BeastNian

Smart Contract Audit Report

SYSFIXED

14 Jan 2023

TABLE OF CONTENTS

Audited Details

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

Conclusion

Audit Results

Smart Contract Analysis

- Detected Vulnerabilities

Disclaimer

About Us

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
BeastNian	Nian	Binance Smart Chain

Addresses

Contract address	0x990696d6a75058b83D2D2e539810D73a7bBCBeA0
Contract deployer address	0x9B7726f677c956a5fE3BA147ceB7fDeD27e6349D

Project Website

https://www.beastnian.top/

Codebase

https://bscscan.com/address/0x990696d6a75058b83D2D2e539810D73a7bBCBeA0#code

SUMMARY

BeastNian is designed to create a complex ecosystem in which DeFi and Metaverse are integrated together. Beast Nian dedicated to GameFi, also includes additional utilities in the store. Tax: 1% foundation 1% Marketing 3% Reward Tax 1% NFT Reward Tax 1%

Contract Summary

Documentation Quality

BeastNian 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 BeastNian 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 943, 944, 1076 and 1077.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 271, 292, 325, 348, 349, 388, 428, 439, 442, 588, 592, 610, 611, 612, 654, 679, 688, 689, 728, 729, 735, 760, 764, 765, 776, 778, 780, 791, 800, 811, 815, 831, 833, 837, 840, 841, 842, 872, 896, 914, 971, 1005, 1011, 1013, 1014, 1043, 1094, 1100, 1111, 1119, 1147, 1152, 1154, 1155, 1094, 1100 and 1111.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- SWC-110 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 942, 1075, 818, 819, 820, 850, 851, 852, 898, 915, 953, 999, 1086, 1100, 1111 and 1144.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 679, 718, 765, 904, 971, 1017, 1119 and 1157.

CONCLUSION

We have audited the BeastNian project released on January 2023 to discover issues and identify potential security vulnerabilities in BeastNian 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 code on BeastNian smart contract 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, a public state variable with array type causing reachable exception by default, weak sources of randomness and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	ISSUE FOUND
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive 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 Callee	SWC-112	Delegate calls should only be allowed to trusted addresses.	PASS
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	ISSUE FOUND
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS

SMART CONTRACT ANALYSIS

Started	Friday Jan 13 2023 05:49:28 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Jan 14 2023 22:21:43 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Nian.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	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	low	acknowledged
SWC-110	PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.	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-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged
SWC-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 271

low SEVERITY

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

Source File

- Nian.sol

Locations

```
270   function add(uint256 a, uint256 b) internal pure returns (uint256) {
271   uint256 c = a + b;
272   require(c >= a, 'SafeMath: addition overflow');
273
274   return c;
275
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 292

low SEVERITY

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

Source File

- Nian.sol

Locations

```
291   else
292   return a-b;
293   }
294
295   /**
296
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 325

low SEVERITY

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

Source File

- Nian.sol

Locations

```
324   require(b <= a, errorMessage);
325   uint256 c = a - b;
326
327   return c;
328   }
329
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 348

low SEVERITY

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

Source File

- Nian.sol

Locations

```
347
348   uint256 c = a * b;
349   require(c / a == b, 'SafeMath: multiplication overflow');
350
351   return c;
352
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 349

low SEVERITY

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

Source File

- Nian.sol

Locations

```
348   uint256 c = a * b;
349   require(c / a == b, 'SafeMath: multiplication overflow');
350
351   return c;
352   }
353
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 388

low SEVERITY

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

Source File

- Nian.sol

Locations

```
387   require(b > 0, errorMessage);
388   uint256 c = a / b;
389   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
390
391   return c;
392
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 428

low SEVERITY

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

Source File

- Nian.sol

Locations

```
427   require(b != 0, errorMessage);
428   return a % b;
429   }
430
431   function min(uint256 x, uint256 y) internal pure returns (uint256 z) {
432
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 439

low SEVERITY

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

Source File

- Nian.sol

Locations

```
438   z = y;
439   uint256 x = y / 2 + 1;
440   while (x < z) {
441   z = x;
442   x = (y / x + x) / 2;
443
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 442

low SEVERITY

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

Source File

- Nian.sol

Locations

```
441   z = x;
442   x = (y / x + x) / 2;
443   }
444   } else if (y != 0) {
445   z = 1;
446
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 588

low SEVERITY

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

Source File

- Nian.sol

Locations

