Nitro Pyro

Smart Contract Audit Report

SYSFIXED

04 Nov 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Nitro Pyro	NIPYRO	Arbitrum

Addresses

Contract address	0x1549d3e06e900452316f8e322fb09026a0dce737
Contract deployer address	0x892f18aA2CFC95bfbA06EdE543873FE6d787c1a9

Project Website

https://www.nitro-pyro.com/

Codebase

https://arbiscan.io/address/0x1549d3e06e900452316f8e322fb09026a0dce737#code

SUMMARY

NITRO-PYRO is devoted to allowing our community to best gain value from Nitro pyro by innovating on-chain utility to enable our investors to profit.NITRO-PYRO empowers the community through transparency, security, and experience. Our team is of the mindset that any investment is a bet on the team, so we are here to provide value to investors through our expansive inventory of market Flames and Nitro.

Contract Summary

Documentation Quality

Nitro Pyro 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 Nitro Pyro with the discovery of several low issues.

Test Coverage

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

Audit Findings Summary

- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 479, 511, 534, 535, 570, 606, 672, 676, 688, 695, 704, 943, 943, 944, 950, 950, 951, 951, 952, 952, 956, 960, 984, 988, 1016, 1023, 1037, 1037, 1038, 1038, 1039, 1039, 1087, 1094, 1102, 1137, 1137, 1137, 1138, 1138, 1138, 1145, 1145, 1145, 1146, 1146, 1146, 1159, 1191 and 1192.
- 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 1169 and 1170.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1086 and 1087.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 1086 and 1087.

CONCLUSION

We have audited the Nitro Pyro project released in November 2022 to discover issues and identify potential security vulnerabilities in Nitro Pyro 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 Nitro Pyro 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, the use of "tx.origin" as a part of authorization control, out-of-bounds array access, and the potential use of "block.number" as a source of randomness. 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 the use of these variables introduces a certain level of trust in miners. 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.

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.	PASS
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.	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
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.	ISSUE FOUND
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	Thursday Nov 03 2022 02:00:02 GMT+0000 (Coordinated Universal Time)
Finished	Friday Nov 04 2022 10:02:45 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Token.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

SYSFIXED

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-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-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 479

low SEVERITY

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

Source File

- Token.sol

Locations

```
478   function add(uint256 a, uint256 b) internal pure returns (uint256) {
479   uint256 c = a + b;
480   require(c >= a, "SafeMath: addition overflow");
481
482   return c;
483
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 511

low SEVERITY

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

Source File

- Token.sol

Locations

```
510   require(b <= a, errorMessage);
511   uint256 c = a - b;
512
513   return c;
514   }
515
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 534

low SEVERITY

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

Source File

- Token.sol

Locations

```
533
534   uint256 c = a * b;
535   require(c / a == b, "SafeMath: multiplication overflow");
536
537   return c;
538
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 535

low SEVERITY

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

Source File

- Token.sol

Locations

```
534   uint256 c = a * b;
535   require(c / a == b, "SafeMath: multiplication overflow");
536
537   return c;
538   }
539
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 570

low SEVERITY

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

Source File

- Token.sol

Locations

```
569   require(b > 0, errorMessage);
570   uint256 c = a / b;
571   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
572
573   return c;
574
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 606

low SEVERITY

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

Source File

- Token.sol

Locations

```
605   require(b != 0, errorMessage);
606   return a % b;
607   }
608   }
609
610
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 672

low SEVERITY

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

Source File

- Token.sol

Locations

```
671   function mul(int256 a, int256 b) internal pure returns (int256) {
672   int256 c = a * b;
673
674   // Detect overflow when multiplying MIN_INT256 with -1
675   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
676
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 676

low SEVERITY

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

Source File

- Token.sol

Locations

```
675   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
676   require((b == 0) || (c / b == a));
677   return c;
678   }
679
680
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 688

low SEVERITY

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

Source File

- Token.sol

Locations

```
687   // Solidity already throws when dividing by 0.
688   return a / b;
689   }
690
691   /**
692
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 695

low SEVERITY

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

Source File

- Token.sol

Locations

```
694   function sub(int256 a, int256 b) internal pure returns (int256) {
695   int256 c = a - b;
696   require((b >= 0 && c <= a) || (b < 0 && c > a));
697   return c;
698   }
699
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 704

low SEVERITY

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

Source File

- Token.sol

Locations

```
703   function add(int256 a, int256 b) internal pure returns (int256) {
704   int256 c = a + b;
705   require((b >= 0 && c >= a) || (b < 0 && c < a));
706   return c;
707   }
708
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 943

low SEVERITY

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

Source File

- Token.sol

Locations

