



Zodiac

# Smart Contract Audit Report

# TABLE OF CONTENTS

## Audited Details

- Audited Project
- Blockchain
- Addresses
- Project Website
- Codebase

## Summary

- Contract Summary
- Audit Findings Summary
- Vulnerabilities Summary

## Conclusion

## Audit Results

## Smart Contract Analysis

- Detected Vulnerabilities

## Disclaimer

## About Us

# AUDITED DETAILS

## Audited Project

Project name	Token ticker	Blockchain
ZODIAC	ZODIAC	BSC

## Addresses

Contract address	0x27dFE1c82EE68Ba2AFBe83126530cd32D437af7d
Contract deployer address	0xC4AE78492D08cf60C1b96DfF9C04a3A7aB1F6384

## Project Website

<https://zodiacoin.io/>

## Codebase

<https://bscscan.com/address/0x27dFE1c82EE68Ba2AFBe83126530cd32D437af7d#code>

# SUMMARY

Zodiac has a unique contract that works in a way whereby every time an investor buys/sells, they instantly earn extra tokens and also provide ZODIAC rewards to holders of \$Zodiac | Renounced Ownership | Zodiac Grow blockchain | WALLET | GAMES | BRIDGE | MARKET PLACE | 8% Tax | 999-day liquidity lock | Protoken Pinksale Contract | CMC&CG fast track. Unlocked tokens go to the next token "Aries" in 15 days. View our website /TG to learn more about the contract and utilities!

## Contract Summary

### Documentation Quality

This project has a standard of documentation.

- Technical description provided.

### Code Quality

The quality of the code in this project is up to standard.

- The official Solidity style guide is followed.

### Test Scope

Project test coverage is 100% ( Via Codebase ).

## Audit Findings Summary

- SWC-101 | Arithmetic operation issues discovered on lines 116, 152, 175, 176, 215, 255, 282, 286, 298, 305, 314, 405, 1056, 1228, 1247, 1248, 1252, 1550, 1641, 1662, 1677, 1686, 1783, 1802, 1850, 1876, 1882, 1895, 1903, 2240, 2250, 2253, 2321, 2400, 2402, 2443, 2469, 2476, 2555, 2560, and 2572.
- SWC-101 | Compiler-rewritable issue discovered on lines 405 and 2402.
- SWC-103 | A floating pragma is set. Issue discovered on line 7, the current pragma Solidity directive is `^0.8.15`.
- SWC-108 | State variable visibility is not set. Issues were discovered on lines 1039, 1147, 1148, 1149, 1151, 1152, 1153, and 1155.
- SWC-110 | Out of bounds array access discovered on lines 374, 406, 411, 1254, 1255, 1643, 1644, 1646, 1647, 1953, 1954, 1971, 1972, 1993, 1994, 1995, 2246, 2401, and 2042.
- SWC-115 | Use of "tx.origin" as a part of authorization control. Issues were discovered on lines 1824 and 2429.

## CONCLUSION

We have audited the Zodiac project which has released on January 2023, to discover issues and identify potential security vulnerabilities in Zodiac Project. This process is used to find technical issues and security loopholes that find common issues in the code.

The security audit report produced satisfactory results with low-risk issues.

The most common issue found in writing code on contracts that do not pose a big risk is that writing on contracts is close to the standard of writing contracts in general. The low-level issue found is a floating pragma is set, state variable visibility is not set on some lines, the use of "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 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.	<b>ISSUE FOUND</b>
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	<b>ISSUE FOUND</b>
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	<b>PASS</b>
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	<b>ISSUE FOUND</b>
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	<b>PASS</b>
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	<b>PASS</b>
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	<b>PASS</b>
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	<b>ISSUE FOUND</b>
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	<b>PASS</b>
Delegate call to Untrusted Caller	SWC-112	Delegatecalls should only be allowed to trusted addresses.	<b>PASS</b>
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	<b>PASS</b>
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	<b>PASS</b>

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	<b>ISSUE FOUND</b>
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	<b>PASS</b>
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.	<b>PASS</b>
Shadowing State Variable	SWC-119	State variables should not be shadowed.	<b>PASS</b>
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	<b>PASS</b>
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/.	<b>PASS</b>

