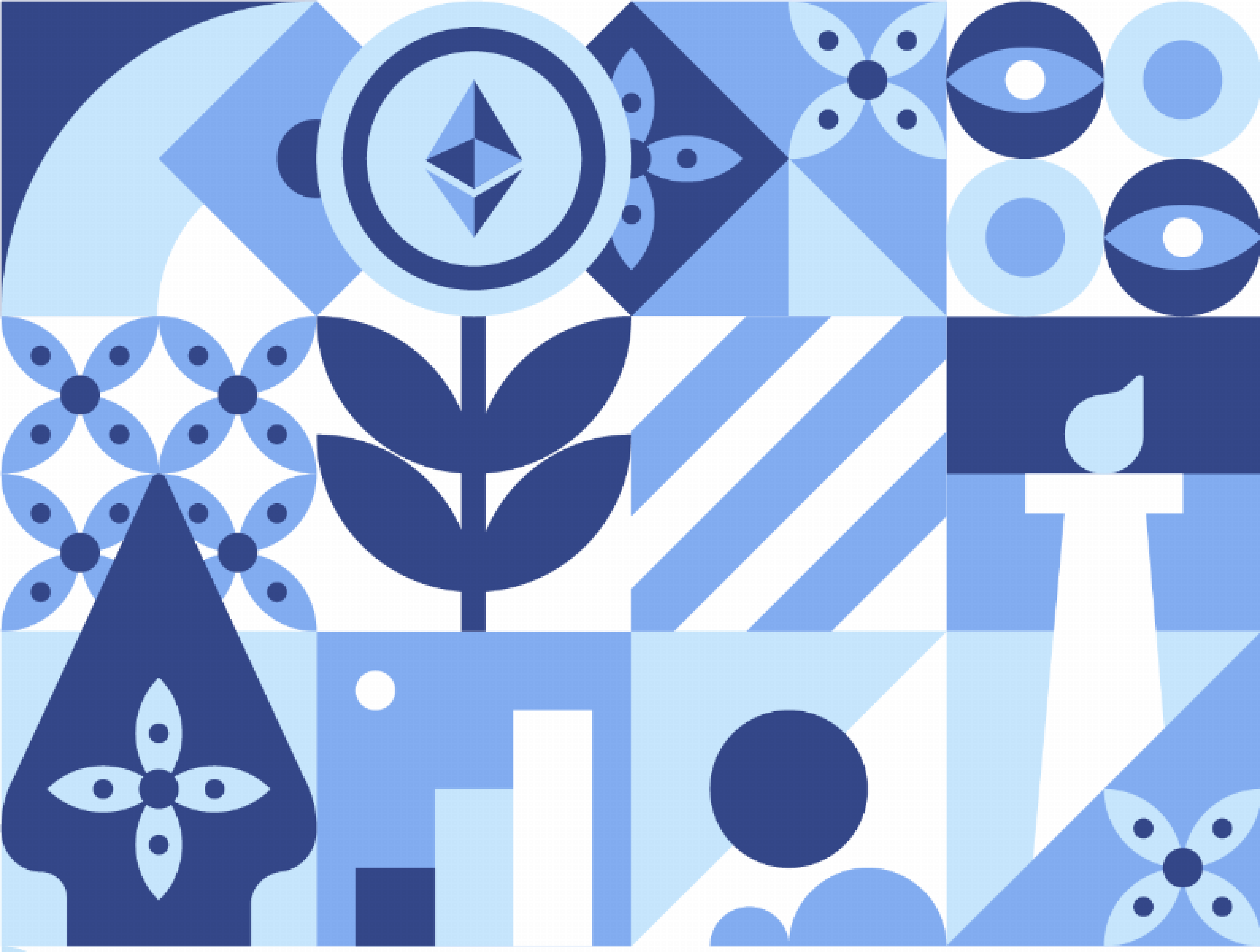

Rising Sun

Smart Contract Audit Report

SYSFIXED

24 Mar 2022

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Rising Sun	SUN	Ethereum

Addresses

Contract address	0x50522c769e01eb06c02bd299066509d8f97a69ae
Contract deployer address	0xA46b0841396C539C7a92905d4984856E1D177A40

Project Website

https://risingsuncoin.io/

Codebase

https://etherscan.io/address/0x50522c769e01eb06c02bd299066509d8f97a69ae#code

SUMMARY

The Rising Sun project is focused on building generational wealth and providing passive income for our investors through ETH reflections, NFT staking and leveraging utilities we can bring to our platform creating an ecosystem of wealth.

