## Billiard Crypto Reward

# Smart Contract Audit Report

SYSFIXED

21 Feb 2023

# TABLE OF CONTENTS

# AUDITED DETAILS

## Audited Project

| Project name | Token ticker | Blockchain |
|---|---|---|
| Billiard Crypto Reward | BICR | Binance Smart Chain |

## Addresses

| | |
|---|---|
| **Contract address** | 0xedbac1830c1b3280882c73449198ebf6a35ede43 |
| **Contract deployer address** | 0xA7618C49B0C419969F012B3e44a6DA9281744cc3 |

## Project Website

**https://billiardcrypto.com/**

## Codebase

**https://bscscan.com/address/0xedbac1830c1b3280882c73449198ebf6a35ede43#code**

# SUMMARY

Billiard Crypto is a simple game, but it takes players a lot of practice to get used to it. The primary operations are drag and drop and correct angles. Different game modes, such as Solo, PvP, and Tournament, will be released gradually according to the schedule.

## Contract Summary

**Documentation Quality**

Billiard Crypto Reward 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 Billiard Crypto Reward 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 453.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 198, 220, 245, 276, 277, 292, 293, 315, 316, 460, 460, 529, 539, 550, 580, 589, 595, 604, 604, 611, 615, 615, 635, 636, 636, 638, 644, 645, 645, 647, 647, 655, 707, 707, 728, 736, 749, 767 and 770.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 11.
- 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 667, 668, 750, 768 and 771.
- SWC-120 | It is recommended to use external sources of randomness via oracles on lines 580 and 714.

# CONCLUSION

We have audited the Billiard Crypto Reward project released on February 2023 to discover issues and identify potential security vulnerabilities in Billiard Crypto Reward 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 satisfactory results with low-risk issues.

The issues found in the NamaFile 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, the potential use of "block.number" as a source 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. The current pragma Solidity directive is ""^0.8.17"". Specifying a fixed compiler version is recommended 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. It is best practice to set the visibility of state variables explicitly. The default visibility for "IERCliquidityPairToken" is internal. Other possible visibility settings are public and private. Potential use of "block.number" as a source of randomness, the environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number, and timestamp are predictable and can be manipulated by a malicious miner. Also, keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness; be aware that using these variables introduces a certain level of trust in miners.

# AUDIT RESULT

| Article | Category | Description | Result |
|---|---|---|---|
| Default Visibility | SWC-100<br>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<br>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<br>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<br>SWC-121<br>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 Feb 20 2023 21:05:16 GMT+0000 (Coordinated Universal Time) |
| **Finished** | Tuesday Feb 21 2023 08:06:48 GMT+0000 (Coordinated Universal Time) |
| **Mode** | Standard |
| **Main Source File** | BilliardCryptoReward.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-103 | A FLOATING PRAGMA IS SET. | low | acknowledged |
| SWC-108 | STATE VARIABLE VISIBILITY IS NOT 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-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 198

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
197   require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
198   _approve(sender, _msgSender(), currentAllowance - amount);
199
200   return true;
201   }
202
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 220

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
219   {
220   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
221   return true;
222   }
223
224
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 245

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
244   require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below
zero");
245   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
246
247   return true;
248   }
249
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 276

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
275   require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");
276   _balances[sender] = senderBalance - amount;
277   _balances[recipient] += amount;
278
279   emit Transfer(sender, recipient, amount);
280
```

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

LINE 277

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
276   _balances[sender] = senderBalance - amount;
277   _balances[recipient] += amount;
278
279   emit Transfer(sender, recipient, amount);
280   }
281
```

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

LINE 292

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
291
292   _totalSupply += amount;
293   _balances[account] += amount;
294   emit Transfer(address(0), account, amount);
295   }
296
```

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

LINE 293

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
292   _totalSupply += amount;
293   _balances[account] += amount;
294   emit Transfer(address(0), account, amount);
295   }
296
297
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 315

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
314   require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
315   _balances[account] = accountBalance - amount;
316   _totalSupply -= amount;
317
318   emit Transfer(account, address(0), amount);
319
```

