Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project: AstroBirdz Protocol
Platform: Binance Smart Chain
Language: Solidity
Date: February 8th, 2022

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the AstroBirdz team to perform the Security audit of the AstroBirdz Protocol smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on February 8th, 2022.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

AstroBirdz is a standard BEP20 token smart contract. This audit only considers AstroBirdz protocol smart contract, and does not cover any other smart contracts on the platform.

Audit scope

Name	**Code Review and Security Analysis Report for AstroBirdz Protocol Smart Contracts**
Platform	**BSC / Solidity**
File 1	AstroBirdsV2.sol
File 1 MD5 Hash	B33A037FDD7783C41CDB63997CE2CD5C
File 2	AstroBirdzDividendTracker.sol
File 2 MD5 Hash	949E6D20DBFED7C06449B48ED6598561
Audit Date	February 8th,2022

Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1 AstroBirdsV2.sol • Decimals: 18 • PSI rewards Fee: 1% • Liquidity Pool: 3% • Marketing Fee: 3% • Team Fee: 1% • Buy back Fee: 3% • Sell Limit: 50000 • Maximum Amount Per Transaction: 5 Million • Minimum Tokens Before Swap: 10,000 • Gas For Processing: 0.3 Million	**YES, This is valid.** **Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.**
File 2 AstroBirdzDividendTracker.sol • Name: AstroBirdz Dividend Tracker • Symbol: ABZDT • Decimals: 18	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. These contracts do contain owner control, which does not make them fully decentralized.

We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 5 low and some very low level issues. These issues are not critical ones.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	N/A
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Features claimed	Passed
	Other programming issues	Moderated
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Unused code	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Moderated
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 2 smart contract files. Smart contracts contain Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the AstroBirdz Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the AstroBirdz Protocol.

The AstroBirdz Protocol team has provided unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not** well commented on smart contracts.

Documentation

We were given an AstroBirdz Protocol smart contract code in the form of a Github web link. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **not well** commented. So it is not easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Use of Dependencies

As per our observation, the libraries are used in this smart contracts infrastructure that are based on well known industry standard open source projects.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

AstroBirdsV2.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	initialize	write	Passed	No Issue
3	initializer	modifier	Passed	No Issue
4	onlyInitializing	modifier	Passed	No Issue
5	_isConstructo	read	Passed	No Issue
6	__Context_init	internal	access only Initializing	No Issue
7	__Context_init_unchained	internal	access only Initializing	No Issue
8	_msgSender	internal	Passed	No Issue
9	_msgData	internal	Passed	No Issue
10	onlyOwner	modifier	Passed	No Issue
11	lockTheSwap	modifier	Passed	No Issue
12	receive	external	Passed	No Issue
13	_ERC20_init	internal	access only Initializer	No Issue
14	initPSIDividendTracker	external	Critical operation lacks event log	Refer Audit Findings
15	name	read	Passed	No Issue
16	symbol	read	Passed	No Issue
17	decimals	read	Passed	No Issue
18	totalSupply	read	Passed	No Issue
19	balanceOf	read	Passed	No Issue
20	calculateLiquidityFee	internal	Passed	No Issue
21	calculatePSIFee	internal	Passed	No Issue
22	calculateMarketingFee	internal	Passed	No Issue
23	calculateTeamFee	internal	Passed	No Issue
24	calculateBuybackFee	internal	Passed	No Issue
25	setPSIFee	write	access only Owner	No Issue
26	setLiquidityFee	write	access only Owner	No Issue
27	setBuybackFee	write	access only Owner	No Issue
28	setMarketingFee	write	access only Owner	No Issue
29	setTeamFee	write	access only Owner	No Issue
30	toggleSellLimit	write	access only Owner	No Issue
31	setBuybackAddress	write	access only Owner	No Issue
32	changeMarketingAddress	write	access only Owner	No Issue
33	changeTeamAddress	write	access only Owner	No Issue
34	changePSIAddress	write	access only Owner	No Issue
35	changeLiquidityAddress	write	access only Owner	No Issue
36	changeSellLimit	write	access only Owner	No Issue
37	changeMaxtx	write	access only Owner	No Issue

38	addExcludedAddress	write	access only Owner	No Issue
39	removeExcludedAddress	write	access only Owner	No Issue
40	excludeFromFeesAndDividends	write	access only Owner	No Issue
41	addNewRouter	external	Function input parameters lack of check	Refer Audit Findings
42	setAutomatedMarketMakerPair	external	access only Owner	No Issue
43	_setAutomatedMarketMakerPair	write	Passed	No Issue
44	updateGasForProcessing	external	access only Owner	No Issue
45	transferOwnership	write	access only Owner	No Issue
46	getUnlockTime	read	Passed	No Issue
47	lock	write	access only Owner	No Issue
48	unlock	write	Regain Ownership	Refer Audit Findings
49	multiTransfer	write	Passed	No Issue
50	processDividendTracker	external	Passed	No Issue
51	transfer	write	Passed	No Issue
52	allowance	read	Passed	No Issue
53	approve	write	Passed	No Issue
54	transferFrom	write	Passed	No Issue
55	increaseAllowance	write	Passed	No Issue
56	decreaseAllowance	write	Passed	No Issue
57	setSwapAndLiquifyEnabled	write	access only Owner	No Issue
58	_transferExcluded	internal	Passed	No Issue
59	_transfer	internal	Passed	No Issue
60	_fixDividendTrackerBalancer	write	Passed	No Issue
61	_simpleTransfer	internal	Passed	No Issue
62	performSwapAndLiquify	external	access only Owner	No Issue
63	swapAndLiquify	write	access by lock The Swap	No Issue
64	toggleTrading	write	access only Owner	No Issue
65	togglePaused	write	access only Owner	No Issue
66	swapTokensForEth	write	Passed	No Issue
67	addLiquidity	write	Centralized risk in addLiquidity	Refer Audit Findings
68	swapAndSendDividends	write	Passed	No Issue
69	swapETHForPSI	write	Passed	No Issue
70	_mint	internal	Passed	No Issue
71	mint	external	Unlimited mint	Refer Audit Findings
72	_burn	internal	Passed	No Issue
73	burn	external	Passed	No Issue
74	_approve	internal	Passed	No Issue
75	_setupDecimals	internal	Passed	No Issue
76	_beforeTokenTransfer	internal	Passed	No Issue

AstroBirdzDividendTracker.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	getOwner	read	Passed	No Issue
3	recoverERC20	write	onlyOwnerOrParentToken	No Issue
4	_transfer	internal	Passed	No Issue
5	withdrawDividend	write	Passed	No Issue
6	excludeFromDividends	external	onlyOwnerOrParentToken	No Issue
7	includeInDividends	external	onlyOwnerOrParentToken	No Issue
8	updateClaimWait	external	access only Owner	No Issue
9	updateMinTokenBalance	external	access only Owner	No Issue
10	getLastProcessedIndex	external	Passed	No Issue
11	getNumberOfTokenHolders	external	Passed	No Issue
12	getAccount	read	Passed	No Issue
13	getAccountAtIndex	external	Passed	No Issue
14	canAutoClaim	read	Passed	No Issue
15	ensureBalance	external	Passed	No Issue
16	ensureBalanceForUsers	external	Function input parameters lack of check	Refer Audit Findings
17	bytesToAddress	write	Passed	No Issue
18	ensureBalanceForUser	write	access only Owner	No Issue
19	setBalance	external	onlyOwnerOrParentToken	No Issue
20	process	external	Passed	No Issue
21	processAccount	external	Function input parameters lack of check	Refer Audit Findings
22	owner	read	Passed	No Issue
23	onlyOwner	modifier	Passed	No Issue
24	renounceOwnership	write	access only Owner	No Issue
25	transferOwnership	write	access only Owner	No Issue
26	_transferOwnership	internal	Passed	No Issue
27	onlyOwnerOrParentToken	modifier	Passed	No Issue
28	distributeDividends	write	access only Owner Or Parent Token	No Issue
29	withdrawDividend	write	Passed	No Issue

30	_withdrawDividendOfUser	internal	Passed	No Issue
31	dividendOf	read	Passed	No Issue
32	withdrawableDividendOf	read	Passed	No Issue
33	withdrawnDividendOf	read	Passed	No Issue
34	accumulativeDividendOf	read	Passed	No Issue
35	_transfer	internal	Passed	No Issue
36	_mint	internal	Passed	No Issue
37	_burn	internal	Passed	No Issue
38	_setBalance	internal	Passed	No Issue
39	recoverERC20	write	access only Owner	No Issue
40	recoverETH	write	access only Owner	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Email: audit@EtherAuthority.io

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Function input parameters lack of check: **AstroBirdzDividendTracker.sol**
Variable validation is not performed in below functions :

- ensureBalanceForUser
- processAccount
- addNewRouter

  The owner can update the router that generates liquidity to an address or contract of choice (including the zero address). This contract could be a malicious contract that simply keeps the tokens sent to it and thus drains all the fees. Additionally, this contract could be used to revert sell transactions turning the token into a honeypot.

Resolution: We advise to put validation : int type variables should be greater than 0 and address type variables should not be address(0).
AddNewRouter - Consider removing this function. If this is not possible, consider using an Owner account that is behind a significantly long time lock so investors can reasonably see this change coming and inspect the new router. Also consider requiring the router address to be non-zero.

(2) Critical operation lacks event log:

Missing event log for:

- initPSIDividendTracker

Resolution: Please write an event log for listed events.

