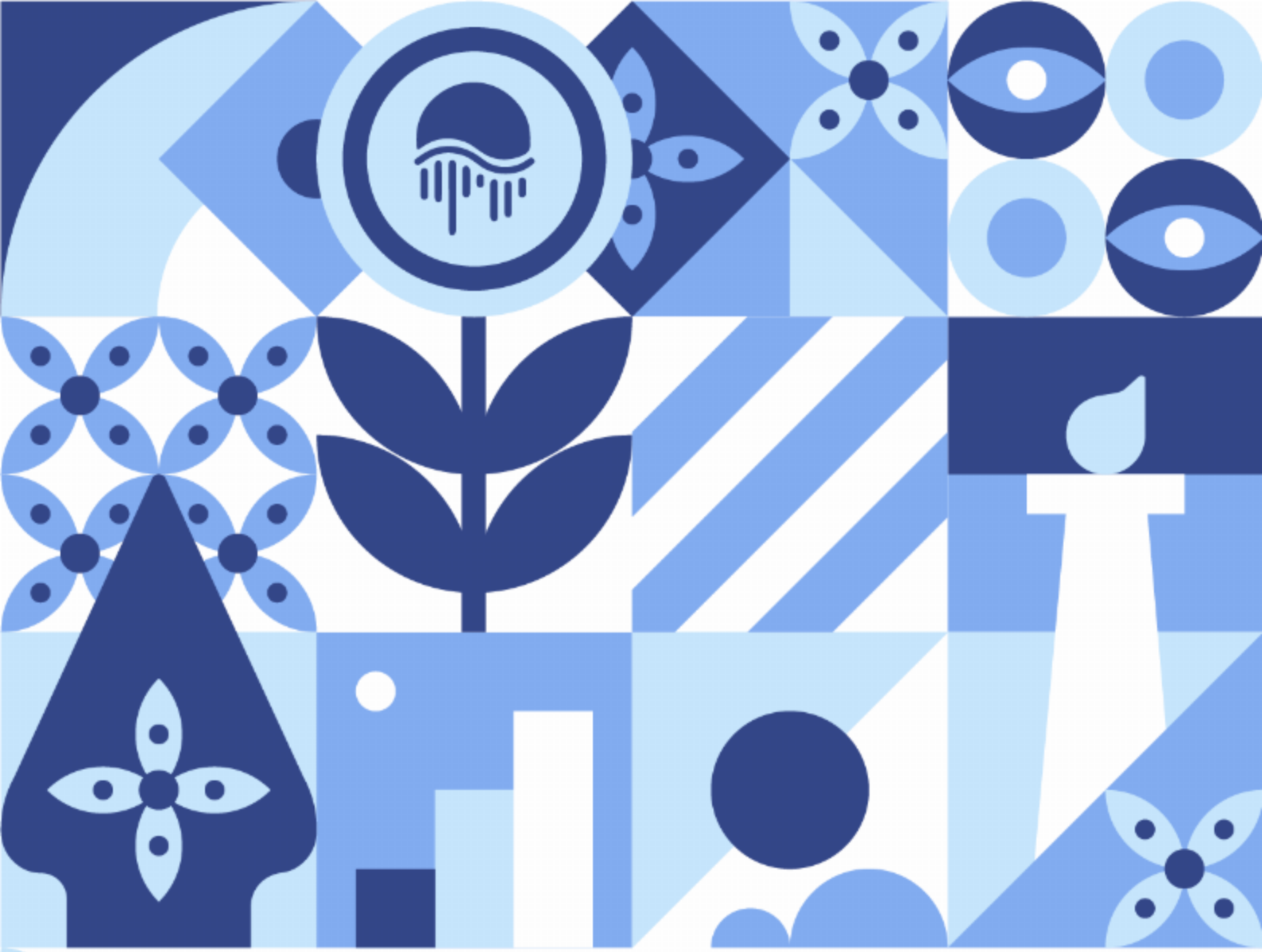

MoonBeans

Smart Contract Audit Report

SYSFIXED

08 Sep 2021

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
MoonBeans	BEANS	Moonriver

Addresses

Contract address	0xc2392dd3e3fed2c8ed9f7f0bdf6026fcd1348453
Contract deployer address	0x24312a0b911fE2199fbea92efab55e2ECCeC637D

Project Website

https://moonbeans.io/

Codebase

https://moonriver.moonscan.io/address/0xc2392dd3e3fed2c8ed9f7f0bdf6026fcd1348453#code

SUMMARY

MoonBeans is one of the leading NFT marketplaces in the Moonriver and Moonbeam network, the first EVM-compatible para chains built on Kusama and Polkadot.