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

LINE 316

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
315   _balances[account] = accountBalance - amount;
316   _totalSupply -= amount;
317
318   emit Transfer(account, address(0), amount);
319   }
320
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 460

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
459
460   uint256 public tokenLiquidityThreshold = 1e4 * 10**18;
461
462   uint256 public genesis_block;
463   uint256 private deadline = 1;
464
```

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

LINE 460

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
459
460   uint256 public tokenLiquidityThreshold = 1e4 * 10**18;
461
462   uint256 public genesis_block;
463   uint256 private deadline = 1;
464
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 529

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
528   require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
529   _approve(sender, _msgSender(), currentAllowance - amount);
530
531   return true;
532   }
533
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 539

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
538   {
539   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
540   return true;
541   }
542
543
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 550

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
549   require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below
zero");
550   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
551
552   return true;
553   }
554
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 580

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

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

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 589

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
588   feeswap =
589   sellTaxes.liquidity +
590   sellTaxes.marketing ;
591   feesum = feeswap;
592   currentTaxes = sellTaxes;
593
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 595

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
594   feeswap =
595   taxes.liquidity +
596   taxes.marketing ;
597   feesum = feeswap;
598   currentTaxes = taxes;
599
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 604

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
603
604   fee = (amount * feesum) / 100;
605
606   //send fees if threshold has been reached
607   //don't do this on buys, breaks swap
608
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 604

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
603
604   fee = (amount * feesum) / 100;
605
606   //send fees if threshold has been reached
607   //don't do this on buys, breaks swap
608
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 611

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
610   //rest to recipient
611   super._transfer(sender, recipient, amount - fee);
612   if (fee > 0) {
613   //send the fee to the contract
614   if (feeswap > 0) {
615
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 615

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
614   if (feeswap > 0) {
615   uint256 feeAmount = (amount * feeswap) / 100;
616   super._transfer(sender, address(this), feeAmount);
617   }
618
619
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 615

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
614   if (feeswap > 0) {
615   uint256 feeAmount = (amount * feeswap) / 100;
616   super._transfer(sender, address(this), feeAmount);
617   }
618
619
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 635

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

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

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 636

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
635   uint256 denominator = feeswap * 2;
636   uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /
637   denominator;
638   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
639
640
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 636

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
635   uint256 denominator = feeswap * 2;
636   uint256 tokensToAddLiquidityWith = (contractBalance * swapTaxes.liquidity) /
637   denominator;
638   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
639
640
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 638

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
637   denominator;
638   uint256 toSwap = contractBalance - tokensToAddLiquidityWith;
639
640   uint256 initialBalance = IERCliquidityPairToken.balanceOf(address(liquifier));
641
642
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 644

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
643
644   uint256 deltaBalance = IERCliquidityPairToken.balanceOf(address(liquifier)) -
initialBalance;
645   uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
646
647   uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
648
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 645

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
644   uint256 deltaBalance = IERCliquidityPairToken.balanceOf(address(liquifier)) -
initialBalance;
645   uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
646
647   uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
648   if (marketingAmt > 0) {
649
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 645

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
644   uint256 deltaBalance = IERCliquidityPairToken.balanceOf(address(liquifier)) -
initialBalance;
645   uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity);
646
647   uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
648   if (marketingAmt > 0) {
649
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 647

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
646
647   uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
648   if (marketingAmt > 0) {
649   IERCliquidityPairToken.transferFrom(address(liquifier), marketingWallet,
marketingAmt);
650   }
651
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 647

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
646
647   uint256 marketingAmt = unitBalance * 2 * swapTaxes.marketing;
648   if (marketingAmt > 0) {
649   IERCliquidityPairToken.transferFrom(address(liquifier), marketingWallet,
marketingAmt);
650   }
651
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 655

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
654
655   uint256 ethPairToAddLiquidityWith = unitBalance * swapTaxes.liquidity;
656   if (ethPairToAddLiquidityWith > 0) {
657   // Add liquidity to pancake
658   addLiquidity(tokensToAddLiquidityWith, ethPairToAddLiquidityWith);
659
```

# SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 707

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
706   require(new_amount <= 1e5, "Swap threshold amount should be lower or equal to 1% of
tokens");
707   tokenLiquidityThreshold = new_amount * 10**decimals();
708   }
709
710   function EnableTrading() external onlyOwner {
711
```

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

LINE 707

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
706   require(new_amount <= 1e5, "Swap threshold amount should be lower or equal to 1% of
tokens");
707   tokenLiquidityThreshold = new_amount * 10**decimals();
708   }
709
710   function EnableTrading() external onlyOwner {
711
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 728

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
727   taxes = Taxes(_marketing, _liquidity);
728   require((_marketing + _liquidity) <= 5, "Must keep fees at 5% or less");
729   }
730
731   function SetSellTaxes(
732
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 736

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
735   sellTaxes = Taxes(_marketing, _liquidity);
736   require((_marketing + _liquidity ) <= 5, "Must keep fees at 5% or less");
737   }
738
739   function updateMarketingWallet(address newWallet) external onlyOwner {
740
```

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

LINE 749

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
748   function bulkExemptFee(address[] memory accounts, bool state) external onlyOwner {
749   for (uint256 i = 0; i < accounts.length; i++) {
750   exemptFee[accounts[i]] = state;
751   }
752   }
753
```

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

LINE 767

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
766   ) public onlyOwner {
767   for (uint256 index; index < newAddr.length; index++) {
768   blackList[newAddr[index]] = true;
769   }
770   for (uint256 index; index < removedAddr.length; index++) {
771
```

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

LINE 770

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
769   }
770   for (uint256 index; index < removedAddr.length; index++) {
771   blackList[removedAddr[index]] = false;
772   }
773   }
774
```

# SWC-103 | A FLOATING PRAGMA IS SET.

LINE 11

## 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

- BilliardCryptoReward.sol

## Locations

```
10
11   pragma solidity ^0.8.17;
12
13   abstract contract Context {
14   function _msgSender() internal view virtual returns (address) {
15
```

# SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 453

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
452   address public liquidityPairToken;
453   IERC20 IERCliquidityPairToken;
454   Liquifier public liquifier;
455
456   bool private _interlock = false;
457
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 667

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
666   address[] memory path = new address[](2);
667   path[0] = address(this);
668   path[1] = liquidityPairToken;
669
670   _approve(address(this), address(router), tokenAmount);
671
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 668

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
667   path[0] = address(this);
668   path[1] = liquidityPairToken;
669
670   _approve(address(this), address(router), tokenAmount);
671
672
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 750

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
749   for (uint256 i = 0; i < accounts.length; i++) {
750   exemptFee[accounts[i]] = state;
751   }
752   }
753
754
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 768

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
767   for (uint256 index; index < newAddr.length; index++) {
768   blackList[newAddr[index]] = true;
769   }
770   for (uint256 index; index < removedAddr.length; index++) {
771   blackList[removedAddr[index]] = false;
772
```

# SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 771

## low SEVERITY

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

## Source File

- BilliardCryptoReward.sol

## Locations

```
770   for (uint256 index; index < removedAddr.length; index++) {
771   blackList[removedAddr[index]] = false;
772   }
773   }
774
775
```

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

LINE 580

## 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

- BilliardCryptoReward.sol

## Locations

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

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

LINE 714

## 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

- BilliardCryptoReward.sol

## Locations

```
713   providingLiquidity = true;
714   genesis_block = block.number;
715   }
716
717   function updatedeadline(uint256 _deadline) external onlyOwner {
718
```

# 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.