(3) Unlimited mint: **AstroBirdsV2.sol**

```
function mint(address account, uint256 amount) external onlyOwner {
    require(_msgSender() == tx.origin, "Invalid Request");
    _mint(account, amount);
}
```

Token minting without any maximum limit is considered inappropriate for tokenomics.

Resolution: We recommend placing some limit on token minting to mitigate this issue.

(4) Centralized risk in addLiquidity: **AstroBirdsV2.sol**

In addLiquidity function, _liquidityPoolAddress gets Tokens from the Pool. If the private key of the _liquidityPoolAddress wallet would be compromised, then it would create a problem.

Resolution: Ideally this can be a governance smart contract. On another hand, the _liquidityPoolAddress can accept this risk and handle the private key very securely.

(5) Regain Ownership: **AstroBirdsV2.sol**

unlock() function can be used to take back ownership after ownership being transferred to a new owner.

Resolution: We advise to set previousOwner = address(0) after unlocking the contract for the owner.

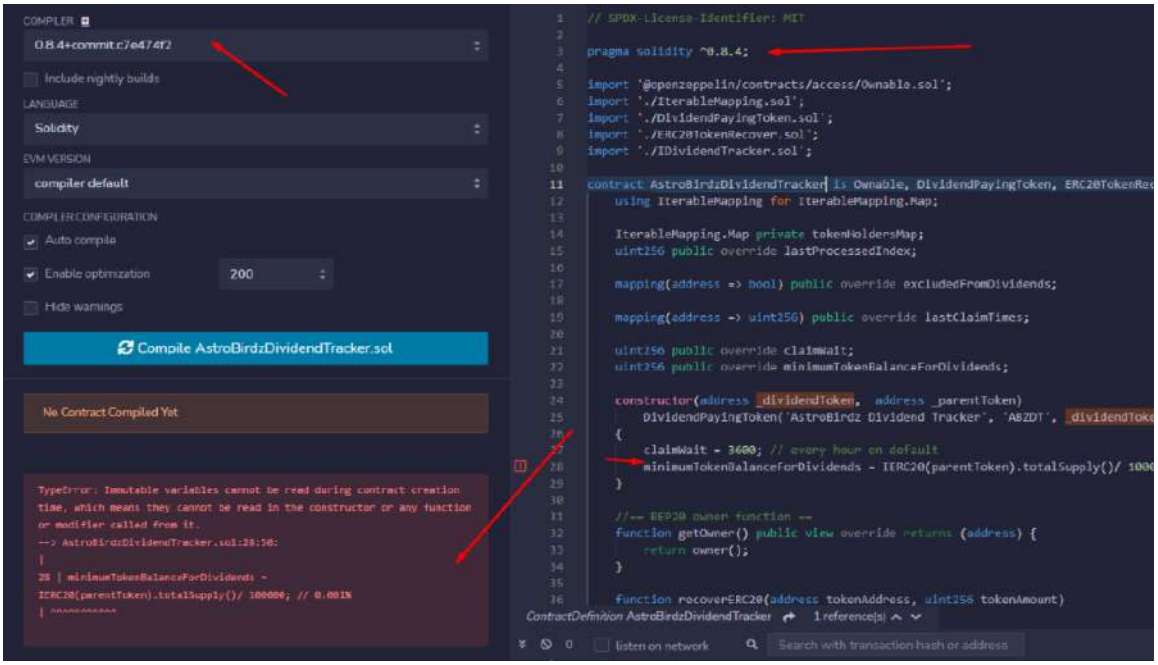
Very Low / Informational / Best practices:

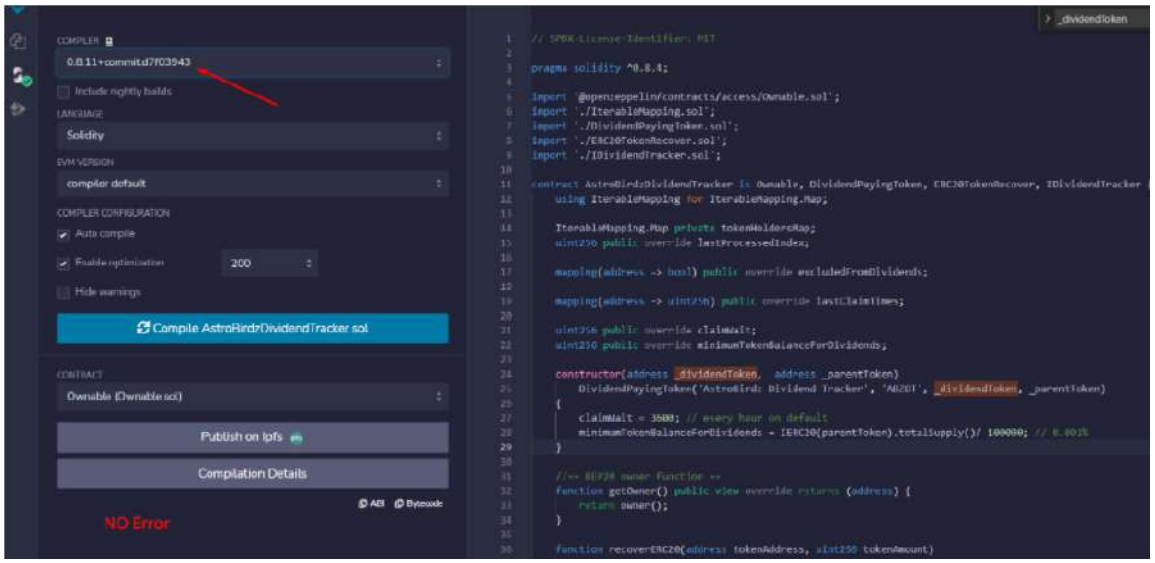
(1) Make variable constant: **AstroBirdsV2.sol**

_name, _symbol , _decimal: Values of these variables will be unchanged.

Resolution: We suggest adding a "constant" keyword for these variables. This will save some gas.

(2) Other Programming Issue: **AstroBirdzDividendTracker.sol**





Immutable variables cannot be read during contract creation time, which means they cannot be read in the constructor or any function or modifier called from it. It requires the latest solidity version.

Resolution: We advise to deploy with the latest solidity version.

Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- recoverERC20: The AstroBirdzDividendTracker owner or partner can recover ERC20 Token address and amount.
- excludeFromDividends: The AstroBirdzDividendTracker owner or partner can exclude from dividends.
- includeInDividends: The AstroBirdzDividendTracker owner or partner can be included in dividends.
- updateClaimWait: The AstroBirdzDividendTracker owner or partner can update claim time.
- updateMinTokenBalance: The AstroBirdzDividendTracker owner or partner can update minimum token balance.
- ensureBalanceForUsers: The AstroBirdzDividendTracker owner or partner can ensure balance for multiple users.
- ensureBalanceForUser: The AstroBirdzDividendTracker owner or partner can ensure balance for a single user.
- setBalance: The AstroBirdzDividendTracker owner or partner can set balance.
- processAccount: The AstroBirdzDividendTracker owner or partner can process the account automatically.

Conclusion

We were given a contract code in the form of files. And we have used all possible tests based on given objects as files. We have not observed any major issues in the smart contracts. So, **it's good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

Email: audit@EtherAuthority.io

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

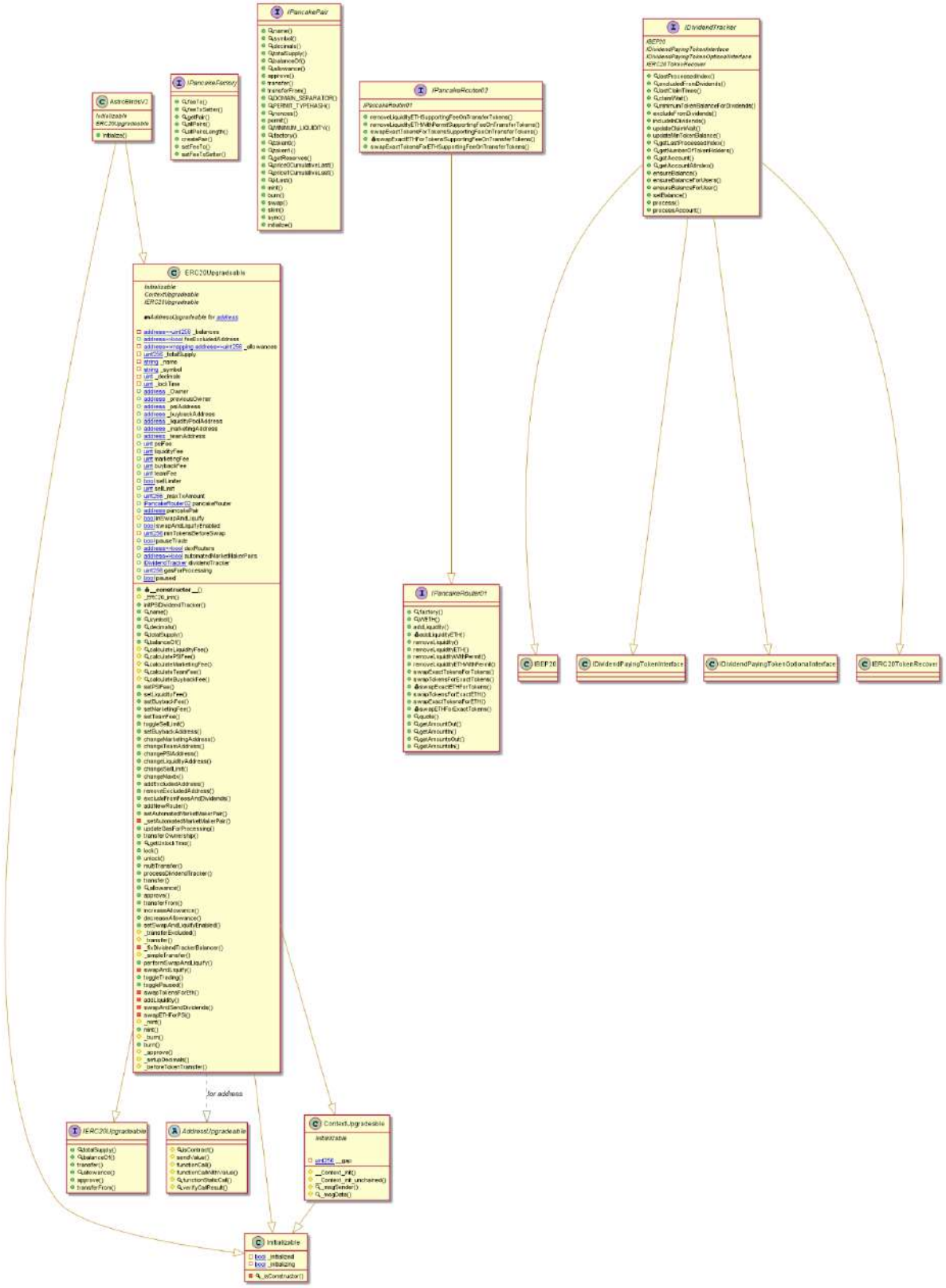
Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