```
587
588   uint256 total = Supply * 10 ** Decimals;
589   _tTotal = total;
590   lpAddress = ReceiveAddress;
591   _balances[ReceiveAddress] = total;
592
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 592

low SEVERITY

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

Source File

- Nian.sol

Locations

```
591   _balances[ReceiveAddress] = total;
592   Startprice = 2 * 1e12;
593   emit Transfer(address(0), ReceiveAddress, total);
594
595   fundAddress = FundAddress;
596
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 610

low SEVERITY

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

Source File

- Nian.sol

Locations

```
609
610   holderRewardCondition = 100 ** IERC20(USDTAddress).decimals();
611   holderCondition = 200000 * 10 ** Decimals;
612   NFTRewardCondition = 20 ** IERC20(USDTAddress).decimals();
613   _tokenDistributor = new TokenDistributor(USDTAddress);
614
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 611

low SEVERITY

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

Source File

- Nian.sol

Locations

```
610   holderRewardCondition = 100 ** IERC20(USDTAddress).decimals();
611   holderCondition = 200000 * 10 ** Decimals;
612   NFTRewardCondition = 20 ** IERC20(USDTAddress).decimals();
613   _tokenDistributor = new TokenDistributor(USDTAddress);
614   _nftDistributor = new NFTRewardDistributor(USDTAddress);
615
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 612

low SEVERITY

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

Source File

- Nian.sol

Locations

```
611   holderCondition = 200000 * 10 ** Decimals;
612   NFTRewardCondition = 20 ** IERC20(USDTAddress).decimals();
613   _tokenDistributor = new TokenDistributor(USDTAddress);
614   _nftDistributor = new NFTRewardDistributor(USDTAddress);
615   }
616
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 654

low SEVERITY

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

Source File

- Nian.sol

Locations

```
653   if (_allowances[sender][msg.sender] != MAX) {
654   _allowances[sender][msg.sender] = _allowances[sender][msg.sender] - amount;
655   }
656   return true;
657   }
658
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 679

low SEVERITY

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

Source File

- Nian.sol

Locations

```
678   require(startTradeBlock>0);
679   if (block.number < startTradeBlock + kb) {
680   _funTransfer(from, to, amount);
681   return;
682   }
683
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 688

low SEVERITY

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

Source File

- Nian.sol

Locations

```
687   if (contractTokenBalance > 0) {
688   uint256 swapFee = _buyFundFee + _buyDividendFee + _buyNFTFee + _sellFundFee +
_sellDividendFee + _sellNFTFee;
689   uint256 numTokensSellToFund = amount * swapFee / 2000;
690   if (numTokensSellToFund > contractTokenBalance) {
691   numTokensSellToFund = contractTokenBalance;
692
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 689

low SEVERITY

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

Source File

- Nian.sol

Locations

```
688   uint256 swapFee = _buyFundFee + _buyDividendFee + _buyNFTFee + _sellFundFee +
_sellDividendFee + _sellNFTFee;
689   uint256 numTokensSellToFund = amount * swapFee / 2000;
690   if (numTokensSellToFund > contractTokenBalance) {
691   numTokensSellToFund = contractTokenBalance;
692   }
693
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 728

low SEVERITY

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

Source File

- Nian.sol

Locations

```
727   ) private {
728   _balances[sender] = _balances[sender] - tAmount;
729   uint256 feeAmount = tAmount * 60 / 100;
730   _takeTransfer(
731   sender,
732
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 729

low SEVERITY

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

Source File

- Nian.sol

Locations

```
728   _balances[sender] = _balances[sender] - tAmount;
729   uint256 feeAmount = tAmount * 60 / 100;
730   _takeTransfer(
731   sender,
732   address(this),
733
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 735

low SEVERITY

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

Source File

- Nian.sol

Locations

```
734   );
735   _takeTransfer(sender, recipient, tAmount - feeAmount);
736   }
737
738   function _tokenTransfer(
739
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 760

low SEVERITY

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

Source File

- Nian.sol

Locations