Contract Summary

Documentation Quality

MoonBeans 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 MoonBeans 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 239, 260, 279, 280, 324, 358, 735, 738, 746, 748, 756, 903, 938, 992, 1108, 1358, 1358, 1434, 1434, 1487, 1662, 1824, 1824, 1961, 1961, 1961 and 1108.
- SWC-103 | Pragma statements can be allowed to float when a contract is intended on lines 78, 175, 194, 225, 358, 671, 717, 772, 811, 849, 877, 1078, 1113, 1186, 1208 and 1340.
- 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 1090, 1110, 1112, 1487, 1734, 1734, 1749, 1751, 1754 and 1961.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 1607 and 1682.

CONCLUSION

We have audited the MoonBeans project released on September 2023 to discover issues and identify potential security vulnerabilities in MoonBeans 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 MoonBeans 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, tx.origin as a part of authorization control and out-of-bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value. 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. Use of "tx.origin" as a part of authorization control. Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

AUDIT RESULT

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

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	ISSUE FOUND
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
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	Tuesday Sep 07 2021 08:06:04 GMT+0000 (Coordinated Universal Time)
Finished	Wednesday Sep 08 2021 13:58:54 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	MOONBEANS.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	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged

SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-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-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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 239

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
238   /**
239   * @dev Returns the integer division of two unsigned integers. Reverts on
240   * division by zero. The result is rounded towards zero.
241   *
242   * Counterpart to Solidity's `/` operator. Note: this function uses a
243
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 260

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
259   * `revert` opcode (which leaves remaining gas untouched) while Solidity
260   * uses an invalid opcode to revert (consuming all remaining gas).
261   *
262   * Requirements:
263   *
264
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 279

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
278   * Counterpart to Solidity's `%` operator. This function uses a `revert`
279   * opcode (which leaves remaining gas untouched) while Solidity uses an
280   * invalid opcode to revert (consuming all remaining gas).
281   *
282   * Requirements:
283
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 280

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
279   * opcode (which leaves remaining gas untouched) while Solidity uses an
280   * invalid opcode to revert (consuming all remaining gas).
281   *
282   * Requirements:
283   *
284
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 324

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
323   * TIP: For a detailed writeup see our guide
324   * https://forum.zeppelin.solutions/t/how-to-implement-erc20-supply-
mechanisms/226[How
325   * to implement supply mechanisms].
326   *
327   * We have followed general OpenZeppelin guidelines: functions revert instead
328
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 358

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
357   *
358   * All two of these values are immutable: they can only be set once during
359   * construction.
360   */
361   constructor(string memory name_, string memory symbol_) public {
362
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 735

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
734   interface DividendPayingTokenInterface {
735   /// @notice View the amount of dividend in wei that an address can withdraw.
736   /// @param _owner The address of a token holder.
737   /// @return The amount of dividend in wei that `_owner` can withdraw.
738   function dividendOf(address _owner) external view returns(uint256);
739
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 738

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
737   /// @return The amount of dividend in wei that `_owner` can withdraw.
738   function dividendOf(address _owner) external view returns(uint256);
739
740
741   /// @notice Withdraws the ether distributed to the sender.
742
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 746

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
745
746   /// @dev This event MUST emit when ether is distributed to token holders.
747   /// @param from The address which sends ether to this contract.
748   /// @param weiAmount The amount of distributed ether in wei.
749   event DividendsDistributed(
750
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 748

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
747   /// @param from The address which sends ether to this contract.
748   /// @param weiAmount The amount of distributed ether in wei.
749   event DividendsDistributed(
750   address indexed from,
751   uint256 weiAmount
752
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 756

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
755   /// @param to The address which withdraws ether from this contract.
756   /// @param weiAmount The amount of withdrawn ether in wei.
757   event DividendWithdrawn(
758   address indexed to,
759   uint256 weiAmount
760
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 903

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
902   /// @notice Withdraws the ether distributed to the sender.
903   /// @dev It emits a `DividendWithdrawn` event if the amount of withdrawn ether is
greater than 0.
904   function withdrawDividend() public virtual override {
905   _withdrawDividendOfUser(msg.sender);
906   }
907
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 938

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
937   /// @param _owner The address of a token holder.
938   /// @return The amount of dividend in wei that `_owner` can withdraw.
939   function withdrawableDividendOf(address _owner) public view override
returns(uint256) {
940   return accumulativeDividendOf(_owner).sub(withdrawnDividends[_owner]);
941   }
942
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 992

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
991
992   magnifiedDividendCorrections[account] = magnifiedDividendCorrections[account]
993   .add( (magnifiedDividendPerShare.mul(value)).toInt256Safe() );
994   }
995
996
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1108