# SMART CONTRACT ANALYSIS

Started	SAT Jan 14 2023 23:14:51 GMT+0000 (Coordinated Universal Time)
Finished	Sun Jan 15 2023 00:14:51 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Zodiac.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	COMPILER-REWRITABLE "<UINT> - 1" 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-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT SET.	low	acknowledged
SWC-108	STATE VARIABLE VISIBILITY IS NOT 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-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 116

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
115 function add(uint256 a, uint256 b) internal pure returns (uint256) {  
116     uint256 c = a + b;  
117     require(c >= a, "SafeMath: addition overflow");  
118  
119     return c;  
}
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 152

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
151   require(b <= a, errorMessage);
152   uint256 c = a - b;
153
154   return c;
155 }
```

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

LINE 175

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
174
175  uint256 c = a * b;
176  require(c / a == b, "SafeMath: multiplication overflow");
177
178  return c;
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 176

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
175  uint256 c = a * b;  
176  require(c / a == b, "SafeMath: multiplication overflow");  
177  
178  return c;  
179  }
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 215

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
214   require(b > 0, errorMessage);
215   uint256 c = a / b;
216   // assert(a == b * c + a % b); // There is no case in which this doesn't hold
217
218   return c;
```



# SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 255

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
254   require(b != 0, errorMessage);
255   return a % b;
256   }
257   }
258
```

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

LINE 282

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
281 function mul(int256 a, int256 b) internal pure returns (int256) {
282     int256 c = a * b;
283
284     // Detect overflow when multiplying MIN_INT256 with -1
285     require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 286

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
285   require(c != MIN_INT256 || (a & MIN_INT256) != (b & MIN_INT256));
286   require((b == 0) || (c / b == a));
287   return c;
288   }
289
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 298

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
297 // Solidity already throws when dividing by 0.  
298 return a / b;  
299 }  
300  
301 /**
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 305

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
304 function sub(int256 a, int256 b) internal pure returns (int256) {
305     int256 c = a - b;
306     require((b >= 0 && c <= a) || (b < 0 && c > a));
307     return c;
308 }
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 314

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
313 function add(int256 a, int256 b) internal pure returns (int256) {  
314     int256 c = a + b;  
315     require((b >= 0 && c >= a) || (b < 0 && c < a));  
316     return c;  
317 }
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 405

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
404 uint256 index = map.indexOf[key];
405 uint256 lastIndex = map.keys.length - 1;
406 address lastKey = map.keys[lastIndex];
407
408 map.indexOf[lastKey] = index;
```

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

LINE 1056

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1055 uint256 public _tDividendTotal = 0;
1056 uint256 internal constant magnitude = 2**128;
1057 uint256 internal magnifiedDividendPerShare;
1058 mapping(address => int256) internal magnifiedDividendCorrections;
1059 mapping(address => uint256) internal withdrawnDividends;
```



# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 1228

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1227  _tTotal = amountOfTokenWei;  
1228  _rTotal = (MAX - (MAX % _tTotal));  
1229  
1230  _rOwned[_msgSender()] = _rTotal;  
1231
```

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

LINE 1247

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1246
1247  _maxTxAmount = _tTotal.mul(setMxTxPer).div(10**4);
1248  _maxWalletAmount = _tTotal.mul(setMxWalletPer).div(10**4);
1249
1250  swapAmount = amountOfTokenWei.mul(1).div(10000);
```

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

LINE 1248

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1247  _maxTxAmount = _tTotal.mul(setMxTxPer).div(10**4);
1248  _maxWalletAmount = _tTotal.mul(setMxWalletPer).div(10**4);
1249
1250  swapAmount = amountOfTokenWei.mul(1).div(10000);
1251
```

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

LINE 1252

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1251
1252     buyBackUpperLimit = 10**18;
1253
1254     router = _addrs[0];
1255     payable(_addrs[1]).transfer(msg.value);
```

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

LINE 1550

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1549     require(  
1550     amount >= (10**decimals()) && amount <= totalSupply().div(100),  
1551     "not valid amount"  
1552     );  
1553     swapAmount = amount;
```