```
759   tAmount = tAmount.sub(cutcount);
760   swapFee = _sellFundFee + _sellDividendFee + _sellNFTFee;
761   }
762   else if(_swapPairList[sender])
763   {
764
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 764

low SEVERITY

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

Source File

- Nian.sol

Locations

```
763   {
764   swapFee = _buyFundFee + _buyDividendFee + _buyNFTFee;
765   if (block.number <= startTradeBlock + kb+2)swapFee+=2000;
766   }
767   else{
768
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 765

low SEVERITY

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

Source File

- Nian.sol

Locations

```
764   swapFee = _buyFundFee + _buyDividendFee + _buyNFTFee;
765   if (block.number <= startTradeBlock + kb+2)swapFee+=2000;
766   }
767   else{
768   uint256 cutcount = getCutCount(sender,tAmount,currentprice);
769
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 776

low SEVERITY

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

Source File

- Nian.sol

Locations

```
775   tAmount = tAmount.sub(cutcount);
776   swapFee = _sellFundFee + _sellDividendFee + _sellNFTFee;
777   }
778   swapAmount += tAmount * swapFee / 10000;
779   if (swapAmount > 0) {
780
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 778

low SEVERITY

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

Source File

- Nian.sol

Locations

```
777   }
778   swapAmount += tAmount * swapFee / 10000;
779   if (swapAmount > 0) {
780   feeAmount += swapAmount;
781   _takeTransfer(
782
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 780

low SEVERITY

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

Source File

- Nian.sol

Locations

```
779   if (swapAmount > 0) {
780   feeAmount += swapAmount;
781   _takeTransfer(
782   sender,
783   address(this),
784
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 791

low SEVERITY

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

Source File

- Nian.sol

Locations

```
790   uint256 totalvalue = _userHoldPrice[recipient].mul(oldbalance);
791   totalvalue += tAmount.mul(currentprice);
792   _userHoldPrice[recipient]= totalvalue.div(oldbalance.add(tAmount));
793   }
794   }
795
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 800

low SEVERITY

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

Source File

- Nian.sol

Locations

```
799   uint256 totalvalue = _userHoldPrice[recipient].mul(oldbalance);
800   totalvalue += tAmount.mul(Startprice);
801   _userHoldPrice[recipient]= totalvalue.div(oldbalance.add(tAmount));
802   }
803   else if(!_swapPairList[recipient])
804
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 811

low SEVERITY

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

Source File

- Nian.sol

Locations

```
810
811   _takeTransfer(sender, recipient, tAmount - feeAmount);
812   }
813
814   function swapTokenForFund(uint256 tokenAmount, uint256 swapFee) private lockTheSwap
{
815
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 815

low SEVERITY

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

Source File

- Nian.sol

Locations

```
814   function swapTokenForFund(uint256 tokenAmount, uint256 swapFee) private lockTheSwap
{
815   uint256 lpFee = _buyNFTFee + _sellNFTFee;
816   if(tokenAmount > balanceOf(address(this))) return;
817   address[] memory path = new address[](3);
818   path[0] = address(this);
819
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 831

low SEVERITY

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

Source File

- Nian.sol

Locations

```
830   uint256 USDTBalance = USDT.balanceOf(address(_tokenDistributor));
831   uint256 fundAmount = USDTBalance * (_buyFundFee + _sellFundFee + 100)/ swapFee;
832   uint256 fundAmount_A = fundAmount.mul(25).div(100);
833   uint256 fundAmount_B = fundAmount - fundAmount_A;
834
835
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 833

low SEVERITY

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

Source File

- Nian.sol

Locations

```
832   uint256 fundAmount_A = fundAmount.mul(25).div(100);
833   uint256 fundAmount_B = fundAmount - fundAmount_A;
834
835   USDT.transferFrom(address(_tokenDistributor), fundAddress, fundAmount_A);
836   USDT.transferFrom(address(_tokenDistributor), fundAddress2, fundAmount_B);
837
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 837

low SEVERITY

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

Source File

- Nian.sol

Locations

```
836   USDT.transferFrom(address(_tokenDistributor), fundAddress2, fundAmount_B);
837   USDT.transferFrom(address(_tokenDistributor), address(this), USDTBalance -
fundAmount);
838
839   if (lpFee > 0) {
840   uint256 lpUSDT = USDTBalance * lpFee / swapFee;
841
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 840

low SEVERITY

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

Source File

- Nian.sol

Locations

```
839   if (lpFee > 0) {
840   uint256 lpUSDT = USDTBalance * lpFee / swapFee;
841   if (lpUSDT > 0) {USDT.transfer(funder, lpUSDT - lpUSDT/2);
842   USDT.transfer(address(_nftDistributor), lpUSDT/2);
843   }
844
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 841

low SEVERITY

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

Source File

- Nian.sol

Locations