Contract Summary

Documentation Quality

Rising Sun 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 Rising Sun 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 1724.
- SWC-101 | It is recommended to use vetted safe math libraries for arithmetic operations consistently on lines 178, 192, 206, 220, 242, 256, 271, 272, 285, 297, 312, 326, 340, 354, 370, 393, 416, 442, 953, 972, 994, 1027, 1029, 1050, 1051, 1076, 1078, 1267, 1353, 1400, 1463, 1536, 1536, 1680, 1690, 1694, 1763, 1781, 1781, 1782, 1782, 1902, 1902, 1933, 2081, 2081, 2085, 2085, 2086, 2088, 2114, 2124, 2140 and 1267.
- 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 1238, 1268, 1273, 1686, 1764, 1934, 2151 and 2152.
- SWC-115 | tx.origin should not be used for authorization, use msg.sender instead on lines 2024 and 2101.

CONCLUSION

We have audited the Rising Sun project released on march 2023 to discover issues and identify potential security vulnerabilities in Rising Sun Project. This process is used to find technical issues and security loopholes which might be found in the smart contract.

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

The issues found in the Rising Sun 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, tx.origin as a part of authorization control, tx.origin should not be used for authorization, use msg.sender instead. Out-of-bounds array access which the index access expression can cause an exception in case of the use of an invalid array index value.

AUDIT RESULT

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

DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	ISSUE FOUND
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique ID	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Incorrect Constructor Name	SWC-118	Constructors are special functions that are called only once during the contract creation.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	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	Wednesday Mar 23 2022 15:50:11 GMT+0000 (Coordinated Universal Time)
Finished	Thursday Mar 24 2022 11:24:24 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	SUN.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-=" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged

SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-115	USE OF "TX.ORIGIN" AS A PART OF AUTHORIZATION CONTROL.	low	acknowledged
SWC-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-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 178

low SEVERITY

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

Source File

- SUN.sol

Locations

```
177   function mul(int256 a, int256 b) internal pure returns (int256) {
178   return a * b;
179   }
180
181   /**
182
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 192

low SEVERITY

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

Source File

- SUN.sol

Locations

```
191   function div(int256 a, int256 b) internal pure returns (int256) {
192   return a / b;
193   }
194
195   /**
196
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 206

low SEVERITY

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

Source File

- SUN.sol

Locations

```
205   function sub(int256 a, int256 b) internal pure returns (int256) {
206   return a - b;
207   }
208
209   /**
210
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 220

low SEVERITY

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

Source File

- SUN.sol

Locations

```
219   function add(int256 a, int256 b) internal pure returns (int256) {
220   return a + b;
221   }
222   }
223
224
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 242

low SEVERITY

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

Source File

- SUN.sol

Locations

```
241   unchecked {
242   uint256 c = a + b;
243   if (c < a) return (false, 0);
244   return (true, c);
245   }
246
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 256

low SEVERITY

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

Source File

- SUN.sol

Locations

```
255   if (b > a) return (false, 0);
256   return (true, a - b);
257   }
258   }
259
260
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 271

low SEVERITY

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

Source File

- SUN.sol

Locations

```
270   if (a == 0) return (true, 0);
271   uint256 c = a * b;
272   if (c / a != b) return (false, 0);
273   return (true, c);
274   }
275
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 272

low SEVERITY

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

Source File

- SUN.sol

Locations

```
271   uint256 c = a * b;
272   if (c / a != b) return (false, 0);
273   return (true, c);
274   }
275   }
276
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 285

low SEVERITY

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

Source File

- SUN.sol

Locations

```
284   if (b == 0) return (false, 0);
285   return (true, a / b);
286   }
287   }
288
289
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 297

low SEVERITY

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

Source File

- SUN.sol

Locations