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

LINE 1641

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1640 uint256 tSupply = _tTotal;
1641 for (uint256 i = 0; i < _excluded.length; i++) {
1642     if (
1643         _rOwned[_excluded[i]] > rSupply ||
1644         _tOwned[_excluded[i]] > tSupply
```

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

LINE 1662

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1661 function calculateTaxFee(uint256 _amount) private view returns (uint256) {
1662     return _amount.mul(_taxFee).div(10**2);
1663 }
1664
1665 function calculateLiquidityFee(uint256 _amount)
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1677

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1676     .mul(  
1677     _liquidityFee +  
1678     _burnFee +  
1679     _walletFee +  
1680     _buybackFee +
```



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

LINE 1686

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1685 )  
1686 .div(10**2);  
1687 }  
1688  
1689 function removeAllFee() private {
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1783

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1782     require(  
1783     contractBalanceReceipient + amount <= _maxWalletAmount,  
1784     "Exceeds maximum wallet amount"  
1785     );  
1786 }
```

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

LINE 1802

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1801 uint256 balance = address(this).balance;
1802 if (balance > uint256(1 * 10**uint256(_decimals))) {
1803     if (balance > buyBackUpperLimit)
1804         balance = buyBackUpperLimit;
1805 }
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1850

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1849 function swapAndLiquify(uint256 contractTokenBalance) private lockTheSwap {  
1850     uint8 totFee = _burnFee +  
1851     _walletFee +  
1852     _liquidityFee +  
1853     _buybackFee +
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1876

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1875
1876     totSpentAmount = totSpentAmount + spentAmount;
1877 }
1878
1879     if (_buybackFee != 0) {
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1882

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1881 swapTokensForBNB(spentAmount);
1882 totSpentAmount = totSpentAmount + spentAmount;
1883 }
1884
1885 if (_walletCharityFee != 0) {
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1895

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1894
1895     totSpentAmount = totSpentAmount + spentAmount;
1896 }
1897
1898     if (_walletDevFee != 0) {
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 1903

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
1902
1903     totSpentAmount = totSpentAmount + spentAmount;
1904     }
1905
1906     if (_rewardFee != 0) {
```



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

LINE 2240

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2239 while (gasUsed < gas && iterations < numberOfTokenHolders) {  
2240   _lastProcessedIndex++;  
2241  
2242   if (_lastProcessedIndex >= tokenHoldersMap.keys.length) {  
2243     _lastProcessedIndex = 0;
```

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

LINE 2250

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2249  if (processAccount(payable(account), true)) {  
2250  claims++;  
2251  }  
2252  }  
2253  iterations++;
```

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

LINE 2253

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2252     }
2253     iterations++;
2254     uint256 newGasLeft = gasleft();
2255     if (gasLeft > newGasLeft) {
2256         gasUsed = gasUsed.add(gasLeft.sub(newGasLeft));
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 2321

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2320     return
2321     magnifiedDividendPerShare
2322     .mul(balanceOf(_owner))
2323     .toInt256Safe()
2324     .add(magnifiedDividendCorrections[_owner])
```

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

LINE 2400

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2399   require(!_isExcluded[account], "Already excluded");
2400   for (uint256 i = 0; i < _excluded.length; i++) {
2401     if (_excluded[i] == account) {
2402       _excluded[i] = _excluded[_excluded.length - 1];
2403       _tOwned[account] = 0;
```

# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2402

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2401  if (_excluded[i] == account) {  
2402  _excluded[i] = _excluded[_excluded.length - 1];  
2403  _tOwned[account] = 0;  
2404  _isExcluded[account] = false;  
2405  _excluded.pop();
```

# SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 2443

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2442 magnifiedDividendPerShare = magnifiedDividendPerShare.add(  
2443 (amount).mul(magnitude) / _tDividendTotal  
2444 );  
2445 emit DividendsDistributed(amount);  
2446
```

# SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 2469

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2468 function _dmint(address account, uint256 value) internal {
2469     _tDividendTotal = _tDividendTotal + value;
2470     magnifiedDividendCorrections[account] = magnifiedDividendCorrections[
2471     account
2472     ].sub((magnifiedDividendPerShare.mul(value)).toInt256Safe());
```



# SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 2476

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2475 function _dburn(address account, uint256 value) internal {
2476     _tDividendTotal = _tDividendTotal - value;
2477     magnifiedDividendCorrections[account] = magnifiedDividendCorrections[
2478     account
2479     ].add((magnifiedDividendPerShare.mul(value)).toInt256Safe());
```

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

LINE 2555

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2554 function setBuybackUpperLimit(uint256 buyBackLimit) external onlyOwner {  
2555     buyBackUpperLimit = buyBackLimit * (10**18);  
2556 }  
2557  
2558 function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner {
```

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

LINE 2560

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2559     require(maxTxPercent >= minMxTxPercentage && maxTxPercent <= 10000, "err");
2560     _maxTxAmount = _tTotal.mul(maxTxPercent).div(10**4);
2561 }
2562
2563 function excludeFromMaxTx(address account, bool isExcluded) public onlyOwner {
```

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

LINE 2572

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2571 );  
2572 _maxWalletAmount = _tTotal.mul(maxWalletPercent).div(10**4);  
2573 }  
2574  
2575 function excludeFromMaxWallet(address account, bool isExcluded)
```

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

LINE 405

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
404 uint256 index = map.indexOf[key];
405 uint256 lastIndex = map.keys.length - 1;
406 address lastKey = map.keys[lastIndex];
407
408 map.indexOf[lastKey] = index;
```

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

LINE 2402

## low SEVERITY

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

## Source File

- Zodiac.sol

## Locations

```
2401  if (_excluded[i] == account) {
2402  _excluded[i] = _excluded[_excluded.length - 1];
2403  _tOwned[account] = 0;
2404  _isExcluded[account] = false;
2405  _excluded.pop();
```

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

LINE 7

### low SEVERITY

The current pragma Solidity directive is `""^0.8.15"`. 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

- Zodiac.sol

### Locations

```
6
7  pragma solidity ^0.8.15;
8
9  interface IERC20 {
10     function totalSupply() external view returns (uint256);
```

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

LINE 1039

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1038
1039  address dead = address(0xdead);
1040
1041  uint8 public maxLiqFee = 10;
1042  uint8 public maxTaxFee = 10;
```



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

LINE 1147

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1146
1147     bool walletFeeInBNB;
1148     bool walletCharityFeeInBNB;
1149     bool walletDevFeeInBNB;
1150
```

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

LINE 1148

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1147     bool walletFeeInBNB;  
1148     bool walletCharityFeeInBNB;  
1149     bool walletDevFeeInBNB;  
1150  
1151     address marketingFeeToken;
```

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

LINE 1149

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1148    bool walletCharityFeeInBNB;  
1149    bool walletDevFeeInBNB;  
1150  
1151    address marketingFeeToken;  
1152    address charityFeeToken;
```

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

LINE 1151

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1150
1151 address marketingFeeToken;
1152 address charityFeeToken;
1153 address devFeeToken;
1154
```

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

LINE 1152

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1151 address marketingFeeToken;  
1152 address charityFeeToken;  
1153 address devFeeToken;  
1154  
1155 bool inSwapAndLiquify;
```

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

LINE 1153

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1152 address charityFeeToken;  
1153 address devFeeToken;  
1154  
1155 bool inSwapAndLiquify;  
1156 bool public swapAndLiquifyEnabled = true;
```

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

LINE 1155

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1154
1155  bool inSwapAndLiquify;
1156  bool public swapAndLiquifyEnabled = true;
1157
1158  uint256 public _maxTxAmount;
```

