Safemoon chain

Smart Contract Audit Report

SYSFIXED

TABLE OF CONTENTS

SYSFIXED

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Safemoon chainn	SMC	Binance Smart Chain

Addresses

Contract address	0xef4429cE2C87aa3fDFf5aD910f41dD557382Cc39
Contract deployer address	0xd3Be29D3CeeC37682119cf5B35aaC41543Cd423D

Project Website

https://www.safemoonchain.org/

Codebase

https://bscscan.com/address/0xef4429cE2C87aa3fDFf5aD910f41dD557382Cc39#code

SUMMARY

Did you Miss Dogechain? & Shibchain? Then you won't want to miss Safemoon Chain. Safemoon Chain will speed up Blockchain adoption by connecting systems like Internet Web data, Dapps and Web 5 softwares, IOT-internet of databases, Machine Learning and Artificial Intelligence.

Contract Summary

Documentation Quality

Safemoon chain 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 Safemoon chain 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 194, 216, 241, 270, 271, 400, 400, 431, 431, 462, 472, 483, 511, 520, 526, 535, 535, 542, 546, 546, 566, 567, 567, 569, 575, 576, 576, 577, 584, 584, 633, 633 and 659.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 7.
- 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 595, 596 and 660.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 511 and 640.

CONCLUSION

We have audited the Safemoon chain project released on January 2023 to discover issues and identify potential security vulnerabilities in Safemoon chain 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 Safemoon chain 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 set, 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. We recommend to Don't using any of those environment variables as sources of randomness and being aware that the use of these variables introduces a certain level of trust in miners.

AUDIT RESULT

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Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.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

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	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	Monday Jan 30 2023 19:17:48 GMT+0000 (Coordinated Universal Time)
Finished	Tuesday Jan 31 2023 21:25:08 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	SafeMoonChain.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-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-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

SYSFIXED

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 194

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
193    require(currentAllowance >= amount, "BEP20: transfer amount exceeds allowance");
194    _approve(sender, _msgSender(), currentAllowance - amount);
195
196    return true;
197    }
198
```

SYSFIXED

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED
LINE 216

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
215    {
216    _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
217    return true;
218    }
219
220
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 241

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
240    require(currentAllowance >= subtractedValue, "BEP20: decreased allowance below
zero");
241    _approve(_msgSender(), spender, currentAllowance - subtractedValue);
242
243    return true;
244    }
245
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 270

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
269    require(senderBalance >= amount, "BEP20: transfer amount exceeds balance");
270    _balances[sender] = senderBalance - amount;
271    _balances[recipient] += amount;
272
273    emit Transfer(sender, recipient, amount);
274
```

SYSFIXED

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED
LINE 271

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
270    _balances[sender] = senderBalance - amount;
271    _balances[recipient] += amount;
272
273    emit Transfer(sender, recipient, amount);
274    }
275
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 400

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
399
400    uint256 public tokenLiquidityThreshold = 21e3 * 10**18;
401
402    uint256 public genesis_block;
403    uint256 private deadline = 3;
404
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 400

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
399
400   uint256 public tokenLiquidityThreshold = 21e3 * 10**18;
401
402   uint256 public genesis_block;
403   uint256 private deadline = 3;
404
```

SYSFIXED

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 431

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
430    constructor() BEP20("Safemoon chain", "SMC") {
431    _tokengeneration(msg.sender, 21e6 * 10**decimals());
432    exemptFee[msg.sender] = true;
433
434    IRouter _router = IRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
435
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 431

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
430    constructor() BEP20("Safemoon chain", "SMC") {
431    _tokengeneration(msg.sender, 21e6 * 10**decimals());
432    exemptFee[msg.sender] = true;
433
434    IRouter _router = IRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
435
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 462

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

Source File
- SafeMoonChain.sol

Locations

```
461    require(currentAllowance >= amount, "BEP20: transfer amount exceeds allowance");
462    _approve(sender, _msgSender(), currentAllowance - amount);
463
464    return true;
465    }
466
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED
LINE 472

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

Source File
- SafeMoonChain.sol

Locations

```
471    {
472    _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
473    return true;
474    }
475
476
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 483

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
482   require(currentAllowance >= subtractedValue, "BEP20: decreased allowance below
zero");
483   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
484
485   return true;
486   }
487
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED
LINE 511

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

Source File
- SafeMoonChain.sol

Locations

```
510    !exemptFee[recipient] &&
511    block.number < genesis_block + deadline;
512
513    //set fee to zero if fees in contract are handled or exempted
514    if (_interlock || exemptFee[sender] || exemptFee[recipient])
515
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED
LINE 520

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
519    feeswap =
520    sellTaxes.liquidity +
521    sellTaxes.marketing;
522    feesum = feeswap;
523    currentTaxes = sellTaxes;
524
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED
LINE 526

