



PYROmatic

Smart Contract Audit Report

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
PYROmatic	PYRO	Ethereum

Addresses

Contract address	0x1e2d230c7a7f4c679fb1378f1f51dedeae85cd72
Contract deployer address	0xA486120564D67599dEc94AdB84DF9dee98d76D26

Project Website

https://www.pyrotokenerc.com/

Codebase

https://etherscan.io/address/0x1e2d230c7a7f4c679fb1378f1f51dedeae85cd72#code

SUMMARY

PYRO is a fast-burning hyper-deflationary token that gains value with every buy and sell. The burn in the PYRO contract is a true burn function that removes tokens from the total supply with every buy and sell transaction. In addition to this, the PYRO team has developed a proprietary burn bot for use by other projects and developers who wish to utilize burn functions within their smart contracts allowing them to join the PYRO burn bot ecosystem. PYRO's number one goal is to become the fastest burning token on the ERC20 blockchain and the go-to utility provider for the PYRO burn bots and custom true burn contracts.

Contract Summary

Documentation Quality

PYROmatic provides a very good documentation with 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 standard.

- Standard solidity basecode and rules are already followed by PYROmatic 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 106, 160 and 172.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 123, 123, 167, 167, 168, 168, 293, 321, 361, 361, 392, 393, 402, 402, 402, 402, 403, 403, 407, 407, 407, 408, 408, 412, 412, 416, 416, 420, 420, 424, 424, 425, 425, 427, 427, 428, 429, 485, 499, 499, 560, 560, 561, 561, 578, 579, 579, 580, 580, 594, 596, 609, 622, 622, 623, 623, 624, 626, 633, 638, 638, 640 and 645.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 6.
- SWC-110 SWC-123 | It is recommended to use of revert(), assert(), and require() in Solidity, and the new REVERT opcode in the EVM on lines 520, 521, 579, 580 and 580.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 446.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 557.

CONCLUSION

We have audited the PYROmatic project released on November 2022 to discover issues and identify potential security vulnerabilities in PYROmatic 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 PYROmatic smart contract code do not pose a considerable risk. The writing of the contract is close to the standard of writing contracts in general. The low-risk issues found are some arithmetic operation issues, a floating pragma is set, a state variable visibility is not set, weak sources of randomness, tx.origin as a part of authorization control and out of bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

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

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	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 grieving attacks can be performed on contracts which accept data and use it in a sub-call on another contract.	PASS
Arbitrary Jump Function	SWC-127	As Solidity doesnt support pointer arithmetics, it is impossible to change such variable to an arbitrary value.	PASS

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

SMART CONTRACT ANALYSIS

Started	Monday Nov 28 2022 21:33:39 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Nov 29 2022 07:49:18 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	PYROmatic.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	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-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-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-120	POTENTIAL USE OF "BLOCK.NUMBER" AS SOURCE OF RANDOMNESS.	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 123

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
122  uint8 constant private _decimals = 18;
123  uint256 private _tTotal = startingSupply * 10**_decimals;
124
125  struct Fees {
126    uint16 buyFee;
127
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 123

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
122  uint8 constant private _decimals = 18;
123  uint256 private _tTotal = startingSupply * 10**_decimals;
124
125  struct Fees {
126    uint16 buyFee;
127
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 167

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
166
167  uint256 private _maxTxAmount = (_tTotal * 2) / 100;
168  uint256 private _maxWalletSize = (_tTotal * 2) / 100;
169
170  bool public tradingEnabled = false;
171
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 167