```
296   if (b == 0) return (false, 0);
297   return (true, a % b);
298   }
299   }
300
301
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 312

low SEVERITY

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

Source File

- SUN.sol

Locations

```
311   function add(uint256 a, uint256 b) internal pure returns (uint256) {
312   return a + b;
313   }
314
315   /**
316
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 326

low SEVERITY

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

Source File

- SUN.sol

Locations

```
325   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
326   return a - b;
327   }
328
329   /**
330
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 340

low SEVERITY

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

Source File

- SUN.sol

Locations

```
339   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
340   return a * b;
341   }
342
343   /**
344
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 354

low SEVERITY

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

Source File

- SUN.sol

Locations

```
353   function div(uint256 a, uint256 b) internal pure returns (uint256) {
354   return a / b;
355   }
356
357   /**
358
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 370

low SEVERITY

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

Source File

- SUN.sol

Locations

```
369   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
370   return a % b;
371   }
372
373   /**
374
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 393

low SEVERITY

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

Source File

- SUN.sol

Locations

```
392   require(b <= a, errorMessage);
393   return a - b;
394   }
395   }
396
397
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 416

low SEVERITY

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

Source File

- SUN.sol

Locations

```
415   require(b > 0, errorMessage);
416   return a / b;
417   }
418   }
419
420
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 442

low SEVERITY

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

Source File

- SUN.sol

Locations

```
441   require(b > 0, errorMessage);
442   return a % b;
443   }
444   }
445   }
446
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 953

low SEVERITY

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

Source File

- SUN.sol

Locations

```
952   unchecked {
953   _approve(sender, _msgSender(), currentAllowance - amount);
954   }
955
956   return true;
957
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 972

low SEVERITY

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

Source File

- SUN.sol

Locations

```
971   function increaseAllowance(address spender, uint256 addedValue) public virtual
returns (bool) {
972   _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
973   return true;
974   }
975
976
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 994

low SEVERITY

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

Source File

- SUN.sol

Locations

```
993   unchecked {
994   _approve(_msgSender(), spender, currentAllowance - subtractedValue);
995   }
996
997   return true;
998
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1027

low SEVERITY

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

Source File

- SUN.sol

Locations

```
unchecked {
_balances[sender] = senderBalance - amount;
}
_balances[recipient] += amount;
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1029

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1028   }
1029   _balances[recipient] += amount;
1030
1031   emit Transfer(sender, recipient, amount);
1032
1033
```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1050

low SEVERITY

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

Source File

- SUN.sol

Locations

```

    _totalSupply += amount;
    _balances[account] += amount;
    emit Transfer(address(0), account, amount);


```

SWC-101 | ARITHMETIC OPERATION "+=" DISCOVERED

LINE 1051

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1050   _totalSupply += amount;
1051   _balances[account] += amount;
1052   emit Transfer(address(0), account, amount);
1053
1054   _afterTokenTransfer(address(0), account, amount);
1055
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1076

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1075   unchecked {
1076   _balances[account] = accountBalance - amount;
1077   }
1078   _totalSupply -= amount;
1079
1080
```

SWC-101 | ARITHMETIC OPERATION "-=" DISCOVERED

LINE 1078

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1077   }
1078   _totalSupply -= amount;
1079
1080   emit Transfer(account, address(0), amount);
1081
1082
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1267

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1266   uint index = map.indexOf[key];
1267   uint lastIndex = map.keys.length - 1;
1268   address lastKey = map.keys[lastIndex];
1269
1270   map.indexOf[lastKey] = index;
1271
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1353

low SEVERITY

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

Source File

- SUN.sol

Locations

```
//  see https://github.com/ethereum/EIPs/issues/1726#issuecomment-472352728
uint256 constant internal magnitude = 2**128;

uint256 internal magnifiedDividendPerShare;


```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1400

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1399   magnifiedDividendPerShare = magnifiedDividendPerShare.add(
1400   (msg.value).mul(magnitude) / totalSupply()
1401   );
1402   emit DividendsDistributed(msg.sender, msg.value);
1403
1404
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 1463

