747
1748   require(balanceOf(_msgSender()) >= SCCC, "Not enough tokens in wallet");
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1751

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1750   for (uint256 i = 0; i < addresses.length; i++) {
1751   _transfer(_msgSender(), addresses[i], tokens[i]);
1752   }
1753   }
1754
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1903

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1902   address[] memory path = new address[](2);
1903   path[0] = address(this);
1904   path[1] = uniswapV2Router.WETH();
1905
1906   _approve(address(this), address(uniswapV2Router), tokenAmount);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1904

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1903   path[0] = address(this);
1904   path[1] = uniswapV2Router.WETH();
1905
1906   _approve(address(this), address(uniswapV2Router), tokenAmount);
1907
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1920

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1919   address[] memory path = new address[](3);
1920   path[0] = address(this);
1921   path[1] = uniswapV2Router.WETH();
1922   path[2] = dToken;
1923   _approve(address(this), address(uniswapV2Router), tokenAmount);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1921

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1920   path[0] = address(this);
1921   path[1] = uniswapV2Router.WETH();
1922   path[2] = dToken;
1923   _approve(address(this), address(uniswapV2Router), tokenAmount);
1924   // make the swap
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1922

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1921   path[1] = uniswapV2Router.WETH();
1922   path[2] = dToken;
1923   _approve(address(this), address(uniswapV2Router), tokenAmount);
1924   // make the swap
1925   uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2206

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2205
2206   address account = tokenHoldersMap.keys[_lastProcessedIndex];
2207
2208   if (canAutoClaim(lastClaimTimes[account])) {
2209   if (processAccount(payable(account), true)) {
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2258

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2257   function MAPGetKeyAtIndex(uint256 index) public view returns (address) {
2258   return tokenHoldersMap.keys[index];
2259   }
2260
2261   function MAPSize() public view returns (uint256) {
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2286

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2285   uint256 lastIndex = tokenHoldersMap.keys.length - 1;
2286   address lastKey = tokenHoldersMap.keys[lastIndex];
2287
2288   tokenHoldersMap.indexOf[lastKey] = index;
2289   delete tokenHoldersMap.indexOf[key];
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2291

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
2290
2291   tokenHoldersMap.keys[index] = lastKey;
2292   tokenHoldersMap.keys.pop();
2293   }
2294   }
```

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

LINE 1665

low SEVERITY

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

Source File

- PiKing.Sol

Locations

```
1664   if (launchedAt == 0 || launchedTime == 0) {
1665   launchedAt = block.number;
1666   launchedTime = block.timestamp;
1667   }
1668   }
```

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.