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
166
167  uint256 private _maxTxAmount = (_tTotal * 2) / 100;
168  uint256 private _maxWalletSize = (_tTotal * 2) / 100;
169
170  bool public tradingEnabled = false;
171
```


SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 168

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
167 uint256 private _maxTxAmount = (_tTotal * 2) / 100;
168 uint256 private _maxWalletSize = (_tTotal * 2) / 100;
169
170 bool public tradingEnabled = false;
171 bool public _hasLiqBeenAdded = false;
172
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 168

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
167  uint256 private _maxTxAmount = (_tTotal * 2) / 100;
168  uint256 private _maxWalletSize = (_tTotal * 2) / 100;
169
170  bool public tradingEnabled = false;
171  bool public _hasLiqBeenAdded = false;
172
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 293

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
292     if (_allowances[sender][msg.sender] != type(uint256).max) {  
293         _allowances[sender][msg.sender] -= amount;  
294     }  
295  
296     return _transfer(sender, recipient, amount);  
297
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 321

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
320   if (timeSinceLastPair != 0) {  
321       require(block.timestamp - timeSinceLastPair > 3 days, "3 Day cooldown.");  
322   }  
323   require(!lpPairs[pair], "Pair already added to list.");  
324   lpPairs[pair] = true;  
325
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 361

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
360 function getCirculatingSupply() public view returns (uint256) {  
361     return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));  
362 }  
363  
364 function removeSniper(address account) external onlyOwner {  
365
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 361

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
360 function getCirculatingSupply() public view returns (uint256) {  
361     return (_tTotal - (balanceOf(DEAD) + balanceOf(address(0))));  
362 }  
363  
364 function removeSniper(address account) external onlyOwner {  
365
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 392

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
391  _ratios.totalSwap = marketing;  
392  uint256 total = _taxRates.buyFee + _taxRates.sellFee;  
393  require(_ratios.totalSwap + _ratios.burn <= total, "Cannot exceed sum of buy and  
sell fees.");  
394  }  
395  
396
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 393

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
392     uint256 total = _taxRates.buyFee + _taxRates.sellFee;
393     require(_ratios.totalSwap + _ratios.burn <= total, "Cannot exceed sum of buy and
sell fees.");
394 }
395
396 function setWallets(address payable marketing) external onlyOwner {
397
```


SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 402

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
401     function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {  
402         require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt  
must be above 0.5% of total supply.");  
403         _maxTxAmount = (_tTotal * percent) / divisor;  
404     }  
405  
406
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 402

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
401     function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {  
402         require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt  
must be above 0.5% of total supply.");  
403         _maxTxAmount = (_tTotal * percent) / divisor;  
404     }  
405  
406
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 402

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
401     function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {  
402         require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt  
must be above 0.5% of total supply.");  
403         _maxTxAmount = (_tTotal * percent) / divisor;  
404     }  
405  
406
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 402

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
401     function setMaxTxPercent(uint256 percent, uint256 divisor) external onlyOwner {  
402         require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt  
must be above 0.5% of total supply.");  
403         _maxTxAmount = (_tTotal * percent) / divisor;  
404     }  
405  
406
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 403

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
402     require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
403     _maxTxAmount = (_tTotal * percent) / divisor;
404 }
405
406 function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
407
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 403

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
402     require((_tTotal * percent) / divisor >= (_tTotal * 5 / 1000), "Max Transaction amt
must be above 0.5% of total supply.");
403     _maxTxAmount = (_tTotal * percent) / divisor;
404 }
405
406 function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {
407
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 407

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
406     function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {  
407         require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be  
         above 1% of total supply.");  
408         _maxWalletSize = (_tTotal * percent) / divisor;  
409     }  
410  
411
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 407