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

LINE 1824

## 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

- Zodiac.sol

## Locations

```
1823     gas,  
1824     tx.origin  
1825   );  
1826   }  
1827   }
```



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

LINE 1920

## 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

- Zodiac.sol

## Locations

```
1919     gas,  
1920     tx.origin  
1921   );  
1922   }  
1923
```

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

LINE 2429

## 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

- Zodiac.sol

## Locations

```
2428     gas,  
2429     tx.origin  
2430   );  
2431   }  
2432
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 374

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
373  {  
374  return map.keys[index];  
375  }  
376  
377  function size(Map storage map) internal view returns (uint256) {
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 406

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
405  uint256 lastIndex = map.keys.length - 1;
406  address lastKey = map.keys[lastIndex];
407
408  map.indexOf[lastKey] = index;
409  delete map.indexOf[key];
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 411

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
410
411  map.keys[index] = lastKey;
412  map.keys.pop();
413  }
414  }
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1254

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1253
1254     router = _addrs[0];
1255     payable(_addrs[1]).transfer(msg.value);
1256
1257     IUniswapV2Router02 _pcsV2Router = IUniswapV2Router02(router);
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1255

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1254     router = _addrs[0];
1255     payable(_addrs[1]).transfer(msg.value);
1256
1257     IUniswapV2Router02 _pcsV2Router = IUniswapV2Router02(router);
1258     // Create a uniswap pair for this new token
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1643

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1642  if (  
1643  _rOwned[_excluded[i]] > rSupply ||  
1644  _tOwned[_excluded[i]] > tSupply  
1645  ) return (_rTotal, _tTotal);  
1646  rSupply = rSupply.sub(_rOwned[_excluded[i]]);
```



## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1644

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1643  _rOwned[_excluded[i]] > rSupply ||  
1644  _tOwned[_excluded[i]] > tSupply  
1645  ) return (_rTotal, _tTotal);  
1646  rSupply = rSupply.sub(_rOwned[_excluded[i]]);  
1647  tSupply = tSupply.sub(_tOwned[_excluded[i]]);
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1646

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1645 ) return (_rTotal, _tTotal);
1646 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1647 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1648 }
1649 if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1647

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1646 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
1647 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
1648 }
1649 if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
1650 return (rSupply, tSupply);
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1953

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1952 address[] memory path = new address[](2);
1953 path[0] = address(this);
1954 path[1] = pcsV2Router.WETH();
1955
1956 _approve(address(this), address(pcsV2Router), tokenAmount);
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1954

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1953 path[0] = address(this);
1954 path[1] = pcsV2Router.WETH();
1955
1956 _approve(address(this), address(pcsV2Router), tokenAmount);
1957
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1971

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1970 address[] memory path = new address[](2);
1971 path[0] = pcsV2Router.WETH();
1972 path[1] = address(this);
1973
1974 // make the swap
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1972

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1971 path[0] = pcsV2Router.WETH();
1972 path[1] = address(this);
1973
1974 // make the swap
1975 pcsV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1993

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1992 address[] memory path = new address[](3);
1993 path[0] = address(this);
1994 path[1] = pcsV2Router.WETH();
1995 path[2] = feeToken;
1996
```



## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1994

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1993 path[0] = address(this);
1994 path[1] = pcsV2Router.WETH();
1995 path[2] = feeToken;
1996
1997 _approve(address(this), address(pcsV2Router), tokenAmount);
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 1995

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
1994     path[1] = pcsV2Router.WETH();
1995     path[2] = feeToken;
1996
1997     _approve(address(this), address(pcsV2Router), tokenAmount);
1998
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2246

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
2245
2246 address account = tokenHoldersMap.keys[_lastProcessedIndex];
2247
2248 if (canAutoClaim(lastClaimTimes[account])) {
2249     if (processAccount payable(account), true) {
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2401

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
2400 for (uint256 i = 0; i < _excluded.length; i++) {
2401   if (_excluded[i] == account) {
2402     _excluded[i] = _excluded[_excluded.length - 1];
2403     _tOwned[account] = 0;
2404     _isExcluded[account] = false;
```

## SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 2402

### low SEVERITY

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

### Source File

- Zodiac.sol

### Locations

```
2401  if (_excluded[i] == account) {  
2402  _excluded[i] = _excluded[_excluded.length - 1];  
2403  _tOwned[account] = 0;  
2404  _isExcluded[account] = false;  
2405  _excluded.pop();
```

# 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.