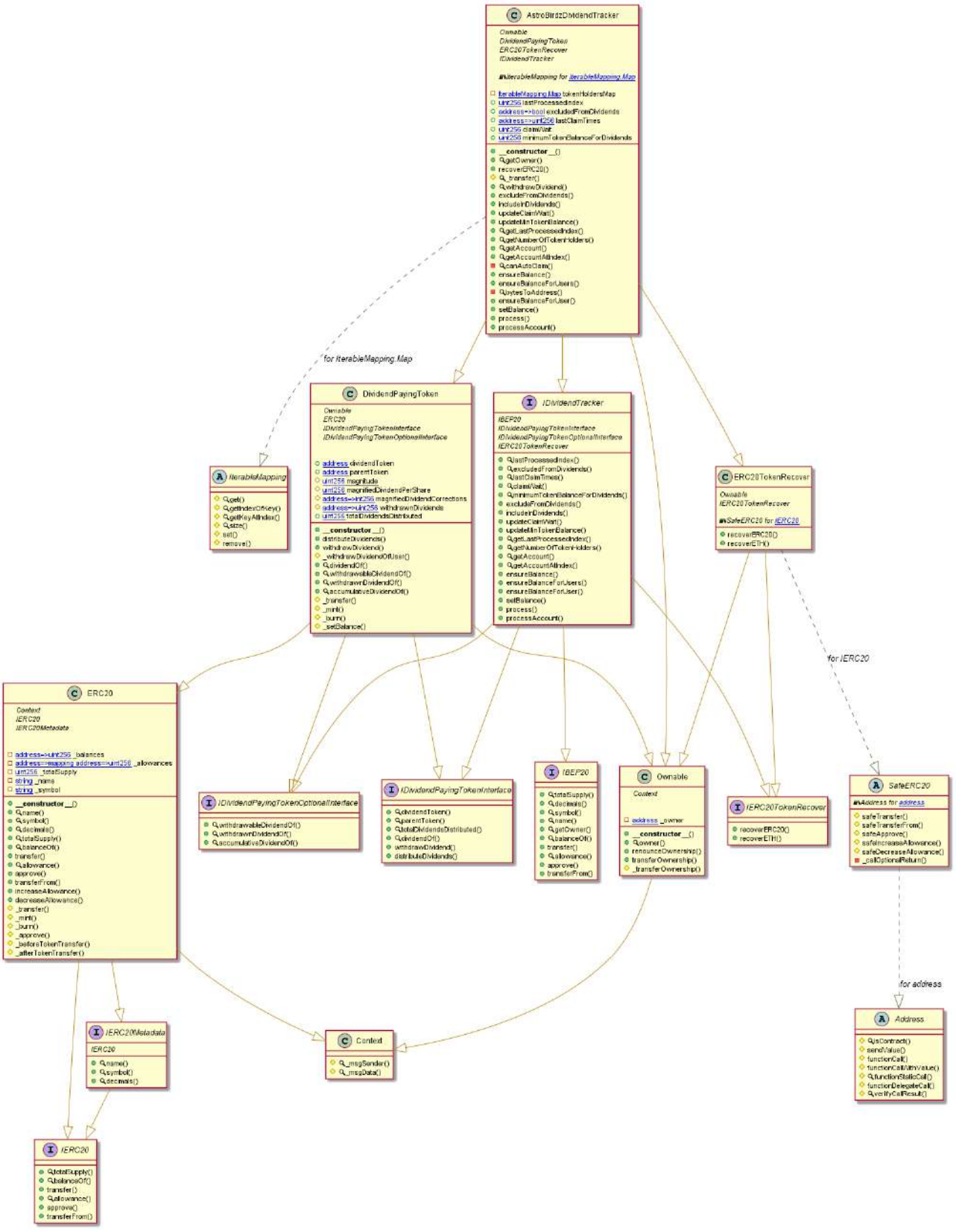
Appendix

Code Flow Diagram - AstroBirdz Protocol

AstroBirdsV2 Diagram



AstroBirdzDividendTracker Diagram



Slither Results Log

Slither log >> AstroBirdsV2.sol

```
INFO:Detectors:
ERC20Upgradeable.addLiquidity(uint256,uint256) (AstroBirdsV2.sol#1426-1439) sends eth to arbitrary user
        Dangerous calls:
        - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.timestamp) (
AstroBirdsV2.sol#1431-1438)
ERC20Upgradeable.swapETHForPSI(uint256,address) (AstroBirdsV2.sol#1450-1463) sends eth to arbitrary user
        Dangerous calls:
        - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.timestamp) (
AstroBirdsV2.sol#1457-1462)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in ERC20Upgradeable._transfer(address,address,uint256) (AstroBirdsV2.sol#1298-1342):
        External calls:
        - _fixDividendTrackerBalancer(sender,recipient,amount) (AstroBirdsV2.sol#1322)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timest
amp) (AstroBirdsV2.sol#1417-1423)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        State variables written after the call(s):
        - _simpleTransfer(sender,recipient,amount) (AstroBirdsV2.sol#1335)
                - _balances[sender] = senderBalance - amount (AstroBirdsV2.sol#1366)
                - _balances[recipient] += amount (AstroBirdsV2.sol#1367)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
ERC20Upgradeable.swapAndLiquify(uint256) (AstroBirdsV2.sol#1378-1399) performs a multiplication on the result of a division:
        -feeBalance = swappedBalance - ((swappedBalance * (liquidityFee / 2)) / (totalFees - (liquidityFee / 2))) (AstroBirdsV2.
sol#1385-1386)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in ERC20Upgradeable._transferExcluded(address,address,uint256) (AstroBirdsV2.sol#1283-1296):
        External calls:
        - _fixDividendTrackerBalancer(sender,recipient,amount) (AstroBirdsV2.sol#1294)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
        State variables written after the call(s):
        - _simpleTransfer(sender,recipient,amount) (AstroBirdsV2.sol#1295)
                - _balances[sender] = senderBalance - amount (AstroBirdsV2.sol#1366)
                - _balances[recipient] += amount (AstroBirdsV2.sol#1367)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ERC20Upgradeable.mint(address,uint256) (AstroBirdsV2.sol#1484-1487) uses tx.origin for authorization: require(bool,string)(_msgS
ender() == tx.origin,Invalid Request) (AstroBirdsV2.sol#1485)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-usage-of-txorigin
INFO:Detectors:
ERC20Upgradeable._transfer(address,address,uint256).lastProcessedIndex (AstroBirdsV2.sol#1338) is a local variable never initial
ized
ERC20Upgradeable._transfer(address,address,uint256).iterations (AstroBirdsV2.sol#1338) is a local variable never initialized
ERC20Upgradeable._transfer(address,address,uint256).claims (AstroBirdsV2.sol#1338) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
ERC20Upgradeable._transfer(address,address,uint256) (AstroBirdsV2.sol#1298-1342) ignores return value by dividendTracker.process
(gasForProcessing) (AstroBirdsV2.sol#1338-1340)
ERC20Upgradeable.addLiquidity(uint256,uint256) (AstroBirdsV2.sol#1426-1439) ignores return value by pancakeRouter.addLiquidityET
H{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.timestamp) (AstroBirdsV2.sol#1431-1438)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
AstroBirdsV2.initialize(string,string,address,address,address,address,address)._name (AstroBirdsV2.sol#1568) shadows:
        - ERC20Upgradeable._name (AstroBirdsV2.sol#799) (state variable)
```

```
AstroBirdsV2.initialize(string,string,address,address,address,address,address)._symbol (AstroBirdsV2.sol#1569) shadows:
        - ERC20Upgradeable._symbol (AstroBirdsV2.sol#800) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Variable 'ERC20Upgradeable._transfer(address,address,uint256).lastProcessedIndex (AstroBirdsV2.sol#1338)' in ERC20Upgradeable._t
ransfer(address,address,uint256) (AstroBirdsV2.sol#1298-1342) potentially used before declaration: ProcessedDividendTracker(iter
ations,claims,lastProcessedIndex,true,gasForProcessing,tx.origin) (AstroBirdsV2.sol#1339)
Variable 'ERC20Upgradeable._transfer(address,address,uint256).claims (AstroBirdsV2.sol#1338)' in ERC20Upgradeable._transfer(addr
ess,address,uint256) (AstroBirdsV2.sol#1298-1342) potentially used before declaration: ProcessedDividendTracker(iterations,claim
s,lastProcessedIndex,true,gasForProcessing,tx.origin) (AstroBirdsV2.sol#1339)
Variable 'ERC20Upgradeable._transfer(address,address,uint256).iterations (AstroBirdsV2.sol#1338)' in ERC20Upgradeable._transfer(
address,address,uint256) (AstroBirdsV2.sol#1298-1342) potentially used before declaration: ProcessedDividendTracker(iterations,c
laims,lastProcessedIndex,true,gasForProcessing,tx.origin) (AstroBirdsV2.sol#1339)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in ERC20Upgradeable._ERC20_init(string,string,address,address,address,address,address) (AstroBirdsV2.sol#878-910):
        External calls:
        - pancakePair = IPancakeFactory(pancakeRouter.factory()).createPair(address(this),pancakeRouter.WETH()) (AstroBirdsV2.so
l#907)
        State variables written after the call(s):
        - feeExcludedAddress[_msgSender()] = true (AstroBirdsV2.sol#909)
Reentrancy in ERC20Upgradeable._transfer(address,address,uint256) (AstroBirdsV2.sol#1298-1342):
```

```
        External calls:
        - pancakePair = IPancakeFactory(pancakeRouter.factory()).createPair(address(this),pancakeRouter.WETH()) (AstroBirdsV2.so
l#907)
        State variables written after the call(s):
        - feeExcludedAddress[_msgSender()] = true (AstroBirdsV2.sol#909)
Reentrancy in ERC20Upgradeable._transfer(address,address,uint256) (AstroBirdsV2.sol#1298-1342):
        External calls:
        - _fixDividendTrackerBalancer(sender,recipient,amount) (AstroBirdsV2.sol#1322)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timest
amp) (AstroBirdsV2.sol#1417-1423)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        State variables written after the call(s):
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - _allowances[owner][spender] = amount (AstroBirdsV2.sol#1534)
Reentrancy in ERC20Upgradeable.addNewRouter(address,bool) (AstroBirdsV2.sol#1075-1090):
        External calls:
        - dividendTracker.excludeFromDividends(_router) (AstroBirdsV2.sol#1077)
        - _pancakePair = IPancakeFactory(_pancakeRouter.factory()).createPair(address(this),_pancakeRouter.WETH()) (AstroBirdsV2
.sol#1082)
        - _setAutomatedMarketMakerPair(_pancakePair,true) (AstroBirdsV2.sol#1083)
                - dividendTracker.excludeFromDividends(pair) (AstroBirdsV2.sol#1105)
```

```
        State variables written after the call(s):
        - pancakePair = _pancakePair (AstroBirdsV2.sol#1088)
        - pancakeRouter = _pancakeRouter (AstroBirdsV2.sol#1087)
Reentrancy in ERC20Upgradeable.initPSIDividendTracker(IDividendTracker) (AstroBirdsV2.sol#912-934):
        External calls:
        - dividendTracker.excludeFromDividends(address(dividendTracker)) (AstroBirdsV2.sol#917)
        - dividendTracker.excludeFromDividends(address(pancakeRouter)) (AstroBirdsV2.sol#918)
        - dividendTracker.excludeFromDividends(address(0x000000000000000000000000000000000000dEaD)) (AstroBirdsV2.sol#919)
        State variables written after the call(s):
        - dexRouters[address(pancakeRouter)] = true (AstroBirdsV2.sol#922)
Reentrancy in ERC20Upgradeable.initPSIDividendTracker(IDividendTracker) (AstroBirdsV2.sol#912-934):
        External calls:
        - dividendTracker.excludeFromDividends(address(dividendTracker)) (AstroBirdsV2.sol#917)
        - dividendTracker.excludeFromDividends(address(pancakeRouter)) (AstroBirdsV2.sol#918)
        - dividendTracker.excludeFromDividends(address(0x000000000000000000000000000000000000dEaD)) (AstroBirdsV2.sol#919)
        - _setAutomatedMarketMakerPair(pancakePair,true) (AstroBirdsV2.sol#923)
                - dividendTracker.excludeFromDividends(pair) (AstroBirdsV2.sol#1105)
        - excludeFromFeesAndDividends(address(this)) (AstroBirdsV2.sol#926)
                - dividendTracker.excludeFromDividends(excludedA) (AstroBirdsV2.sol#1072)
        State variables written after the call(s):
        - excludeFromFeesAndDividends(address(this)) (AstroBirdsV2.sol#926)
                - feeExcludedAddress[excludedA] = true (AstroBirdsV2.sol#1065)
Reentrancy in ERC20Upgradeable.initPSIDividendTracker(IDividendTracker) (AstroBirdsV2.sol#912-934):
        External calls:
        - dividendTracker.excludeFromDividends(address(dividendTracker)) (AstroBirdsV2.sol#917)
        - dividendTracker.excludeFromDividends(address(pancakeRouter)) (AstroBirdsV2.sol#918)
        - dividendTracker.excludeFromDividends(address(0x000000000000000000000000000000000000dEaD)) (AstroBirdsV2.sol#919)
        - _setAutomatedMarketMakerPair(pancakePair,true) (AstroBirdsV2.sol#923)
                - dividendTracker.excludeFromDividends(pair) (AstroBirdsV2.sol#1105)
        - excludeFromFeesAndDividends(address(this)) (AstroBirdsV2.sol#926)
                - dividendTracker.excludeFromDividends(excludedA) (AstroBirdsV2.sol#1072)
        - excludeFromFeesAndDividends(_Owner) (AstroBirdsV2.sol#927)
                - dividendTracker.excludeFromDividends(excludedA) (AstroBirdsV2.sol#1072)
        State variables written after the call(s):
        - _liquidityPoolAddress = _Owner (AstroBirdsV2.sol#932)
        - _psiAddress = dividendTracker.dividendToken() (AstroBirdsV2.sol#933)
        - excludeFromFeesAndDividends(_Owner) (AstroBirdsV2.sol#927)
```

```
                - feeExcludedAddress[excludedA] = true (AstroBirdsV2.sol#1065)
        - gasForProcessing = 300000 (AstroBirdsV2.sol#930)
        - minTokensBeforeSwap = 10000 * (10 ** decimals()) (AstroBirdsV2.sol#931)
Reentrancy in AstroBirdsV2.initialize(string,string,address,address,address,address,address) (AstroBirdsV2.sol#1567-1587):
        External calls:
        - ERC20Upgradeable._ERC20_init(_name,_symbol,marketingAddress_,teamAddress_,psiAddress_,buybackAddress_,router_) (AstroB
irdsV2.sol#1576-1584)
                - pancakePair = IPancakeFactory(pancakeRouter.factory()).createPair(address(this),pancakeRouter.WETH()) (AstroBi
rdsV2.sol#907)
        State variables written after the call(s):
        - _mint(_msgSender(),47000000000000000000000000) (AstroBirdsV2.sol#1586)
                - _balances[account] += amount (AstroBirdsV2.sol#1480)
        - _mint(_msgSender(),47000000000000000000000000) (AstroBirdsV2.sol#1586)
                - _totalSupply += amount (AstroBirdsV2.sol#1479)
Reentrancy in ERC20Upgradeable.swapAndLiquify(uint256) (AstroBirdsV2.sol#1378-1399):
        External calls:
        - swapTokensForEth(contractTokenBalance - (forLiquidity / 2)) (AstroBirdsV2.sol#1383)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timest
amp) (AstroBirdsV2.sol#1417-1423)
        - swapAndSendDividends((feeBalance * psiFee) / totalFees) (AstroBirdsV2.sol#1392)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
        External calls sending eth:
        - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
        - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
        - swapAndSendDividends((feeBalance * psiFee) / totalFees) (AstroBirdsV2.sol#1392)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
        State variables written after the call(s):
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - _allowances[owner][spender] = amount (AstroBirdsV2.sol#1534)
```

```
Reentrancy in ERC20Upgradeable.transferFrom(address,address,uint256) (AstroBirdsV2.sol#1211-1224):
        External calls:
        - _transferExcluded(sender,recipient,amount) (AstroBirdsV2.sol#1215)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
        - _transfer(sender,recipient,amount) (AstroBirdsV2.sol#1217)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timest
amp) (AstroBirdsV2.sol#1417-1423)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
                - dividendTracker.process(gasForProcessing) (AstroBirdsV2.sol#1338-1340)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (AstroBirdsV2.sol#1217)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),currentAllowance - amount) (AstroBirdsV2.sol#1222)
                - _allowances[owner][spender] = amount (AstroBirdsV2.sol#1534)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ERC20Upgradeable._setAutomatedMarketMakerPair(address,bool) (AstroBirdsV2.sol#1098-1107):
        External calls:
        - dividendTracker.excludeFromDividends(pair) (AstroBirdsV2.sol#1105)
        Event emitted after the call(s):