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
525    feeswap =
526    taxes.liquidity +
527    taxes.marketing;
528    feesum = feeswap;
529    currentTaxes = taxes;
530
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED
LINE 535

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
534
535    fee = (amount * feesum) / 100;
536
537    //send fees if threshold has been reached
538    //don't do this on buys, breaks swap
539
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 535

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
534
535    fee = (amount * feesum) / 100;
536
537    //send fees if threshold has been reached
538    //don't do this on buys, breaks swap
539
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 542

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
541    //rest to recipient
542    super._transfer(sender, recipient, amount - fee);
543    if (fee > 0) {
544    //send the fee to the contract
545    if (feeswap > 0) {
546
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED
LINE 546

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
545    if (feeswap > 0) {
546    uint256 feeAmount = (amount * feeswap) / 100;
547    super._transfer(sender, address(this), feeAmount);
548    }
549
550
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 546

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

Source File
- SafeMoonChain.sol

Locations

```
545    if (feeswap > 0) {
546    uint256 feeAmount = (amount * feeswap) / 100;
547    super._transfer(sender, address(this), feeAmount);
548    }
549
550
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 566

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
565    // Split the contract balance into halves
566    uint256 denominator = feeswap * 2;
567    uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /
568    denominator;
569    uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
570
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED
LINE 567

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
566    uint256 denominator = feeswap * 2;
567    uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /
568    denominator;
569    uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
570
571
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 567

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
566   uint256 denominator = feeswap * 2;
567   uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /
568   denominator;
569   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
570
571
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 569

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

Source File
- SafeMoonChain.sol

Locations

```
568    denominator;
569    uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
570
571    uint256 initialBalance = address(this).balance;
572
573
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 575

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
574
575    uint256 deltaBalance = address(this).balance - initialBalance;
576    uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
577    uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
578
579
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED
LINE 576

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
575    uint256 deltaBalance = address(this).balance - initialBalance;
576    uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
577    uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
578
579    if (ethToAddLiquidityWith > 0) {
580
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED
LINE 576

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

Source File
- SafeMoonChain.sol

Locations

```
575    uint256 deltaBalance = address(this).balance - initialBalance;
576    uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
577    uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
578
579    if (ethToAddLiquidityWith > 0) {
580
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 577

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
576    uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
577    uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
578
579    if (ethToAddLiquidityWith > 0) {
580    // Add liquidity to pancake
581
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 584

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
583
584    uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
585    if (marketingAmt > 0) {
586    payable(marketingWallet).sendValue(marketingAmt);
587    }
588
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 584

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
583
584    uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
585    if (marketingAmt > 0) {
586    payable(marketingWallet).sendValue(marketingAmt);
587    }
588
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED
LINE 633

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

Source File
- SafeMoonChain.sol

Locations

```
632   require(new_amount <= 21e4, "Swap threshold amount should be lower or equal to 1%
of tokens");
633   tokenLiquidityThreshold = new_amount * 10**decimals();
634   }
635
636   function EnableTrading() external onlyOwner {
637
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED
LINE 633

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
632    require(new_amount <= 21e4, "Swap threshold amount should be lower or equal to 1%
of tokens");
633    tokenLiquidityThreshold = new_amount * 10**decimals();
634    }
635
636    function EnableTrading() external onlyOwner {
637
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED
LINE 659

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
658    function bulkExemptFee(address[] memory accounts, bool state) external onlyOwner {
659    for (uint256 i = 0; i < accounts.length; i++) {
660    exemptFee[accounts[i]] = state;
661    }
662    }
663
```

SWC-103 | A FLOATING PRAGMA IS SET.
LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.8"". 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

- SafeMoonChain.sol

Locations

```
6
7    pragma solidity ^0.8.8;
8
9    abstract contract Context {
10     function _msgSender() internal view virtual returns (address) {
11
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 595

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
594    address[] memory path = new address[](2);
595    path[0] = address(this);
596    path[1] = router.WETH();
597
598    _approve(address(this), address(router), tokenAmount);
599
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 596

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
595    path[0] = address(this);
596    path[1] = router.WETH();
597
598    _approve(address(this), address(router), tokenAmount);
599
600
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS
LINE 660

low SEVERITY

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

Source File

- SafeMoonChain.sol

Locations

```
659    for (uint256 i = 0; i < accounts.length; i++) {
660    exemptFee[accounts[i]] = state;
661    }
662    }
663
664
```

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

LINE 511

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

- SafeMoonChain.sol

Locations

```
510    !exemptFee[recipient] &&
511    block.number < genesis_block + deadline;
512
513    //set fee to zero if fees in contract are handled or exempted
514    if (_interlock || exemptFee[sender] || exemptFee[recipient])
515
```

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

LINE 640

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

- SafeMoonChain.sol

Locations

```
639   providingLiquidity = true;
640   genesis_block = block.number;
641   }
642
643   function updatedeadline(uint256 _deadline) external onlyOwner {
644
```

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.