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1107   );
1108   event Sync(uint112 reserve0, uint112 reserve1);
1109
1110   function MINIMUM_LIQUIDITY() external pure returns (uint);
1111   function factory() external view returns (address);
1112
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1358

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1357   uint256 claims,
1358   uint256 lastProcessedIndex,
1359   bool indexed automatic,
1360   uint256 gas,
1361   address indexed processor
1362
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1358

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1357   uint256 claims,
1358   uint256 lastProcessedIndex,
1359   bool indexed automatic,
1360   uint256 gas,
1361   address indexed processor
1362
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1434

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1433
1434   function excludeMultipleAccountsFromFees(address[] calldata accounts, bool
excluded) public onlyOwner {
1435   for(uint256 i = 0; i < accounts.length; i++) {
1436   _isExcludedFromFees[accounts[i]] = excluded;
1437   }
1438
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1434

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1433
1434   function excludeMultipleAccountsFromFees(address[] calldata accounts, bool
excluded) public onlyOwner {
1435   for(uint256 i = 0; i < accounts.length; i++) {
1436   _isExcludedFromFees[accounts[i]] = excluded;
1437   }
1438
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1487

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1486   function updateGasForProcessing(uint256 newValue) public onlyOwner {
1487   require(newValue >= 10000 && newValue <= 1000000, "MOONBEANS: gasForProcessing
must be between 10,000 and 1,000,000");
1488   require(newValue != gasForProcessing, "MOONBEANS: Cannot update gasForProcessing
to same value");
1489   emit GasForProcessingUpdated(newValue, gasForProcessing);
1490   gasForProcessing = newValue;
1491
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1662

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1661   // how much ETH did we just swap into?
1662   uint256 newBalance = address(this).balance.sub(initialBalance);
1663
1664   // add liquidity to uniswap
1665   addLiquidity(otherHalf, newBalance);
1666
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1824

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1823   uint256 secondsUntilAutoClaimAvailable) {
1824   account = _account;
1825
1826   index = tokenHoldersMap.getIndexOfKey(account);
1827
1828
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1824

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1823   uint256 secondsUntilAutoClaimAvailable) {
1824   account = _account;
1825
1826   index = tokenHoldersMap.getIndexOfKey(account);
1827
1828
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1961

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1960   }
1961   }
1962
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1961

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1960   }
1961   }
1962
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1961

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1960   }
1961   }
1962
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 1108

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1107   );
1108   event Sync(uint112 reserve0, uint112 reserve1);
1109
1110   function MINIMUM_LIQUIDITY() external pure returns (uint);
1111   function factory() external view returns (address);
1112
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 78

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
77   * Emits an {Approval} event.
78   */
79   function approve(address spender, uint256 amount) external returns (bool);
80
81   /**
82
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 175

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
174   * - Addition cannot overflow.
175   */
176   function add(uint256 a, uint256 b) internal pure returns (uint256) {
177   uint256 c = a + b;
178   require(c >= a, "SafeMath: addition overflow");
179
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 194

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
193   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
194   return sub(a, b, "SafeMath: subtraction overflow");
195   }
196
197   /**
198
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 225

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
224   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
225   // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
226   // benefit is lost if 'b' is also tested.
227   // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
228   if (a == 0) {
229
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 358

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
357   *
358   * All two of these values are immutable: they can only be set once during
359   * construction.
360   */
361   constructor(string memory name_, string memory symbol_) public {
362
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 671

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
670   /**
671   * @dev Multiplies two int256 variables and fails on overflow.
672   */
673   function mul(int256 a, int256 b) internal pure returns (int256) {
674   int256 c = a * b;
675
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 717

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
716   return a < 0 ? -a : a;
717   }
718
719
720   function toUint256Safe(int256 a) internal pure returns (uint256) {
721
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 772

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
771   interface DividendPayingTokenOptionalInterface {
772   /// @notice View the amount of dividend in wei that an address can withdraw.
773   /// @param _owner The address of a token holder.
774   /// @return The amount of dividend in wei that `_owner` can withdraw.
775   function withdrawableDividendOf(address _owner) external view returns(uint256);
776
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 811

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
810   * @dev Returns the address of the current owner.
811   */
812   function owner() public view returns (address) {
813   return _owner;
814   }
815
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 849

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
848
849   /// @title Dividend-Paying Token
850   /// @author Roger Wu (https://github.com/roger-wu)
851   /// @dev A mintable ERC20 token that allows anyone to pay and distribute ether
852   ///  to token holders as dividends and allows token holders to withdraw their
dividends.
853
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 877