        - SetAutomatedMarketMakerPair(pair,value) (AstroBirdsV2.sol#1106)
Reentrancy in ERC20Upgradeable._transfer(address,address,uint256) (AstroBirdsV2.sol#1298-1342):
```

```
        External calls:
        - _fixDividendTrackerBalancer(sender,recipient,amount) (AstroBirdsV2.sol#1322)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timest
amp) (AstroBirdsV2.sol#1417-1423)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (AstroBirdsV2.sol#1535)
                - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
        - SwapAndLiquify(contractTokenBalance,swappedBalance,forLiquidity / 2) (AstroBirdsV2.sol#1398)
                - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
        - Transfer(sender,recipient,amount) (AstroBirdsV2.sol#1368)
                - _simpleTransfer(sender,recipient,amount) (AstroBirdsV2.sol#1335)
Reentrancy in ERC20Upgradeable._transfer(address,address,uint256) (AstroBirdsV2.sol#1298-1342):
        External calls:
        - _fixDividendTrackerBalancer(sender,recipient,amount) (AstroBirdsV2.sol#1322)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
```

```
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timest
amp) (AstroBirdsV2.sol#1417-1423)
        - dividendTracker.process(gasForProcessing) (AstroBirdsV2.sol#1338-1340)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        Event emitted after the call(s):
        - ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gasForProcessing,tx.origin) (AstroBirdsV2.sol#1339)
Reentrancy in ERC20Upgradeable._transferExcluded(address,address,uint256) (AstroBirdsV2.sol#1283-1296):
        External calls:
        - _fixDividendTrackerBalancer(sender,recipient,amount) (AstroBirdsV2.sol#1294)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (AstroBirdsV2.sol#1368)
                - _simpleTransfer(sender,recipient,amount) (AstroBirdsV2.sol#1295)
Reentrancy in ERC20Upgradeable.addNewRouter(address,bool) (AstroBirdsV2.sol#1075-1090):
        External calls:
        - dividendTracker.excludeFromDividends(_router) (AstroBirdsV2.sol#1077)
        - _pancakePair = IPancakeFactory(_pancakeRouter.factory()).createPair(address(this),_pancakeRouter.WETH()) (AstroBirdsV2
.sol#1082)
```

```
        - _setAutomatedMarketMakerPair(_pancakePair,true) (AstroBirdsV2.sol#1083)
                - dividendTracker.excludeFromDividends(pair) (AstroBirdsV2.sol#1105)
        Event emitted after the call(s):
        - SetAutomatedMarketMakerPair(pair,value) (AstroBirdsV2.sol#1106)
                - _setAutomatedMarketMakerPair(_pancakePair,true) (AstroBirdsV2.sol#1083)
        - UpdateDefaultDexRouter(_router,address(pancakeRouter)) (AstroBirdsV2.sol#1086)
Reentrancy in ERC20Upgradeable.initPSIDividendTracker(IDividendTracker) (AstroBirdsV2.sol#912-934):
        External calls:
        - dividendTracker.excludeFromDividends(address(dividendTracker)) (AstroBirdsV2.sol#917)
        - dividendTracker.excludeFromDividends(address(pancakeRouter)) (AstroBirdsV2.sol#918)
        - dividendTracker.excludeFromDividends(address(0x000000000000000000000000000000000000dEaD)) (AstroBirdsV2.sol#919)
        - _setAutomatedMarketMakerPair(pancakePair,true) (AstroBirdsV2.sol#923)
                - dividendTracker.excludeFromDividends(pair) (AstroBirdsV2.sol#1105)
        Event emitted after the call(s):
        - SetAutomatedMarketMakerPair(pair,value) (AstroBirdsV2.sol#1106)
                - _setAutomatedMarketMakerPair(pancakePair,true) (AstroBirdsV2.sol#923)
Reentrancy in AstroBirdsV2.initialize(string,string,address,address,address,address,address) (AstroBirdsV2.sol#1567-1587):
        External calls:
        - ERC20Upgradeable._ERC20_init(_name,_symbol,marketingAddress_,teamAddress_,psiAddress_,buybackAddress_,router_) (AstroB
irdsV2.sol#1576-1584)
                - pancakePair = IPancakeFactory(pancakeRouter.factory()).createPair(address(this),pancakeRouter.WETH()) (AstroBi
rdsV2.sol#907)
        Event emitted after the call(s):
        - Transfer(address(0),account,amount) (AstroBirdsV2.sol#1481)
                - _mint(_msgSender(),47000000000000000000000000) (AstroBirdsV2.sol#1586)
Reentrancy in ERC20Upgradeable.processDividendTracker(uint256) (AstroBirdsV2.sol#1155-1158):
        External calls:
        - (iterations,claims,lastProcessedIndex) = dividendTracker.process(gas) (AstroBirdsV2.sol#1156)
        Event emitted after the call(s):
        - ProcessedDividendTracker(iterations,claims,lastProcessedIndex,false,gas,tx.origin) (AstroBirdsV2.sol#1157)
Reentrancy in ERC20Upgradeable.swapAndLiquify(uint256) (AstroBirdsV2.sol#1378-1399):
        External calls:
        - swapTokensForEth(contractTokenBalance - (forLiquidity / 2)) (AstroBirdsV2.sol#1383)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timest
amp) (AstroBirdsV2.sol#1417-1423)
        - swapAndSendDividends((feeBalance * psiFee) / totalFees) (AstroBirdsV2.sol#1392)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
```

```
stamp) (AstroBirdsV2.sol#1457-1462)
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
        External calls sending eth:
        - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
        - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
        - swapAndSendDividends((feeBalance * psiFee) / totalFees) (AstroBirdsV2.sol#1392)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (AstroBirdsV2.sol#1535)
                - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
Reentrancy in ERC20Upgradeable.swapAndLiquify(uint256) (AstroBirdsV2.sol#1378-1399):
        External calls:
        - swapTokensForEth(contractTokenBalance - (forLiquidity / 2)) (AstroBirdsV2.sol#1383)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timest
amp) (AstroBirdsV2.sol#1417-1423)
        - swapAndSendDividends((feeBalance * psiFee) / totalFees) (AstroBirdsV2.sol#1392)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
        External calls sending eth:
        - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
        - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
        - swapAndSendDividends((feeBalance * psiFee) / totalFees) (AstroBirdsV2.sol#1392)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
        - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
```

```
Reentrancy in ERC20Upgradeable.transferFrom(address,address,uint256) (AstroBirdsV2.sol#1211-1224):
        External calls:
        - _transferExcluded(sender,recipient,amount) (AstroBirdsV2.sol#1215)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
        - _transfer(sender,recipient,amount) (AstroBirdsV2.sol#1217)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - dividendTracker.setBalance(address(sender),balanceOf(sender)) (AstroBirdsV2.sol#1351)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - pancakeRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timest
amp) (AstroBirdsV2.sol#1417-1423)
                - dividendTracker.setBalance(address(sender),balanceOf(sender) - amount) (AstroBirdsV2.sol#1353)
                - dividendTracker.setBalance(address(recipient),balanceOf(recipient) + amount) (AstroBirdsV2.sol#1354)
                - dividendTracker.process(gasForProcessing) (AstroBirdsV2.sol#1338-1340)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (AstroBirdsV2.sol#1217)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (AstroBirdsV2.sol#1535)
                - _approve(sender,_msgSender(),currentAllowance - amount) (AstroBirdsV2.sol#1222)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
ERC20Upgradeable.unlock() (AstroBirdsV2.sol#1140-1145) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp > _lockTime,Contract is still locked) (AstroBirdsV2.sol#1142)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (AstroBirdsV2.sol#255-275) uses assembly
        - INLINE ASM (AstroBirdsV2.sol#267-270)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (AstroBirdsV2.sol#166-168) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (AstroBirdsV2.sol#176-182) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (AstroBirdsV2.sol#195-201) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (AstroBirdsV2.sol#209-220) is never used and should be re
moved
AddressUpgradeable.functionStaticCall(address,bytes) (AstroBirdsV2.sol#228-230) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (AstroBirdsV2.sol#238-247) is never used and should be removed
AddressUpgradeable.isContract(address) (AstroBirdsV2.sol#117-123) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (AstroBirdsV2.sol#141-146) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (AstroBirdsV2.sol#255-275) is never used and should be removed
ContextUpgradeable.__Context_init() (AstroBirdsV2.sol#774-775) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (AstroBirdsV2.sol#777-778) is never used and should be removed
ContextUpgradeable._msgData() (AstroBirdsV2.sol#783-785) is never used and should be removed
ERC20Upgradeable._setupDecimals(uint8) (AstroBirdsV2.sol#1545-1547) is never used and should be removed
Initializable._isConstructo() (AstroBirdsV2.sol#768-770) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (AstroBirdsV2.sol#14) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (AstroBirdsV2.sol#141-146):
        - (success) = recipient.call{value: amount}() (AstroBirdsV2.sol#144)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (AstroBirdsV2.sol#209-220):
        - (success,returndata) = target.call{value: value}(data) (AstroBirdsV2.sol#218)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (AstroBirdsV2.sol#238-247):
        - (success,returndata) = target.staticcall(data) (AstroBirdsV2.sol#245)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IPancakePair.DOMAIN_SEPARATOR() (AstroBirdsV2.sol#308) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (AstroBirdsV2.sol#309) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (AstroBirdsV2.sol#326) is not in mixedCase
Function IPancakeRouter01.WETH() (AstroBirdsV2.sol#346) is not in mixedCase
Function ContextUpgradeable.__Context_init() (AstroBirdsV2.sol#774-775) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (AstroBirdsV2.sol#777-778) is not in mixedCase
Variable ContextUpgradeable.__gap (AstroBirdsV2.sol#788) is not in mixedCase
Function ERC20Upgradeable._ERC20_init(string,string,address,address,address,address,address) (AstroBirdsV2.sol#878-910) is not i
n mixedCase
Parameter ERC20Upgradeable._ERC20_init(string,string,address,address,address,address,address)._nm (AstroBirdsV2.sol#879) is not
in mixedCase
Parameter ERC20Upgradeable._ERC20_init(string,string,address,address,address,address,address)._sym (AstroBirdsV2.sol#880) is not
 in mixedCase
Parameter ERC20Upgradeable.initPSIDividendTracker(IDividendTracker)._dividendTracker (AstroBirdsV2.sol#912) is not in mixedCase
Parameter ERC20Upgradeable.calculateLiquidityFee(uint256)._amount (AstroBirdsV2.sol#982) is not in mixedCase
Parameter ERC20Upgradeable.calculatePSIFee(uint256)._amount (AstroBirdsV2.sol#986) is not in mixedCase
Parameter ERC20Upgradeable.calculateMarketingFee(uint256)._amount (AstroBirdsV2.sol#990) is not in mixedCase
Parameter ERC20Upgradeable.calculateTeamFee(uint256)._amount (AstroBirdsV2.sol#994) is not in mixedCase
Parameter ERC20Upgradeable.calculateBuybackFee(uint256)._amount (AstroBirdsV2.sol#998) is not in mixedCase
Parameter ERC20Upgradeable.setPSIFee(uint256).PSIfee_ (AstroBirdsV2.sol#1002) is not in mixedCase
Parameter ERC20Upgradeable.setLiquidityFee(uint256).LPfee_ (AstroBirdsV2.sol#1007) is not in mixedCase
Parameter ERC20Upgradeable.setBuybackFee(uint256).BBFee_ (AstroBirdsV2.sol#1012) is not in mixedCase
Parameter ERC20Upgradeable.setMarketingFee(uint256).Mfee_ (AstroBirdsV2.sol#1017) is not in mixedCase
Parameter ERC20Upgradeable.setTeamFee(uint256).Tfee_ (AstroBirdsV2.sol#1022) is not in mixedCase
Parameter ERC20Upgradeable.changePSIAddress(address).PSIAddress_ (AstroBirdsV2.sol#1046) is not in mixedCase
Parameter ERC20Upgradeable.changeSellLimit(uint256)._sellLimit (AstroBirdsV2.sol#1056) is not in mixedCase
Parameter ERC20Upgradeable.changeMaxtx(uint256)._maxtx (AstroBirdsV2.sol#1060) is not in mixedCase
Parameter ERC20Upgradeable.addNewRouter(address,bool)._router (AstroBirdsV2.sol#1075) is not in mixedCase
Parameter ERC20Upgradeable.setSwapAndLiquifyEnabled(bool)._enabled (AstroBirdsV2.sol#1264) is not in mixedCase
Variable ERC20Upgradeable._Owner (AstroBirdsV2.sol#803) is not in mixedCase
Variable ERC20Upgradeable._previousOwner (AstroBirdsV2.sol#804) is not in mixedCase
Variable ERC20Upgradeable._psiAddress (AstroBirdsV2.sol#805) is not in mixedCase
Variable ERC20Upgradeable._buybackAddress (AstroBirdsV2.sol#806) is not in mixedCase
Variable ERC20Upgradeable._liquidityPoolAddress (AstroBirdsV2.sol#807) is not in mixedCase