```
942
943   uint256 totalSupply = 1 * 1e6 * 1e6;
944   supply += totalSupply;
945
946   walletDigit = 2;
947
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 943

low SEVERITY

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

Source File

- Token.sol

Locations

```
942
943   uint256 totalSupply = 1 * 1e6 * 1e6;
944   supply += totalSupply;
945
946   walletDigit = 2;
947
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 944

low SEVERITY

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

Source File

- Token.sol

Locations

```
943   uint256 totalSupply = 1 * 1e6 * 1e6;
944   supply += totalSupply;
945
946   walletDigit = 2;
947   transDigit = 2;
948
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 950

low SEVERITY

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

Source File

- Token.sol

Locations

```
949
950   maxTransactionAmount = supply * transDigit / 200;
951   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
952   maxWallet = supply * walletDigit / 200;
953
954
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 950

low SEVERITY

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

Source File

- Token.sol

Locations

```
949
950   maxTransactionAmount = supply * transDigit / 200;
951   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
952   maxWallet = supply * walletDigit / 200;
953
954
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 951

low SEVERITY

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

Source File

- Token.sol

Locations

```
950   maxTransactionAmount = supply * transDigit / 200;
951   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
952   maxWallet = supply * walletDigit / 200;
953
954   buyBurnFee = _buyBurnFee;
955
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 951

low SEVERITY

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

Source File

- Token.sol

Locations

```
950   maxTransactionAmount = supply * transDigit / 200;
951   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
952   maxWallet = supply * walletDigit / 200;
953
954   buyBurnFee = _buyBurnFee;
955
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 952

low SEVERITY

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

Source File

- Token.sol

Locations

```
951   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
952   maxWallet = supply * walletDigit / 200;
953
954   buyBurnFee = _buyBurnFee;
955   buyDevFee = _buyDevFee;
956
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 952

low SEVERITY

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

Source File

- Token.sol

Locations

```
951   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
952   maxWallet = supply * walletDigit / 200;
953
954   buyBurnFee = _buyBurnFee;
955   buyDevFee = _buyDevFee;
956
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 956

low SEVERITY

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

Source File

- Token.sol

Locations

```
955   buyDevFee = _buyDevFee;
956   buyTotalFees = buyBurnFee + buyDevFee;
957
958   sellBurnFee = _sellBurnFee;
959   sellDevFee = _sellDevFee;
960
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 960

low SEVERITY

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

Source File

- Token.sol

Locations

```
959   sellDevFee = _sellDevFee;
960   sellTotalFees = sellBurnFee + sellDevFee;
961
962   devWallet = 0x892f18aA2CFC95bfbA06EdE543873FE6d787c1a9;
963
964
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 984

low SEVERITY

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

Source File

- Token.sol

Locations

```
983   buyDevFee = 5;
984   buyTotalFees = buyBurnFee + buyDevFee;
985
986   sellBurnFee = 4;
987   sellDevFee = 5;
988
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 988

low SEVERITY

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

Source File

- Token.sol

Locations

```
987   sellDevFee = 5;
988   sellTotalFees = sellBurnFee + sellDevFee;
989
990   delayDigit = 5;
991   }
992
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1016

low SEVERITY

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

Source File

- Token.sol

Locations

```
1015   buyDevFee = _devFee;
1016   buyTotalFees = buyBurnFee + buyDevFee;
1017   require(buyTotalFees <= 15, "Must keep fees at 20% or less");
1018   }
1019
1020
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1023

low SEVERITY

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

Source File

- Token.sol

Locations

```
1022   sellDevFee = _devFee;
1023   sellTotalFees = sellBurnFee + sellDevFee;
1024   require(sellTotalFees <= 15, "Must keep fees at 25% or less");
1025   }
1026
1027
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1037

low SEVERITY

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

Source File

- Token.sol

Locations

```
1036   function updateLimits() private {
1037   maxTransactionAmount = supply * transDigit / 100;
1038   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
1039   maxWallet = supply * walletDigit / 100;
1040   }
1041
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1037

low SEVERITY

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

Source File

- Token.sol

Locations

