



SafuChain

# 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
SafuChain	SAFU	Binance Smart Chain

## Addresses

Contract address	0x79C25c639dcd79e34F9705045C35C546f2A4023E
Contract deployer address	0xb3a8262b2927Fc668A6D94452A84d89c0002402b

## Project Website

<https://https://safuchain.info/>

## Codebase

<https://bscscan.com/address/0x79C25c639dcd79e34F9705045C35C546f2A4023E#code>

# SUMMARY

SAFU tokens fuel the SafuChain Network as the fastest, most cost-effective & scalable comparison to any other Web 3 digital asset. Innovation at the finest form as we set the standardization in blockchain technology algorithms & 7 enhancements. Our initiative is to provide the most SAFU Blockchain for the revolution, ensuring investors are always 100% SAFU. No Private sale, (1% Buy | 1% Sell) | Double Manual Buy-back System's | Marketing Campaign's | Testnet Live |

## Contract Summary

### Documentation Quality

SafuChain 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 SafuChain 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 217, 242, 271, 303, 304, 450, 450, 481, 481, 521, 534, 549, 581, 589, 593, 601, 601, 609, 613, 613, 634, 635, 635, 637, 643, 644, 645, 646, 653, 653, 704, 704 and 733.
- 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 663, 664 and 734.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 581 and 711.

## CONCLUSION

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

The security audit report provides a satisfactory result with some low-risk issues.

The most common issue in writing code on contracts that do not pose a big risk is that writing on contracts is close to the standard of writing contracts in general. Some of the common issues that were found stated variable visibility are not set, a floating pragma is set, and the potential use of "block.number" as a source of randomness. We recommended specifying 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. Don't use any of those environment variables as sources of randomness; be aware that using 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
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Reentrancy	SWC-107	Check effect interaction pattern should be followed if the code performs recursive call.	PASS
Assert Violation	SWC-110 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	Delegate calls should only be allowed to trusted addresses.	PASS
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS

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

# SMART CONTRACT ANALYSIS

Started	Friday Jan 06 2023 20:03:05 GMT+0000 (Coordinated Universal Time)
Finished	Saturday Jan 07 2023 14:35:17 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Safuchain.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

## SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 217

### low SEVERITY

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

### Source File

- Safuchain.sol

### Locations

```
216 );  
217 _approve(sender, _msgSender(), currentAllowance - amount);  
218  
219 return true;  
220 }  
221
```

## SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 242

### low SEVERITY

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

### Source File

- Safuchain.sol

### Locations

```
241     spender,  
242     _allowances[_msgSender()][spender] + addedValue  
243     );  
244     return true;  
245     }  
246
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 271

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
270 );  
271 _approve(_msgSender(), spender, currentAllowance - subtractedValue);  
272  
273 return true;  
274 }  
275
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 303

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
302 );  
303 _balances[sender] = senderBalance - amount;  
304 _balances[recipient] += amount;  
305  
306 emit Transfer(sender, recipient, amount);  
307
```

## SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 304

### low SEVERITY

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

### Source File

- Safuchain.sol

### Locations

```
303  _balances[sender] = senderBalance - amount;  
304  _balances[recipient] += amount;  
305  
306  emit Transfer(sender, recipient, amount);  
307  }  
308
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 450

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
449
450  uint256 public tokenLiquidityThreshold = 1e6 * 10**18;
451
452  uint256 public genesis_block;
453  uint256 private deadline = 3;
454
```



# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 450

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
449
450  uint256 public tokenLiquidityThreshold = 1e6 * 10**18;
451
452  uint256 public genesis_block;
453  uint256 private deadline = 3;
454
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 481

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
480     constructor() BEP20("SafuChain", "SAFU") {
481         _tokengeneration(msg.sender, 1e9 * 10**decimals());
482         exemptFee[msg.sender] = true;
483
484         IRouter _router = IRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
485     }
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 481

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
480     constructor() BEP20("SafuChain", "SAFU") {
481         _tokengeneration(msg.sender, 1e9 * 10**decimals());
482         exemptFee[msg.sender] = true;
483
484         IRouter _router = IRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
485     }
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 521

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
520 );  
521 _approve(sender, _msgSender(), currentAllowance - amount);  
522  
523 return true;  
524 }  
525
```

## SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 534

### low SEVERITY

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

### Source File

- Safuchain.sol

### Locations

```
533     spender,  
534     _allowances[_msgSender()][spender] + addedValue  
535     );  
536     return true;  
537 }  
538
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 549

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
548 );  
549 _approve(_msgSender(), spender, currentAllowance - subtractedValue);  
550  
551 return true;  
552 }  
553
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 581

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
580     !exemptFee[recipient] &&  
581     block.number < genesis_block + deadline;  
582  
583     //set fee to zero if fees in contract are handled or exempted  
584     if (_interlock || exemptFee[sender] || exemptFee[recipient])  
585
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 589

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
588     else if (recipient == pair && !useLaunchFee) {
589         feeswap = sellTaxes.liquidity + sellTaxes.marketing;
590         feesum = feeswap;
591         currentTaxes = sellTaxes;
592     } else if (!useLaunchFee) {
593
```



# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 593

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
592     } else if (!useLaunchFee) {  
593     feeswap = taxes.liquidity + taxes.marketing;  
594     feesum = feeswap;  
595     currentTaxes = taxes;  
596     } else if (useLaunchFee) {  
597
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 601

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
600
601  fee = (amount * feesum) / 100;
602
603  //send fees if threshold has been reached
604  //don't do this on buys, breaks swap
605
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 601

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
600
601  fee = (amount * feesum) / 100;
602
603  //send fees if threshold has been reached
604  //don't do this on buys, breaks swap
605
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 609

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
608 //rest to recipient
609 super._transfer(sender, recipient, amount - fee);
610 if (fee > 0) {
611 //send the fee to the contract
612 if (feeswap > 0) {
613
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 613

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
612  if (feeswap > 0) {
613    uint256 feeAmount = (amount * feeswap) / 100;
614    super._transfer(sender, address(this), feeAmount);
615  }
616  }
617
```

## SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 613

### low SEVERITY

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

### Source File

- Safuchain.sol

### Locations

```
612  if (feeswap > 0) {
613    uint256 feeAmount = (amount * feeswap) / 100;
614    super._transfer(sender, address(this), feeAmount);
615  }
616  }
617
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 634

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
633 // Split the contract balance into halves
634 uint256 denominator = feeswap * 2;
635 uint256 tokensToAddLiquidityWith = (contractBalance *
636 swapTaxes.liquidity) / denominator;
637 uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
638
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 635

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
634 uint256 denominator = feeswap * 2;  
635 uint256 tokensToAddLiquidityWith = (contractBalance *  
636 swapTaxes.liquidity) / denominator;  
637 uint256 toSwap = contractBalance - tokensToAddLiquidityWith;  
638  
639
```



# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 635

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
634 uint256 denominator = feeswap * 2;  
635 uint256 tokensToAddLiquidityWith = (contractBalance *  
636 swapTaxes.liquidity) / denominator;  
637 uint256 toSwap = contractBalance - tokensToAddLiquidityWith;  
638  
639
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 637

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
636 swapTaxes.liquidity) / denominator;  
637 uint256 toSwap = contractBalance - tokensToAddLiquidityWith;  
638  
639 uint256 initialBalance = address(this).balance;  
640  
641
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 643

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
642
643  uint256 deltaBalance = address(this).balance - initialBalance;
644  uint256 unitBalance = deltaBalance /
645  (denominator - swapTaxes.liquidity);
646  uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
647
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 644

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
643  uint256 deltaBalance = address(this).balance - initialBalance;  
644  uint256 unitBalance = deltaBalance /  
645  (denominator - swapTaxes.liquidity);  
646  uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;  
647  
648
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 645

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
644 uint256 unitBalance = deltaBalance /  
645 (denominator - swapTaxes.liquidity);  
646 uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;  
647  
648 if (ethToAddLiquidityWith > 0) {  
649
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 646

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
645 (denominator - swapTaxes.liquidity);
646 uint256 ethToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
647
648 if (ethToAddLiquidityWith > 0) {
649     // Add liquidity to pancake
650 }
```

## SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 653

### low SEVERITY

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

### Source File

- Safuchain.sol

### Locations

```
652
653  uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
654  if (marketingAmt > 0) {
655    payable(marketingWallet).sendValue(marketingAmt);
656  }
657
```

# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 653

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
652
653  uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
654  if (marketingAmt > 0) {
655    payable(marketingWallet).sendValue(marketingAmt);
656  }
657
```



# SWC-101 | ARITHMETIC OPERATION "\*" DISCOVERED

LINE 704

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
703     );  
704     tokenLiquidityThreshold = new_amount * 10**decimals();  
705     }  
706  
707     function EnableTrading() external onlyOwner {  
708
```

# SWC-101 | ARITHMETIC OPERATION "\*\*" DISCOVERED

LINE 704

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
703 );  
704 tokenLiquidityThreshold = new_amount * 10**decimals();  
705 }  
706  
707 function EnableTrading() external onlyOwner {  
708
```

# SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 733

## low SEVERITY

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

## Source File

- Safuchain.sol

## Locations

```
732  {  
733  for (uint256 i = 0; i < accounts.length; i++) {  
734  exemptFee[accounts[i]] = state;  
735  }  
736  }  
737
```

## SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

### low SEVERITY

The current pragma Solidity directive is ""^0.8.17"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

### Source File

- Safuchain.sol

### Locations

```
6
7  pragma solidity ^0.8.17;
8
9  abstract contract Context {
10     function _msgSender() internal view virtual returns (address) {
11
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 663

### low SEVERITY

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

### Source File

- Safuchain.sol

### Locations

```
662 address[] memory path = new address[](2);
663 path[0] = address(this);
664 path[1] = router.WETH();
665
666 _approve(address(this), address(router), tokenAmount);
667
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 664

### low SEVERITY

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

### Source File

- Safuchain.sol

### Locations

```
663     path[0] = address(this);
664     path[1] = router.WETH();
665
666     _approve(address(this), address(router), tokenAmount);
667
668
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 734

### low SEVERITY

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

### Source File

- Safuchain.sol

### Locations

```
733     for (uint256 i = 0; i < accounts.length; i++) {  
734         exemptFee[accounts[i]] = state;  
735     }  
736 }  
737  
738
```

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

LINE 581

### 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

- Safuchain.sol

### Locations

```
580     !exemptFee[recipient] &&  
581     block.number < genesis_block + deadline;  
582  
583     //set fee to zero if fees in contract are handled or exempted  
584     if (_interlock || exemptFee[sender] || exemptFee[recipient])  
585
```



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

LINE 711

### 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

- Safuchain.sol

### Locations

```
710 providingLiquidity = true;
711 genesis_block = block.number;
712 }
713
714 function updateddeadline(uint256 _deadline) external onlyOwner {
715
```

# 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.