Variable ERC20Upgradeable._marketingAddress (AstroBirdsV2.sol#808) is not in mixedCase
```

```
Variable ERC20Upgradeable._teamAddress (AstroBirdsV2.sol#809) is not in mixedCase
Variable ERC20Upgradeable._maxTxAmount (AstroBirdsV2.sol#818) is not in mixedCase
Parameter AstroBirdsV2.initialize(string,string,address,address,address,address,address)._name (AstroBirdsV2.sol#1568) is not in
 mixedCase
Parameter AstroBirdsV2.initialize(string,string,address,address,address,address,address)._symbol (AstroBirdsV2.sol#1569) is not
in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Reentrancy in ERC20Upgradeable._transfer(address,address,uint256) (AstroBirdsV2.sol#1298-1342):
        External calls:
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (AstroBirdsV2.sol#1332)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        State variables written after the call(s):
        - _simpleTransfer(sender,recipient,amount) (AstroBirdsV2.sol#1335)
                - _balances[sender] = senderBalance - amount (AstroBirdsV2.sol#1366)
                - _balances[recipient] += amount (AstroBirdsV2.sol#1367)
        Event emitted after the call(s):
        - ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gasForProcessing,tx.origin) (AstroBirdsV2.sol#1339)
        - Transfer(sender,recipient,amount) (AstroBirdsV2.sol#1368)
                - _simpleTransfer(sender,recipient,amount) (AstroBirdsV2.sol#1335)
```

```
Reentrancy in ERC20Upgradeable.swapAndLiquify(uint256) (AstroBirdsV2.sol#1378-1399):
        External calls:
        - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
        - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
        External calls sending eth:
        - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
        - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
        - swapAndSendDividends((feeBalance * psiFee) / totalFees) (AstroBirdsV2.sol#1392)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
        State variables written after the call(s):
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - _allowances[owner][spender] = amount (AstroBirdsV2.sol#1534)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (AstroBirdsV2.sol#1535)
                - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
Reentrancy in ERC20Upgradeable.swapAndLiquify(uint256) (AstroBirdsV2.sol#1378-1399):
        External calls:
        - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
        - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
        - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        External calls sending eth:
        - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
        - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
        - swapAndSendDividends((feeBalance * psiFee) / totalFees) (AstroBirdsV2.sol#1392)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
        - addLiquidity(forLiquidity / 2,(swappedBalance - feeBalance)) (AstroBirdsV2.sol#1394)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
        - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        Event emitted after the call(s):
        - SwapAndLiquify(contractTokenBalance,swappedBalance,forLiquidity / 2) (AstroBirdsV2.sol#1398)
Reentrancy in ERC20Upgradeable.transferFrom(address,address,uint256) (AstroBirdsV2.sol#1211-1224):
```

```
        External calls:
        - _transfer(sender,recipient,amount) (AstroBirdsV2.sol#1217)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (AstroBirdsV2.sol#1217)
                - pancakeRouter.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,_liquidityPoolAddress,block.time
stamp) (AstroBirdsV2.sol#1431-1438)
                - pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethAmount}(0,path,recipient,block.time
stamp) (AstroBirdsV2.sol#1457-1462)
                - address(_marketingAddress).transfer((feeBalance * marketingFee) / totalFees) (AstroBirdsV2.sol#1389)
                - address(_teamAddress).transfer((feeBalance * teamFee) / totalFees) (AstroBirdsV2.sol#1390)
                - address(_buybackAddress).transfer(address(this).balance - initialBalance) (AstroBirdsV2.sol#1396)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),currentAllowance - amount) (AstroBirdsV2.sol#1222)
                - _allowances[owner][spender] = amount (AstroBirdsV2.sol#1534)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (AstroBirdsV2.sol#1535)
                - _approve(sender,_msgSender(),currentAllowance - amount) (AstroBirdsV2.sol#1222)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (AstroBir
dsV2.sol#351) is too similar to IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).a
mountBDesired (AstroBirdsV2.sol#352)
Variable ERC20Upgradeable._buybackAddress (AstroBirdsV2.sol#806) is too similar to ERC20Upgradeable.setBuybackAddress(address).b
uybackAddress_ (AstroBirdsV2.sol#1031)
Variable ERC20Upgradeable._buybackAddress (AstroBirdsV2.sol#806) is too similar to ERC20Upgradeable._ERC20_init(string,string,ad
dress,address,address,address,address).buybackAddress_ (AstroBirdsV2.sol#884)
Variable ERC20Upgradeable._marketingAddress (AstroBirdsV2.sol#808) is too similar to ERC20Upgradeable.changeMarketingAddress(add
ress).marketingAddress_ (AstroBirdsV2.sol#1036)
Variable ERC20Upgradeable._marketingAddress (AstroBirdsV2.sol#808) is too similar to ERC20Upgradeable._ERC20_init(string,string,
address,address,address,address,address).marketingAddress_ (AstroBirdsV2.sol#881)
Variable ERC20Upgradeable.changePSIAddress(address).PSIAddress_ (AstroBirdsV2.sol#1046) is too similar to ERC20Upgradeable._psiA
ddress (AstroBirdsV2.sol#805)
Variable ERC20Upgradeable._psiAddress (AstroBirdsV2.sol#805) is too similar to ERC20Upgradeable._ERC20_init(string,string,addres
s,address,address,address,address).psiAddress_ (AstroBirdsV2.sol#883)
```

```
Variable ERC20Upgradeable._teamAddress (AstroBirdsV2.sol#809) is too similar to ERC20Upgradeable.changeTeamAddress(address).team
Address_ (AstroBirdsV2.sol#1041)
Variable ERC20Upgradeable._teamAddress (AstroBirdsV2.sol#809) is too similar to ERC20Upgradeable._ERC20_init(string,string,addre
ss,address,address,address,address).teamAddress_ (AstroBirdsV2.sol#882)
Variable ERC20Upgradeable._buybackAddress (AstroBirdsV2.sol#806) is too similar to AstroBirdsV2.initialize(string,string,address
,address,address,address,address).buybackAddress_ (AstroBirdsV2.sol#1573)
Variable ERC20Upgradeable._marketingAddress (AstroBirdsV2.sol#808) is too similar to AstroBirdsV2.initialize(string,string,addre
ss,address,address,address,address).marketingAddress_ (AstroBirdsV2.sol#1570)
Variable ERC20Upgradeable._psiAddress (AstroBirdsV2.sol#805) is too similar to AstroBirdsV2.initialize(string,string,address,add
ress,address,address,address).psiAddress_ (AstroBirdsV2.sol#1572)
Variable ERC20Upgradeable._teamAddress (AstroBirdsV2.sol#809) is too similar to AstroBirdsV2.initialize(string,string,address,ad
dress,address,address,address).teamAddress_ (AstroBirdsV2.sol#1571)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
ERC20Upgradeable._ERC20_init(string,string,address,address,address,address,address) (AstroBirdsV2.sol#878-910) uses literals wit
h too many digits:
        - _maxTxAmount = 5000000 * 10 ** 18 (AstroBirdsV2.sol#898)
ERC20Upgradeable.initPSIDividendTracker(IDividendTracker) (AstroBirdsV2.sol#912-934) uses literals with too many digits:
        - dividendTracker.excludeFromDividends(address(0x000000000000000000000000000000000000dEaD)) (AstroBirdsV2.sol#919)
ERC20Upgradeable.initPSIDividendTracker(IDividendTracker) (AstroBirdsV2.sol#912-934) uses literals with too many digits:
        - gasForProcessing = 300000 (AstroBirdsV2.sol#930)
ERC20Upgradeable.updateGasForProcessing(uint256) (AstroBirdsV2.sol#1108-1115) uses literals with too many digits:
        - require(bool,string)(newValue >= 200000 && newValue <= 500000,AstroBirdz: gasForProcessing must be between 200,000 and
 500,000) (AstroBirdsV2.sol#1109-1112)
ERC20Upgradeable._burn(address,uint256) (AstroBirdsV2.sol#1500-1511) uses literals with too many digits:
        - _beforeTokenTransfer(account,address(0x000000000000000000000000000000000000dEaD),amount) (AstroBirdsV2.sol#1504)
ERC20Upgradeable._burn(address,uint256) (AstroBirdsV2.sol#1500-1511) uses literals with too many digits:
        - Transfer(account,address(0x000000000000000000000000000000000000dEaD),amount) (AstroBirdsV2.sol#1510)
AstroBirdsV2.initialize(string,string,address,address,address,address,address) (AstroBirdsV2.sol#1567-1587) uses literals with t
oo many digits:
        - _mint(_msgSender(),47000000000000000000000000) (AstroBirdsV2.sol#1586)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ContextUpgradeable.__gap (AstroBirdsV2.sol#788) is never used in AstroBirdsV2 (AstroBirdsV2.sol#1566-1589)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
name() should be declared external:
```

```
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (AstroBirdsV2.sol#947-949)
totalSupply() should be declared external:
        - ERC20Upgradeable.totalSupply() (AstroBirdsV2.sol#971-973)
setPSIFee(uint256) should be declared external:
        - ERC20Upgradeable.setPSIFee(uint256) (AstroBirdsV2.sol#1002-1005)
setLiquidityFee(uint256) should be declared external:
        - ERC20Upgradeable.setLiquidityFee(uint256) (AstroBirdsV2.sol#1007-1010)
setBuybackFee(uint256) should be declared external:
        - ERC20Upgradeable.setBuybackFee(uint256) (AstroBirdsV2.sol#1012-1015)
setMarketingFee(uint256) should be declared external:
        - ERC20Upgradeable.setMarketingFee(uint256) (AstroBirdsV2.sol#1017-1020)
setTeamFee(uint256) should be declared external:
        - ERC20Upgradeable.setTeamFee(uint256) (AstroBirdsV2.sol#1022-1025)
setBuybackAddress(address) should be declared external:
        - ERC20Upgradeable.setBuybackAddress(address) (AstroBirdsV2.sol#1031-1034)
changeMarketingAddress(address) should be declared external:
        - ERC20Upgradeable.changeMarketingAddress(address) (AstroBirdsV2.sol#1036-1039)
changeTeamAddress(address) should be declared external:
        - ERC20Upgradeable.changeTeamAddress(address) (AstroBirdsV2.sol#1041-1044)
changePSIAddress(address) should be declared external:
        - ERC20Upgradeable.changePSIAddress(address) (AstroBirdsV2.sol#1046-1049)
changeLiquidityAddress(address) should be declared external:
        - ERC20Upgradeable.changeLiquidityAddress(address) (AstroBirdsV2.sol#1051-1054)
changeSellLimit(uint256) should be declared external:
        - ERC20Upgradeable.changeSellLimit(uint256) (AstroBirdsV2.sol#1056-1058)
changeMaxtx(uint256) should be declared external:
        - ERC20Upgradeable.changeMaxtx(uint256) (AstroBirdsV2.sol#1060-1062)
removeExcludedAddress(address) should be declared external:
        - ERC20Upgradeable.removeExcludedAddress(address) (AstroBirdsV2.sol#1067-1069)
transferOwnership(address) should be declared external:
        - ERC20Upgradeable.transferOwnership(address) (AstroBirdsV2.sol#1121-1125)
getUnlockTime() should be declared external:
        - ERC20Upgradeable.getUnlockTime() (AstroBirdsV2.sol#1127-1129)
lock(uint256) should be declared external:
        - ERC20Upgradeable.lock(uint256) (AstroBirdsV2.sol#1132-1137)
unlock() should be declared external:
```

```
unlock() should be declared external:
        - ERC20Upgradeable.unlock() (AstroBirdsV2.sol#1140-1145)
multiTransfer(address[],uint256[]) should be declared external:
        - ERC20Upgradeable.multiTransfer(address[],uint256[]) (AstroBirdsV2.sol#1147-1153)
allowance(address,address) should be declared external:
        - ERC20Upgradeable.allowance(address,address) (AstroBirdsV2.sol#1183-1185)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (AstroBirdsV2.sol#1194-1197)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (AstroBirdsV2.sol#1211-1224)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (AstroBirdsV2.sol#1238-1241)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (AstroBirdsV2.sol#1257-1262)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - ERC20Upgradeable.setSwapAndLiquifyEnabled(bool) (AstroBirdsV2.sol#1264-1267)
toggleTrading() should be declared external:
        - ERC20Upgradeable.toggleTrading() (AstroBirdsV2.sol#1401-1403)
togglePaused() should be declared external:
        - ERC20Upgradeable.togglePaused() (AstroBirdsV2.sol#1404-1406)
initialize(string,string,address,address,address,address,address) should be declared external:
        - AstroBirdsV2.initialize(string,string,address,address,address,address,address) (AstroBirdsV2.sol#1567-1587)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:AstroBirdsV2.sol analyzed (15 contracts with 75 detectors), 148 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> AstroBirdzDividendTracker.sol