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1462   function accumulativeDividendOf(address _owner) public view override
returns(uint256) {
1463   return magnifiedDividendPerShare.mul(balanceOf(_owner)).toInt256()
1464   .add(magnifiedDividendCorrections[_owner]).toUint256() / magnitude;
1465   }
1466
1467
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1536

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1535   claimWait = 3600;
1536   minimumTokenBalanceForDividends = 10000 * (10**5); //must hold 10000+ tokens
1537   }
1538
1539   function _transfer(address, address, uint256) internal pure override {
1540
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1536

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1535   claimWait = 3600;
1536   minimumTokenBalanceForDividends = 10000 * (10**5); //must hold 10000+ tokens
1537   }
1538
1539   function _transfer(address, address, uint256) internal pure override {
1540
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1680

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1679   while(gasUsed < gas && iterations < numberOfTokenHolders) {
1680   _lastProcessedIndex++;
1681
1682   if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {
1683   _lastProcessedIndex = 0;
1684
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1690

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1689   if(processAccount(payable(account), true)) {
1690   claims++;
1691   }
1692   }
1693
1694
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1694

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1693
1694    iterations++;
1695
1696    uint256 newGasLeft = gasleft();
1697
1698
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1763

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1762   function includeToWhiteList(address[] memory _users) external onlyOwner {
1763   for(uint8 i = 0; i < _users.length; i++) {
1764   _whiteList[_users[i]] = true;
1765   }
1766   }
1767
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1781

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1780
1781   uint256 public maxSellTransactionAmount = 1000000000000000000 * (10**5);
1782   uint256 public swapTokensAtAmount = 200000000000000 * (10**5);
1783
1784   uint256 public ETHRewardFee;
1785
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1781

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1780
1781   uint256 public maxSellTransactionAmount = 1000000000000000000 * (10**5);
1782   uint256 public swapTokensAtAmount = 200000000000000 * (10**5);
1783
1784   uint256 public ETHRewardFee;
1785
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1782

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1781   uint256 public maxSellTransactionAmount = 1000000000000000000 * (10**5);
1782   uint256 public swapTokensAtAmount = 200000000000000 * (10**5);
1783
1784   uint256 public ETHRewardFee;
1785   uint256 public liquidityFee;
1786
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1782

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1781   uint256 public maxSellTransactionAmount = 1000000000000000000 * (10**5);
1782   uint256 public swapTokensAtAmount = 200000000000000 * (10**5);
1783
1784   uint256 public ETHRewardFee;
1785   uint256 public liquidityFee;
1786
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 1902

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1901   */
1902   _mint(owner(), 100000000000000000000 * (10**5));
1903   }
1904
1905   receive() external payable {
1906
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 1902

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1901    */
1902    _mint(owner(), 1000000000000000000000 * (10**5));
1903    }
1904
1905    receive() external payable {
1906
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 1933

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1932   function excludeMultipleAccountsFromFees(address[] calldata accounts, bool
excluded) public onlyOwner {
1933   for(uint256 i = 0; i < accounts.length; i++) {
1934   _isExcludedFromFees[accounts[i]] = excluded;
1935   }
1936
1937
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 2081

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2080   if(!_isExcludedFromFees[from] && !_isExcludedFromFees[to]) {
2081   uint256 fees = (amount*totalFees)/100;
2082   uint256 extraFee;
2083
2084   if(automatedMarketMakerPairs[to]) {
2085
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 2081

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2080   if(!_isExcludedFromFees[from] && !_isExcludedFromFees[to]) {
2081   uint256 fees = (amount*totalFees)/100;
2082   uint256 extraFee;
2083
2084   if(automatedMarketMakerPairs[to]) {
2085
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 2085

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2084   if(automatedMarketMakerPairs[to]) {
2085   extraFee =(amount*extraFeeOnSell)/100;
2086   fees=fees+extraFee;
2087   }
2088   amount = amount-fees;
2089
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 2085