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
406     function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {  
407         require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be  
         above 1% of total supply.");  
408         _maxWalletSize = (_tTotal * percent) / divisor;  
409     }  
410  
411
```


SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 407

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
406 function setMaxWalletSize(uint256 percent, uint256 divisor) external onlyOwner {  
407     require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be  
above 1% of total supply.");  
408     _maxWalletSize = (_tTotal * percent) / divisor;  
409 }  
410  
411
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 408

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
407     require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
408     _maxWalletSize = (_tTotal * percent) / divisor;
409 }
410
411 function getMaxTX() external view returns (uint256) {
412
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 408

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
407   require((_tTotal * percent) / divisor >= (_tTotal / 100), "Max Wallet amt must be
above 1% of total supply.");
408   _maxWalletSize = (_tTotal * percent) / divisor;
409   }
410
411   function getMaxTX() external view returns (uint256) {
412
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 412

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
411     function getMaxTX() external view returns (uint256) {  
412         return _maxTxAmount / (10**_decimals);  
413     }  
414  
415     function getMaxWallet() external view returns (uint256) {  
416
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 412

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
411     function getMaxTX() external view returns (uint256) {
412         return _maxTxAmount / (10**_decimals);
413     }
414
415     function getMaxWallet() external view returns (uint256) {
416
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 416

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
415     function getMaxWallet() external view returns (uint256) {  
416         return _maxWalletSize / (10**_decimals);  
417     }  
418  
419     function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view  
returns (uint256) {  
420
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 416

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
415     function getMaxWallet() external view returns (uint256) {  
416         return _maxWalletSize / (10**_decimals);  
417     }  
418  
419     function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view  
returns (uint256) {  
420
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 420

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
419     function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
    returns (uint256) {
420         return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
421     }
422
423     function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
    uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
424
```


SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 420

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
419     function getTokenAmountAtPriceImpact(uint256 priceImpactInHundreds) external view
    returns (uint256) {
420         return((balanceOf(lpPair) * priceImpactInHundreds) / masterTaxDivisor);
421     }
422
423     function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
    uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
424
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 424

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
423     function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
424         swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
425         swapAmount = (_tTotal * amountPercent) / amountDivisor;
426         require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
427         require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
428     }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 424

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
423     function setSwapSettings(uint256 thresholdPercent, uint256 thresholdDivisor,
uint256 amountPercent, uint256 amountDivisor) external onlyOwner {
424         swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
425         swapAmount = (_tTotal * amountPercent) / amountDivisor;
426         require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
427         require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
428     }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 425

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
424     swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
425     swapAmount = (_tTotal * amountPercent) / amountDivisor;
426     require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
427     require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
428     require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
429
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 425

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
424     swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor;
425     swapAmount = (_tTotal * amountPercent) / amountDivisor;
426     require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
427     require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
428     require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
429
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 427

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
426     require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
427     require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
428     require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
429     require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
430 }
431
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 427

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
426     require(swapThreshold <= swapAmount, "Threshold cannot be above amount.");
427     require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
428     require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
429     require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
430 }
431
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 428

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
427     require(swapAmount <= (balanceOf(lpPair) * 150) / masterTaxDivisor, "Cannot be
above 1.5% of current PI.");
428     require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
429     require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
430 }
431
432
```


SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 429

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
428     require(swapAmount >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of total
supply.");
429     require(swapThreshold >= _tTotal / 1_000_000, "Cannot be lower than 0.00001% of
total supply.");
430 }
431
432 function setPriceImpactSwapAmount(uint256 priceImpactSwapPercent) external
onlyOwner {
433
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 485

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
484     if (!_isExcludedFromLimits[to]) {  
485         require(balanceOf(to) + amount <= _maxWalletSize, "Transfer amount exceeds the  
maxWalletSize.");  
486     }  
487 }  
488 }  
489
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 499

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
498     uint256 swapAmt = swapAmount;
499     if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
500     if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
501     contractSwap(contractTokenBalance);
502 }
503
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 499

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
498     uint256 swapAmt = swapAmount;
499     if (piContractSwapsEnabled) { swapAmt = (balanceOf(lpPair) * piSwapPercent) /
masterTaxDivisor; }
500     if (contractTokenBalance >= swapAmt) { contractTokenBalance = swapAmt; }
501     contractSwap(contractTokenBalance);
502 }
503
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 560

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
559     allowedPresaleExclusion = false;
560     swapThreshold = (balanceOf(lpPair) * 10) / 10000;
561     swapAmount = (balanceOf(lpPair) * 30) / 10000;
562     launchStamp = block.timestamp;
563 }
564
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 560

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
559     allowedPresaleExclusion = false;
560     swapThreshold = (balanceOf(lpPair) * 10) / 10000;
561     swapAmount = (balanceOf(lpPair) * 30) / 10000;
562     launchStamp = block.timestamp;
563 }
564
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 561

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
560     swapThreshold = (balanceOf(lpPair) * 10) / 10000;  
561     swapAmount = (balanceOf(lpPair) * 30) / 10000;  
562     launchStamp = block.timestamp;  
563     }  
564  
565
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 561

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
560     swapThreshold = (balanceOf(lpPair) * 10) / 10000;  
561     swapAmount = (balanceOf(lpPair) * 30) / 10000;  
562     launchStamp = block.timestamp;  
563     }  
564  
565
```


SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 578

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
577     require(accounts.length == amounts.length, "Lengths do not match.");
578     for (uint16 i = 0; i < accounts.length; i++) {
579         require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
580         finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
581     }
582
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 579

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
578   for (uint16 i = 0; i < accounts.length; i++) {  
579       require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");  
580       finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,  
true);  
581   }  
582   }  
583
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 579

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
578   for (uint16 i = 0; i < accounts.length; i++) {  
579       require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");  
580       finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,  
true);  
581   }  
582   }  
583
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 580

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
579     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
580     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
581   }
582 }
583
584
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 580

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
579     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
580     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
581   }
582   }
583
584
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 594

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
593     }  
594     _tOwned[from] -= amount;  
595     uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;  
596     _tOwned[to] += amountReceived;  
597     emit Transfer(from, to, amountReceived);  
598
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 596

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
595     uint256 amountReceived = (takeFee) ? takeTaxes(from, buy, sell, amount) : amount;
596     _tOwned[to] += amountReceived;
597     emit Transfer(from, to, amountReceived);
598     if (!_hasLiqBeenAdded) {
599         _checkLiquidityAdd(from, to);
600     }
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 609

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
608 Ratios memory ratios = _ratios;  
609 uint256 total = _ratios.marketing + _ratios.burn;  
610 uint256 currentFee;  
611 if (buy) {  
612     currentFee = _taxRates.buyFee;  
613 }
```


SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 622

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
621  || block.chainid == 56)) { currentFee = 4500; }
622  uint256 feeAmount = amount * currentFee / masterTaxDivisor;
623  uint256 burnAmount = (feeAmount * ratios.burn) / total;
624  uint256 swapAmt = feeAmount - burnAmount;
625  if (swapAmt > 0) {
626
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 622

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
621  || block.chainid == 56)) { currentFee = 4500; }
622  uint256 feeAmount = amount * currentFee / masterTaxDivisor;
623  uint256 burnAmount = (feeAmount * ratios.burn) / total;
624  uint256 swapAmt = feeAmount - burnAmount;
625  if (swapAmt > 0) {
626
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 623

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
622  uint256 feeAmount = amount * currentFee / masterTaxDivisor;
623  uint256 burnAmount = (feeAmount * ratios.burn) / total;
624  uint256 swapAmt = feeAmount - burnAmount;
625  if (swapAmt > 0) {
626    _tOwned[address(this)] += swapAmt;
627
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 623

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
622  uint256 feeAmount = amount * currentFee / masterTaxDivisor;
623  uint256 burnAmount = (feeAmount * ratios.burn) / total;
624  uint256 swapAmt = feeAmount - burnAmount;
625  if (swapAmt > 0) {
626    _tOwned[address(this)] += swapAmt;
627  }
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 624

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
623     uint256 burnAmount = (feeAmount * ratios.burn) / total;
624     uint256 swapAmt = feeAmount - burnAmount;
625     if (swapAmt > 0) {
626         _tOwned[address(this)] += swapAmt;
627         emit Transfer(from, address(this), swapAmt);
628     }
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 626

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
625     if (swapAmt > 0) {  
626         _tOwned[address(this)] += swapAmt;  
627         emit Transfer(from, address(this), swapAmt);  
628     }  
629     if (burnAmount > 0) {  
630
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 633

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
632
633   return amount - feeAmount;
634 }
635
636 function burn(uint256 amountTokens) external {
637
```