```
1036   function updateLimits() private {
1037   maxTransactionAmount = supply * transDigit / 100;
1038   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
1039   maxWallet = supply * walletDigit / 100;
1040   }
1041
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1038

low SEVERITY

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

Source File

- Token.sol

Locations

```
1037   maxTransactionAmount = supply * transDigit / 100;
1038   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
1039   maxWallet = supply * walletDigit / 100;
1040   }
1041
1042
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1038

low SEVERITY

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

Source File

- Token.sol

Locations

```
1037   maxTransactionAmount = supply * transDigit / 100;
1038   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
1039   maxWallet = supply * walletDigit / 100;
1040   }
1041
1042
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1039

low SEVERITY

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

Source File

- Token.sol

Locations

```
1038   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
1039   maxWallet = supply * walletDigit / 100;
1040   }
1041
1042   function setAutomatedMarketMakerPair(address pair, bool value) public onlyOwner {
1043
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1039

low SEVERITY

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

Source File

- Token.sol

Locations

```
1038   swapTokensAtAmount = supply * 5 / 10000; // 0.05% swap wallet;
1039   maxWallet = supply * walletDigit / 100;
1040   }
1041
1042   function setAutomatedMarketMakerPair(address pair, bool value) public onlyOwner {
1043
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1087

low SEVERITY

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

Source File

- Token.sol

Locations

```
1086   require(_holderLastTransferTimestamp[tx.origin] < block.number, "_transfer::
Transfer Delay enabled.  Only one purchase per block allowed.");
1087   _holderLastTransferTimestamp[tx.origin] = block.number + delayDigit;
1088   }
1089   }
1090
1091
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1094

low SEVERITY

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

Source File

- Token.sol

Locations

```
1093   require(amount <= maxTransactionAmount, "Buy transfer amount exceeds the
maxTransactionAmount.");
1094   require(amount + balanceOf(to) <= maxWallet, "Max wallet exceeded");
1095   }
1096
1097   //when sell
1098
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1102

low SEVERITY

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

Source File

- Token.sol

Locations

```
1101   else if(!_isExcludedMaxTransactionAmount[to]){
1102   require(amount + balanceOf(to) <= maxWallet, "Max wallet exceeded");
1103   }
1104   }
1105   }
1106
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1137

low SEVERITY

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

Source File

- Token.sol

Locations

```
1136   fees = amount.mul(sellTotalFees).div(100);
1137   tokensForBurn += fees * sellBurnFee / sellTotalFees;
1138   tokensForDev += fees * sellDevFee / sellTotalFees;
1139   }
1140
1141
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1137

low SEVERITY

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

Source File

- Token.sol

Locations

```
1136   fees = amount.mul(sellTotalFees).div(100);
1137   tokensForBurn += fees * sellBurnFee / sellTotalFees;
1138   tokensForDev += fees * sellDevFee / sellTotalFees;
1139   }
1140
1141
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1137

low SEVERITY

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

Source File

- Token.sol

Locations

```
1136   fees = amount.mul(sellTotalFees).div(100);
1137   tokensForBurn += fees * sellBurnFee / sellTotalFees;
1138   tokensForDev += fees * sellDevFee / sellTotalFees;
1139   }
1140
1141
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1138

low SEVERITY

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

Source File

- Token.sol

Locations

```
1137   tokensForBurn += fees * sellBurnFee / sellTotalFees;
1138   tokensForDev += fees * sellDevFee / sellTotalFees;
1139   }
1140
1141   // on buy
1142
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1138

low SEVERITY

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

Source File

- Token.sol

Locations

```
1137   tokensForBurn += fees * sellBurnFee / sellTotalFees;
1138   tokensForDev += fees * sellDevFee / sellTotalFees;
1139   }
1140
1141   // on buy
1142
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1138

low SEVERITY

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

Source File

- Token.sol

Locations

```
1137   tokensForBurn += fees * sellBurnFee / sellTotalFees;
1138   tokensForDev += fees * sellDevFee / sellTotalFees;
1139   }
1140
1141   // on buy
1142
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1145

low SEVERITY

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

Source File

- Token.sol

Locations

```
1144   fees = amount.mul(buyTotalFees).div(100);
1145   tokensForBurn += fees * buyBurnFee / buyTotalFees;
1146   tokensForDev += fees * buyDevFee / buyTotalFees;
1147   }
1148
1149
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1145

low SEVERITY

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

Source File

- Token.sol

Locations

```
1144   fees = amount.mul(buyTotalFees).div(100);
1145   tokensForBurn += fees * buyBurnFee / buyTotalFees;
1146   tokensForDev += fees * buyDevFee / buyTotalFees;
1147   }
1148
1149
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1145

low SEVERITY

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

Source File

- Token.sol

Locations