```
840   uint256 lpUSDT = USDTBalance * lpFee / swapFee;
841   if (lpUSDT > 0) {USDT.transfer(funder, lpUSDT - lpUSDT/2);
842   USDT.transfer(address(_nftDistributor), lpUSDT/2);
843   }
844   }
845
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 842

low SEVERITY

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

Source File

- Nian.sol

Locations

```
841   if (lpUSDT > 0) {USDT.transfer(funder, lpUSDT - lpUSDT/2);
842   USDT.transfer(address(_nftDistributor), lpUSDT/2);
843   }
844   }
845   }
846
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 872

low SEVERITY

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

Source File

- Nian.sol

Locations

```
871   _balances[sender] = _balances[sender].sub(tAmount);
872   _balances[to] = _balances[to] + tAmount;
873   emit Transfer(sender, to, tAmount);
874   }
875
876
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 896

low SEVERITY

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

Source File

- Nian.sol

Locations

```
895   function setWhiteUserPrice(address[] memory accountArray, uint256 newValue)public
onlyFunder {
896   for(uint256 i=0;i<accountArray.length;i++)
897   {
898   _userHoldPrice[accountArray[i]] = newValue;
899   }
900
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 914

low SEVERITY

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

Source File

- Nian.sol

Locations

```
913   require(addresses.length < 201);
914   for (uint256 i; i < addresses.length; ++i) {
915   _feeWhiteList[addresses[i]] = status;
916   }
917   }
918
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 971

low SEVERITY

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

Source File

- Nian.sol

Locations

```
970   function processReward(uint256 gas) private {
971   if (progressRewardBlock + minRewardTime > block.number) {
972   return;
973   }
974
975
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1005

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1004   if (tokenBalance > holderCondition && !excludeHolder[shareHolder]) {
1005   amount = balance * tokenBalance / holdTokenTotal;
1006   if (amount > 0) {
1007   USDT.transfer(shareHolder, amount);
1008   }
1009
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1011

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1010
1011   gasUsed = gasUsed + (gasLeft - gasleft());
1012   gasLeft = gasleft();
1013   currentIndex++;
1014   iterations++;
1015
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1013

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1012   gasLeft = gasleft();
1013   currentIndex++;
1014   iterations++;
1015   }
1016
1017
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1014

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1013   currentIndex++;
1014   iterations++;
1015   }
1016
1017   progressRewardBlock = block.number;
1018
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1043

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1042   {
1043   uint256 ylcount= amount.mul(currentprice -
_userHoldPrice[user]).div(currentprice);
1044   return ylcount.mul(20).div(100);
1045   }
1046   return 0;
1047
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1094

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1093   else if(!getRewardNFT(adr)){
1094   if(NFTholderIndex[adr] == NFTholders.length-1)
1095   {
1096   NFTholders.pop();
1097   NFTholderIndex[adr] = 0;
1098
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1100

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1099   }
1100   NFTholderIndex[NFTholders[NFTholders.length - 1]] = NFTholderIndex[adr];
1101   removeNFTholders(NFTholderIndex[adr]);
1102   NFTholderIndex[adr] = 0;
1103   }
1104
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1111

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1110
1111   NFTholders[index] = NFTholders[NFTholders.length - 1];
1112   NFTholders.pop();
1113   }
1114   uint256 private currentNFTIndex;
1115
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1119

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1118   function processNFTReward(uint256 gas) private {
1119   if (progressNFTBlock + minNFTRewardTime > block.number) {
1120   return;
1121   }
1122   IERC20 USDT = IERC20(_USDT);
1123
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1147

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1146   if (tokenBalance > 0 && !excludeNFTHolder[shareHolder]) {
1147   amount = balance / nfts;
1148   if (amount > 0 && USDT.balanceOf(address(_nftDistributor)) >= amount) {
1149   USDT.transferFrom(address(_nftDistributor), shareHolder, amount);
1150   }
1151
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1152

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1151   }
1152   gasUsed = gasUsed + (gasLeft - gasleft());
1153   gasLeft = gasleft();
1154   currentNFTIndex++;
1155   iterations++;
1156
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1154

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1153   gasLeft = gasleft();
1154   currentNFTIndex++;
1155   iterations++;
1156   }
1157   progressNFTBlock = block.number;
1158
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1155

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1154   currentNFTIndex++;
1155   iterations++;
1156   }
1157   progressNFTBlock = block.number;
1158   }
1159
```