```
INFO:Detectors:
ERC20TokenRecover.recoverETH(uint256) (AstroBirdzDividendTracker.sol#1247-1250) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = owner().call{value: amount}() (AstroBirdzDividendTracker.sol#1248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in DividendPayingToken._withdrawDividendOfUser(address) (AstroBirdzDividendTracker.sol#1154-1170):
        External calls:
        - success = IERC20(dividendToken).transfer(user,_withdrawableDividend) (AstroBirdzDividendTracker.sol#1159)
        State variables written after the call(s):
        - withdrawnDividends[user] = withdrawnDividends[user] - _withdrawableDividend (AstroBirdzDividendTracker.sol#1162)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
DividendPayingToken.constructor(string,string,address,address)._name (AstroBirdzDividendTracker.sol#1130) shadows:
        - ERC20._name (AstroBirdzDividendTracker.sol#777) (state variable)
DividendPayingToken.constructor(string,string,address,address)._symbol (AstroBirdzDividendTracker.sol#1131) shadows:
        - ERC20._symbol (AstroBirdzDividendTracker.sol#778) (state variable)
DividendPayingToken.dividendOf(address)._owner (AstroBirdzDividendTracker.sol#1172) shadows:
        - Ownable._owner (AstroBirdzDividendTracker.sol#714) (state variable)
DividendPayingToken.withdrawableDividendOf(address)._owner (AstroBirdzDividendTracker.sol#1176) shadows:
        - Ownable._owner (AstroBirdzDividendTracker.sol#714) (state variable)
DividendPayingToken.withdrawnDividendOf(address)._owner (AstroBirdzDividendTracker.sol#1180) shadows:
        - Ownable._owner (AstroBirdzDividendTracker.sol#714) (state variable)
DividendPayingToken.accumulativeDividendOf(address)._owner (AstroBirdzDividendTracker.sol#1184) shadows:
        - Ownable._owner (AstroBirdzDividendTracker.sol#714) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in AstroBirdzDividendTracker.processAccount(address,bool) (AstroBirdzDividendTracker.sol#1515-1530):
        External calls:
        - amount = _withdrawDividendOfUser(account) (AstroBirdzDividendTracker.sol#1521)
                - success = IERC20(dividendToken).transfer(user,_withdrawableDividend) (AstroBirdzDividendTracker.sol#1159)
        State variables written after the call(s):
        - lastClaimTimes[account] = block.timestamp (AstroBirdzDividendTracker.sol#1524)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in AstroBirdzDividendTracker.processAccount(address,bool) (AstroBirdzDividendTracker.sol#1515-1530):
```