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2084   if(automatedMarketMakerPairs[to]) {
2085   extraFee =(amount*extraFeeOnSell)/100;
2086   fees=fees+extraFee;
2087   }
2088   amount = amount-fees;
2089
```

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2086

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2085   extraFee =(amount*extraFeeOnSell)/100;
2086   fees=fees+extraFee;
2087   }
2088   amount = amount-fees;
2089   super._transfer(from, address(this), fees); // get total fee first
2090
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2088

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2087   }
2088   amount = amount-fees;
2089   super._transfer(from, address(this), fees); // get total fee first
2090   }
2091
2092
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2114

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2113   // swap the remaining to BNB
2114   uint256 toSwap = contractTokenBalance-tokensToAddLiquidityWith;
2115   // capture the contract's current ETH balance.
2116   // this is so that we can capture exactly the amount of ETH that the
2117   // swap creates, and not make the liquidity event include any ETH that
2118
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2124

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2123
2124   uint256 deltaBalance = address(this).balance-initialBalance;
2125
2126   // take worthy amount bnb to add liquidity
2127   // worthyBNB = deltaBalance * liquidity/(2totalFees - liquidityFee)
2128
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2140

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2139   if(success) {
2140   emit SendDividends(toSwap-tokensToAddLiquidityWith, dividends);
2141   }
2142
2143   emit SwapAndLiquify(tokensToAddLiquidityWith, deltaBalance);
2144
```

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

LINE 1267

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1266   uint index = map.indexOf[key];
1267   uint lastIndex = map.keys.length - 1;
1268   address lastKey = map.keys[lastIndex];
1269
1270   map.indexOf[lastKey] = index;
1271
```

SWC-103 | A FLOATING PRAGMA IS SET.

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- SUN.sol

Locations

```
6
7   pragma solidity ^0.8.0;
8
9   interface IUniswapV2Router01 {
10   function factory() external pure returns (address);
11
```

SWC-108 | STATE VARIABLE VISIBILITY IS NOT SET.

LINE 1724

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1723   contract SafeToken is Ownable {
1724   address payable safeManager;
1725
1726   constructor() {
1727   safeManager = payable(msg.sender);
1728
```

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

LINE 2024

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

- SUN.sol

Locations

```
2023   (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) =
dividendTracker.process(gas);
2024   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas,
tx.origin);
2025   }
2026
2027   function claim() external {
2028
```

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

LINE 2101

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

- SUN.sol

Locations

```
2100   try dividendTracker.process(gas) returns (uint256 iterations, uint256 claims,
uint256 lastProcessedIndex) {
2101   emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas,
tx.origin);
2102   }
2103   catch {
2104
2105
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1238

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1237   function getKeyAtIndex(Map storage map, uint index) public view returns (address)
{
1238   return map.keys[index];
1239   }
1240
1241
1242
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1268

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1267   uint lastIndex = map.keys.length - 1;
1268   address lastKey = map.keys[lastIndex];
1269
1270   map.indexOf[lastKey] = index;
1271   delete map.indexOf[key];
1272
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1273

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1272
1273    map.keys[index] = lastKey;
1274    map.keys.pop();
1275    }
1276    }
1277
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1686

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1685
1686   address account = tokenHoldersMap.keys[_lastProcessedIndex];
1687
1688   if(canAutoClaim(lastClaimTimes[account])) {
1689   if(processAccount(payable(account), true)) {
1690
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1764

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1763   for(uint8 i = 0; i < _users.length; i++) {
1764   _whiteList[_users[i]] = true;
1765   }
1766   }
1767   }
1768
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1934

low SEVERITY

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

Source File

- SUN.sol

Locations

```
1933   for(uint256 i = 0; i < accounts.length; i++) {
1934   _isExcludedFromFees[accounts[i]] = excluded;
1935   }
1936
1937   emit ExcludeMultipleAccountsFromFees(accounts, excluded);
1938
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2151

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2150   address[] memory path = new address[](2);
2151   path[0] = address(this);
2152   path[1] = uniswapV2Router.WETH();
2153
2154   if(allowance(address(this), address(uniswapV2Router)) < tokenAmount) {
2155
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2152

low SEVERITY

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

Source File

- SUN.sol

Locations

```
2151   path[0] = address(this);
2152   path[1] = uniswapV2Router.WETH();
2153
2154   if(allowance(address(this), address(uniswapV2Router)) < tokenAmount) {
2155   _approve(address(this), address(uniswapV2Router), ~uint256(0));
2156
```

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.