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

LINE 1094

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1093   else if(!getRewardNFT(adr)){
1094   if(NFTholderIndex[adr] == NFTholders.length-1)
1095   {
1096   NFTholders.pop();
1097   NFTholderIndex[adr] = 0;
1098
```

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

LINE 1100

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1099   }
1100   NFTholderIndex[NFTholders[NFTholders.length - 1]] = NFTholderIndex[adr];
1101   removeNFTholders(NFTholderIndex[adr]);
1102   NFTholderIndex[adr] = 0;
1103   }
1104
```

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

LINE 1111

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1110
1111   NFTholders[index] = NFTholders[NFTholders.length - 1];
1112   NFTholders.pop();
1113   }
1114   uint256 private currentNFTIndex;
1115
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is ""^0.8.14"". 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

- Nian.sol

Locations

```
5   // SPDX-License-Identifier: MIT
6   pragma solidity ^0.8.14;
7
8   interface IERC165 {
9   /**
10
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 943

low SEVERITY

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

Source File

- Nian.sol

Locations

```
942   address[] public holders;
943   mapping(address => uint256) holderIndex;
944   mapping(address => bool) excludeHolder;
945
946   function addHolder(address adr) private {
947
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 944

low SEVERITY

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

Source File

- Nian.sol

Locations

```
943   mapping(address => uint256) holderIndex;
944   mapping(address => bool) excludeHolder;
945
946   function addHolder(address adr) private {
947   uint256 size;
948
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1076

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1075   address[] public NFTholders;
1076   mapping(address => uint256) NFTholderIndex;
1077   mapping(address => bool) excludeNFTHolder;
1078
1079   function addNFTHolder(address adr) private {
1080
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1077

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1076   mapping(address => uint256) NFTholderIndex;
1077   mapping(address => bool) excludeNFTHolder;
1078
1079   function addNFTHolder(address adr) private {
1080   uint256 size;
1081
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 942

low SEVERITY

The public state variable "holders" in "AbsToken" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source File

- Nian.sol

Locations

```
941   address[] public holders;
942   mapping(address => uint256) holderIndex;
943   mapping(address => bool) excludeHolder;
```

SWC-110 | PUBLIC STATE VARIABLE WITH ARRAY TYPE CAUSING REACHABLE EXCEPTION BY DEFAULT.

LINE 1075

low SEVERITY

The public state variable "NFTholders" in "AbsToken" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source File

- Nian.sol

Locations

```
1074   }
1075   address[] public NFTholders;
1076   mapping(address => uint256) NFTholderIndex;
1077   mapping(address => bool) excludeNFTHolder;
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 818

low SEVERITY

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

Source File

- Nian.sol

Locations

```
817   address[] memory path = new address[](3);
818   path[0] = address(this);
819   path[1] = _swapRouter.WETH();
820   path[2] = _USDT;
821   _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(
822
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 819

low SEVERITY

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

Source File

- Nian.sol

Locations

```
818   path[0] = address(this);
819   path[1] = _swapRouter.WETH();
820   path[2] = _USDT;
821   _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(
822   tokenAmount,
823
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 820

low SEVERITY

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

Source File

- Nian.sol

Locations

```
819   path[1] = _swapRouter.WETH();
820   path[2] = _USDT;
821   _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(
822   tokenAmount,
823   0,
824
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 850

low SEVERITY

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

Source File

- Nian.sol

Locations

```
849   if(tokenAmount > balanceOf(address(this))) return;
850   path[0] = address(this);
851   path[1] = _swapRouter.WETH();
852   path[2] = _USDT;
853   _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(
854
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 851

low SEVERITY

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

Source File

- Nian.sol

Locations

```
850   path[0] = address(this);
851   path[1] = _swapRouter.WETH();
852   path[2] = _USDT;
853   _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(
854   tokenAmount,
855
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 852

low SEVERITY

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

Source File

- Nian.sol

Locations