```
        External calls:
        - amount = _withdrawDividendOfUser(account) (AstroBirdzDividendTracker.sol#1521)
                - success = IERC20(dividendToken).transfer(user,_withdrawableDividend) (AstroBirdzDividendTracker.sol#1159)
        Event emitted after the call(s):
        - Claim(account,amount,automatic) (AstroBirdzDividendTracker.sol#1525)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
AstroBirdzDividendTracker.getAccount(address) (AstroBirdzDividendTracker.sol#1343-1383) uses timestamp for comparisons
        Dangerous comparisons:
        - nextClaimTime > block.timestamp (AstroBirdzDividendTracker.sol#1382)
AstroBirdzDividendTracker.canAutoClaim(uint256) (AstroBirdzDividendTracker.sol#1405-1411) uses timestamp for comparisons
        Dangerous comparisons:
        - lastClaimTime > block.timestamp (AstroBirdzDividendTracker.sol#1406)
        - (block.timestamp - lastClaimTime) >= claimWait (AstroBirdzDividendTracker.sol#1410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (AstroBirdzDividendTracker.sol#423-443) uses assembly
        - INLINE ASM (AstroBirdzDividendTracker.sol#435-438)
AstroBirdzDividendTracker.bytesToAddress(bytes) (AstroBirdzDividendTracker.sol#1427-1430) uses assembly
        - INLINE ASM (AstroBirdzDividendTracker.sol#1429)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (AstroBirdzDividendTracker.sol#307-309) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (AstroBirdzDividendTracker.sol#336-342) is never used and should be removed
Address.functionDelegateCall(address,bytes) (AstroBirdzDividendTracker.sol#396-398) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (AstroBirdzDividendTracker.sol#406-415) is never used and should be removed
Address.functionStaticCall(address,bytes) (AstroBirdzDividendTracker.sol#369-371) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (AstroBirdzDividendTracker.sol#379-388) is never used and should be removed
Address.sendValue(address,uint256) (AstroBirdzDividendTracker.sol#282-287) is never used and should be removed
Context._msgData() (AstroBirdzDividendTracker.sol#708-710) is never used and should be removed
DividendPayingToken._transfer(address,address,uint256) (AstroBirdzDividendTracker.sol#1190-1200) is never used and should be rem
oved
ERC20._transfer(address,address,uint256) (AstroBirdzDividendTracker.sol#964-984) is never used and should be removed
IterableMapping.get(IterableMapping.Map,address) (AstroBirdzDividendTracker.sol#13-15) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (AstroBirdzDividendTracker.sol#473-486) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (AstroBirdzDividendTracker.sol#497-508) is never used and should be remo
ved
```