SWC-101 | ARITHMETIC OPERATION "*"=" DISCOVERED

LINE 638

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
637 address sender = msg.sender;
638 amountTokens *= 10**_decimals;
639 require(balanceOf(sender) >= amountTokens, "You do not have enough tokens.");
640 _tOwned[sender] -= amountTokens;
641 _burn(sender, amountTokens);
642
```


SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 638

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
637     address sender = msg.sender;
638     amountTokens *= 10**_decimals;
639     require(balanceOf(sender) >= amountTokens, "You do not have enough tokens.");
640     _tOwned[sender] -= amountTokens;
641     _burn(sender, amountTokens);
642
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 640

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
639     require(balanceOf(sender) >= amountTokens, "You do not have enough tokens.");
640     _tOwned[sender] -= amountTokens;
641     _burn(sender, amountTokens);
642 }
643
644
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 645

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
644 function _burn(address from, uint256 amount) internal {  
645     _tTotal -= amount;  
646     emit Transfer(from, address(0), amount);  
647 }  
648 }  
649
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 6

low SEVERITY

The current pragma Solidity directive is `">=0.6.0<0.9.0"`. 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

- PYROmatic.sol

Locations

```
5 // SPDX-License-Identifier: MIT
6 pragma solidity >=0.6.0 <0.9.0;
7
8 interface IERC20 {
9     function totalSupply() external view returns (uint256);
10
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 106

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
105 mapping (address => uint256) private _tOwned;
106 mapping (address => bool) lpPairs;
107 uint256 private timeSinceLastPair = 0;
108 mapping (address => mapping (address => uint256)) private _allowances;
109 mapping (address => bool) private _liquidityHolders;
110
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 160

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
159
160  bool inSwap;
161  bool public contractSwapEnabled = false;
162  uint256 public swapThreshold;
163  uint256 public swapAmount;
164
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 172

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
171  bool public _hasLiqBeenAdded = false;
172  Protections protections;
173  uint256 public launchStamp;
174
175  event ContractSwapEnabledUpdated(bool enabled);
176
```

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

LINE 446

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

- PYROmatic.sol

Locations

```
445    && to != _owner
446    && tx.origin != _owner
447    && !_liquidityHolders[to]
448    && !_liquidityHolders[from]
449    && to != DEAD
450
```


SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 520

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
519     address[] memory path = new address[](2);
520     path[0] = address(this);
521     path[1] = dexRouter.WETH();
522
523     try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
524
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 521

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
520  path[0] = address(this);  
521  path[1] = dexRouter.WETH();  
522  
523  try dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(  
524  contractTokenBalance,  
525
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 579

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
578   for (uint16 i = 0; i < accounts.length; i++) {  
579       require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");  
580       finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,  
true);  
581   }  
582   }  
583
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 580

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
579     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
580     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
581   }
582 }
583
584
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 580

low SEVERITY

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

Source File

- PYROmatic.sol

Locations

```
579     require(balanceOf(msg.sender) >= amounts[i]*10**_decimals, "Not enough tokens.");
580     finalizeTransfer(msg.sender, accounts[i], amounts[i]*10**_decimals, false, false,
true);
581   }
582 }
583
584
```

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

LINE 557

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

- PYROmatic.sol

Locations

```
556     }  
557     try protections.setLaunch(lpPair, uint32(block.number), uint64(block.timestamp),  
_decimals) {} catch {}  
558     tradingEnabled = true;  
559     allowedPresaleExclusion = false;  
560     swapThreshold = (balanceOf(lpPair) * 10) / 10000;  
561
```

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.