```
851   path[1] = _swapRouter.WETH();
852   path[2] = _USDT;
853   _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(
854   tokenAmount,
855   0,
856
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 898

low SEVERITY

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

Source File

- Nian.sol

Locations

```
897   {
898   _userHoldPrice[accountArray[i]] = newValue;
899   }
900   }
901
902
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 915

low SEVERITY

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

Source File

- Nian.sol

Locations

```
914   for (uint256 i; i < addresses.length; ++i) {
915   _feeWhiteList[addresses[i]] = status;
916   }
917   }
918
919
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 953

low SEVERITY

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

Source File

- Nian.sol

Locations

```
952   if (0 == holderIndex[adr]) {
953   if (0 == holders.length || holders[0] != adr) {
954   holderIndex[adr] = holders.length;
955   holders.push(adr);
956   }
957
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 999

low SEVERITY

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

Source File

- Nian.sol

Locations

```
998   }
999   shareHolder = holders[currentIndex];
1000   tokenBalance = holdToken.balanceOf(shareHolder);
1001   if(Lpwhite[shareHolder]) {
1002   if(tokenBalance > lpMaxNum[shareHolder]) lpMaxNum[shareHolder] = tokenBalance;
1003
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1086

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1085   if (0 == NFTholderIndex[adr]) {
1086   if (0 == NFTholders.length || NFTholders[0] != adr) {
1087   if(getRewardNFT(adr)){
1088   NFTholderIndex[adr] = NFTholders.length;
1089   NFTholders.push(adr);
1090
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1100

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1099   }
1100   NFTholderIndex[NFTholders[NFTholders.length - 1]] = NFTholderIndex[adr];
1101   removeNFTholders(NFTholderIndex[adr]);
1102   NFTholderIndex[adr] = 0;
1103   }
1104
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1111

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1110
1111   NFTholders[index] = NFTholders[NFTholders.length - 1];
1112   NFTholders.pop();
1113   }
1114   uint256 private currentNFTIndex;
1115
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1144

low SEVERITY

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

Source File

- Nian.sol

Locations

```
1143   }
1144   shareHolder = NFTholders[currentNFTIndex];
1145   tokenBalance = holdToken.balanceOf(shareHolder);
1146   if (tokenBalance > 0 && !excludeNFTHolder[shareHolder]) {
1147   amount = balance / nfts;
1148
```

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

LINE 679

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

- Nian.sol

Locations

```
678   require(startTradeBlock>0);
679   if (block.number < startTradeBlock + kb) {
680   _funTransfer(from, to, amount);
681   return;
682   }
683
```

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

LINE 718

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

- Nian.sol

Locations

```
717   processReward(500000);
718   if(progressRewardBlock < block.number)
719   processNFTReward(500000);
720   }
721   }
722
```

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

LINE 765

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

- Nian.sol

Locations

```
764   swapFee = _buyFundFee + _buyDividendFee + _buyNFTFee;
765   if (block.number <= startTradeBlock + kb+2)swapFee+=2000;
766   }
767   else{
768   uint256 cutcount = getCutCount(sender,tAmount,currentprice);
769
```

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

LINE 904

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

- Nian.sol

Locations

```
903   require(0 == startTradeBlock, "trading");
904   startTradeBlock = block.number;
905   kb = num;
906   }
907
908
```

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

LINE 971

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

- Nian.sol

Locations

```
970   function processReward(uint256 gas) private {
971   if (progressRewardBlock + minRewardTime > block.number) {
972   return;
973   }
974
975
```

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

LINE 1017

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

- Nian.sol

Locations

```
1016
1017   progressRewardBlock = block.number;
1018   }
1019
1020   function setHolderRewardCondition(uint256 amount) external onlyFunder {
1021
```

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

LINE 1119

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

- Nian.sol

Locations

```
1118   function processNFTReward(uint256 gas) private {
1119   if (progressNFTBlock + minNFTRewardTime > block.number) {
1120   return;
1121   }
1122   IERC20 USDT = IERC20(_USDT);
1123
```

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

LINE 1157

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

- Nian.sol

Locations

```
1156   }
1157   progressNFTBlock = block.number;
1158   }
1159
1160   function setNFTRewardCondition(uint256 amount) external onlyFunder {
1161
```

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.