```
1144   fees = amount.mul(buyTotalFees).div(100);
1145   tokensForBurn += fees * buyBurnFee / buyTotalFees;
1146   tokensForDev += fees * buyDevFee / buyTotalFees;
1147   }
1148
1149
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1146

low SEVERITY

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

Source File

- Token.sol

Locations

```
1145   tokensForBurn += fees * buyBurnFee / buyTotalFees;
1146   tokensForDev += fees * buyDevFee / buyTotalFees;
1147   }
1148
1149   if(fees > 0){
1150
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1146

low SEVERITY

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

Source File

- Token.sol

Locations

```
1145   tokensForBurn += fees * buyBurnFee / buyTotalFees;
1146   tokensForDev += fees * buyDevFee / buyTotalFees;
1147   }
1148
1149   if(fees > 0){
1150
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1146

low SEVERITY

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

Source File

- Token.sol

Locations

```
1145   tokensForBurn += fees * buyBurnFee / buyTotalFees;
1146   tokensForDev += fees * buyDevFee / buyTotalFees;
1147   }
1148
1149   if(fees > 0){
1150
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1159

low SEVERITY

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

Source File

- Token.sol

Locations

```
1158
1159   amount -= fees;
1160   }
1161
1162   super._transfer(from, to, amount);
1163
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1191

low SEVERITY

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

Source File

- Token.sol

Locations

```
1190
1191   if(contractBalance > swapTokensAtAmount * 20){
1192   contractBalance = swapTokensAtAmount * 20;
1193   }
1194
1195
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1192

low SEVERITY

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

Source File

- Token.sol

Locations

```
1191   if(contractBalance > swapTokensAtAmount * 20){
1192   contractBalance = swapTokensAtAmount * 20;
1193   }
1194
1195   swapTokensForEth(contractBalance);
1196
```

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

LINE 1086

low SEVERITY

The tx.origin environment variable has been found to influence a control flow decision. Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

Source File

- Token.sol

Locations

```
1085   if (to != owner() && to != address(uniswapV2Router) && to !=
address(uniswapV2Pair)){
1086   require(_holderLastTransferTimestamp[tx.origin] < block.number, "_transfer::
Transfer Delay enabled.  Only one purchase per block allowed.");
1087   _holderLastTransferTimestamp[tx.origin] = block.number + delayDigit;
1088   }
1089   }
1090
```

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

LINE 1087

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

- Token.sol

Locations

```
1086   require(_holderLastTransferTimestamp[tx.origin] < block.number, "_transfer::
Transfer Delay enabled.  Only one purchase per block allowed.");
1087   _holderLastTransferTimestamp[tx.origin] = block.number + delayDigit;
1088   }
1089   }
1090
1091
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1169

low SEVERITY

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

Source File

- Token.sol

Locations

```
1168   address[] memory path = new address[](2);
1169   path[0] = address(this);
1170   path[1] = uniswapV2Router.WETH();
1171
1172   _approve(address(this), address(uniswapV2Router), tokenAmount);
1173
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1170

low SEVERITY

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

Source File

- Token.sol

Locations

```
1169   path[0] = address(this);
1170   path[1] = uniswapV2Router.WETH();
1171
1172   _approve(address(this), address(uniswapV2Router), tokenAmount);
1173
1174
```

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

LINE 1086

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

- Token.sol

Locations

```
1085   if (to != owner() && to != address(uniswapV2Router) && to !=
address(uniswapV2Pair)){
1086   require(_holderLastTransferTimestamp[tx.origin] < block.number, "_transfer::
Transfer Delay enabled.  Only one purchase per block allowed.");
1087   _holderLastTransferTimestamp[tx.origin] = block.number + delayDigit;
1088   }
1089   }
1090
```

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

LINE 1087

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

- Token.sol

Locations

```
1086   require(_holderLastTransferTimestamp[tx.origin] < block.number, "_transfer::
Transfer Delay enabled.  Only one purchase per block allowed.");
1087   _holderLastTransferTimestamp[tx.origin] = block.number + delayDigit;
1088   }
1089   }
1090
1091
```

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.