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
876   //   where `dividendCorrectionOf(_user)` is updated whenever `balanceOf(_user)` is
changed:
877   //   `dividendCorrectionOf(_user) = dividendPerShare * (old balanceOf(_user)) -
(new balanceOf(_user))`.
878   // So now `dividendOf(_user)` returns the same value before and after
`balanceOf(_user)` is changed.
879   mapping(address => int256) internal magnifiedDividendCorrections;
880   mapping(address => uint256) internal withdrawnDividends;
881
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1078

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
1077   interface ISolarPair {
1078   event Approval(address indexed owner, address indexed spender, uint value);
1079   event Transfer(address indexed from, address indexed to, uint value);
1080
1081   function name() external pure returns (string memory);
1082
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1113

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
1112   function token0() external view returns (address);
1113   function token1() external view returns (address);
1114   function getReserves() external view returns (uint112 reserve0, uint112 reserve1,
uint32 blockTimestampLast);
1115   function price0CumulativeLast() external view returns (uint);
1116   function price1CumulativeLast() external view returns (uint);
1117
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1186

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
1185   uint liquidity,
1186   uint amountTokenMin,
1187   uint amountETHMin,
1188   address to,
1189   uint deadline
1190
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1208

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
1207   uint deadline,
1208   bool approveMax, uint8 v, bytes32 r, bytes32 s
1209   ) external returns (uint amountToken, uint amountETH);
1210   function swapExactTokensForTokens(
1211   uint amountIn,
1212
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 1340

low SEVERITY

The current pragma Solidity directive is ""^0.6.2"". 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

- MOONBEANS.sol

Locations

```
1339
1340   event LiquidityWalletUpdated(address indexed newLiquidityWallet, address indexed
oldLiquidityWallet);
1341
1342   event GasForProcessingUpdated(uint256 indexed newValue, uint256 indexed oldValue);
1343
1344
```

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

LINE 1607

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- MOONBEANS.sol

Locations

```
1606   // if any account belongs to _isExcludedFromFee account then remove the fee
1607   if(_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
1608   takeFee = false;
1609   }
1610
1611
```

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

LINE 1682

low SEVERITY

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Source File

- MOONBEANS.sol

Locations

```
1681   // make the swap
1682   uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
1683   tokenAmount,
1684   0, // accept any amount of ETH
1685   path,
1686
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1090

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1089   function transfer(address to, uint value) external returns (bool);
1090   function transferFrom(address from, address to, uint value) external returns
(bool);
1091
1092   function DOMAIN_SEPARATOR() external view returns (bytes32);
1093   function PERMIT_TYPEHASH() external pure returns (bytes32);
1094
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1110

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1109
1110   function MINIMUM_LIQUIDITY() external pure returns (uint);
1111   function factory() external view returns (address);
1112   function token0() external view returns (address);
1113   function token1() external view returns (address);
1114
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1112

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1111   function factory() external view returns (address);
1112   function token0() external view returns (address);
1113   function token1() external view returns (address);
1114   function getReserves() external view returns (uint112 reserve0, uint112 reserve1,
uint32 blockTimestampLast);
1115   function price0CumulativeLast() external view returns (uint);
1116
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1487

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1486   function updateGasForProcessing(uint256 newValue) public onlyOwner {
1487   require(newValue >= 10000 && newValue <= 1000000, "MOONBEANS: gasForProcessing
must be between 10,000 and 1,000,000");
1488   require(newValue != gasForProcessing, "MOONBEANS: Cannot update gasForProcessing
to same value");
1489   emit GasForProcessingUpdated(newValue, gasForProcessing);
1490   gasForProcessing = newValue;
1491
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1734

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1733   if (success) {
1734   dividendTracker.distributeCAKEDividends(dividends);
1735   emit SendDividends(tokens, dividends);
1736   }
1737   }
1738
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1734

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1733   if (success) {
1734   dividendTracker.distributeCAKEDividends(dividends);
1735   emit SendDividends(tokens, dividends);
1736   }
1737   }
1738
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1749

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1748
1749   function setRewardToken(address newToken) public onlyOwner {
1750   CAKE = newToken;
1751   }
1752   }
1753
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1751

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1750   CAKE = newToken;
1751   }
1752   }
1753
1754   contract BEANSDividendTracker is Ownable, DividendPayingToken {
1755
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1754

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1753
1754   contract BEANSDividendTracker is Ownable, DividendPayingToken {
1755   using SafeMath for uint256;
1756   using SafeMathInt for int256;
1757   using IterableMapping for IterableMapping.Map;
1758
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1961

low SEVERITY

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

Source File

- MOONBEANS.sol

Locations

```
1960   }
1961   }
1962
```

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.