```
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (AstroBirdzDividendTracker.sol#488-495) is never used and should be remo
ved
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (AstroBirdzDividendTracker.sol#457-464) is never used and should be r
emoved
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (AstroBirdzDividendTracker.sol#3) necessitates a version too recent to be trusted. Consider deploying with
0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (AstroBirdzDividendTracker.sol#282-287):
        - (success) = recipient.call{value: amount}() (AstroBirdzDividendTracker.sol#285)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (AstroBirdzDividendTracker.sol#350-361):
        - (success,returndata) = target.call{value: value}(data) (AstroBirdzDividendTracker.sol#359)
Low level call in Address.functionStaticCall(address,bytes,string) (AstroBirdzDividendTracker.sol#379-388):
        - (success,returndata) = target.staticcall(data) (AstroBirdzDividendTracker.sol#386)
Low level call in Address.functionDelegateCall(address,bytes,string) (AstroBirdzDividendTracker.sol#406-415):
        - (success,returndata) = target.delegatecall(data) (AstroBirdzDividendTracker.sol#413)
Low level call in ERC20TokenRecover.recoverETH(uint256) (AstroBirdzDividendTracker.sol#1247-1250):
        - (sent) = owner().call{value: amount}() (AstroBirdzDividendTracker.sol#1248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter DividendPayingToken.dividendOf(address)._owner (AstroBirdzDividendTracker.sol#1172) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (AstroBirdzDividendTracker.sol#1176) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (AstroBirdzDividendTracker.sol#1180) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (AstroBirdzDividendTracker.sol#1184) is not in mixedCase
Constant DividendPayingToken.magnitude (AstroBirdzDividendTracker.sol#1109) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter AstroBirdzDividendTracker.getAccount(address)._account (AstroBirdzDividendTracker.sol#1343) is not in mixedCase
Parameter AstroBirdzDividendTracker.ensureBalance(bool)._process (AstroBirdzDividendTracker.sol#1413) is not in mixedCase
Parameter AstroBirdzDividendTracker.ensureBalanceForUsers(bytes,bool)._process (AstroBirdzDividendTracker.sol#1417) is not in mi
xedCase
Parameter AstroBirdzDividendTracker.ensureBalanceForUser(address,bool)._process (AstroBirdzDividendTracker.sol#1432) is not in m
ixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (AstroBirdzDividendTracker.sol#1155) is too
```

```
similar to AstroBirdzDividendTracker.getAccount(address).withdrawableDividends (AstroBirdzDividendTracker.sol#1351)
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (AstroBirdzDividendTracker.sol#1155) is too
similar to IDividendTracker.getAccount(address).withdrawableDividends (AstroBirdzDividendTracker.sol#663)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (AstroBirdzDividendTracker.sol#747-749)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (AstroBirdzDividendTracker.sol#755-758)
name() should be declared external:
        - ERC20.name() (AstroBirdzDividendTracker.sol#797-799)
symbol() should be declared external:
        - ERC20.symbol() (AstroBirdzDividendTracker.sol#805-807)
decimals() should be declared external:
        - ERC20.decimals() (AstroBirdzDividendTracker.sol#822-824)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (AstroBirdzDividendTracker.sol#848-851)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (AstroBirdzDividendTracker.sol#856-858)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (AstroBirdzDividendTracker.sol#870-873)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (AstroBirdzDividendTracker.sol#891-907)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (AstroBirdzDividendTracker.sol#921-924)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (AstroBirdzDividendTracker.sol#940-948)
```

```
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (AstroBirdzDividendTracker.sol#940-948)
distributeDividends(uint256) should be declared external:
        - DividendPayingToken.distributeDividends(uint256) (AstroBirdzDividendTracker.sol#1139-1148)
withdrawDividend() should be declared external:
        - AstroBirdzDividendTracker.withdrawDividend() (AstroBirdzDividendTracker.sol#1295-1300)
        - DividendPayingToken.withdrawDividend() (AstroBirdzDividendTracker.sol#1150-1152)
dividendOf(address) should be declared external:
        - DividendPayingToken.dividendOf(address) (AstroBirdzDividendTracker.sol#1172-1174)
withdrawnDividendOf(address) should be declared external:
        - DividendPayingToken.withdrawnDividendOf(address) (AstroBirdzDividendTracker.sol#1180-1182)
recoverETH(uint256) should be declared external:
        - ERC20TokenRecover.recoverETH(uint256) (AstroBirdzDividendTracker.sol#1247-1250)
getOwner() should be declared external:
        - AstroBirdzDividendTracker.getOwner() (AstroBirdzDividendTracker.sol#1274-1276)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:AstroBirdzDividendTracker.sol analyzed (16 contracts with 75 detectors), 64 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity Static Analysis

AstroBirdsV2.sol

Security

Transaction origin:

Use of tx.origin: "tx.origin" is useful only in very exceptional cases. If you use it for authentication, you usually want to replace it by "msg.sender", because otherwise any contract you call can act on your behalf.
more
Pos: 1157:90:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in ERC20Upgradeable.swapTokensForEth(uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1408:4:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1461:12:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 218:50:

Gas & Economy

Gas costs:

Gas requirement of function AstroBirdsV2.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 939:4:

Gas costs:

Gas requirement of function AstroBirdsV2.initialize is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1567:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.

more

Pos: 1150:8:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 299:4:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type

more

Pos: 964:4:

Miscellaneous

Constant/View/Pure functions:

IERC20Upgradeable.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 34:4:

Constant/View/Pure functions:

ERC20Upgradeable._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 1563:4:

Similar variable names:

AstroBirdsV2.initialize(string,string,address payable,address payable,address,address,address) : Variables have very similar names "_psiAddress" and "psiAddress_". Note: Modifiers are currently not considered by this static analysis.

Pos: 1581:12:

Similar variable names:

AstroBirdsV2.initialize(string,string,address payable,address payable,address,address,address) : Variables have very similar names "_buybackAddress" and "buybackAddress_". Note: Modifiers are currently not considered by this static analysis.

Pos: 1582:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1532:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1398:66:

AstroBirdzDividendTracker.sol

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 1429:8:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1406:28:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1410:16:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 1248:24:

Gas & Economy

Gas costs:

Gas requirement of function AstroBirdzDividendTracker.name is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 797:4:

Gas costs:

Gas requirement of function AstroBirdzDividendTracker.recoverERC20 is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1278:4:

Gas costs:

Gas requirement of function ERC20TokenRecover.recoverERC20 is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1278:4:

Gas costs:

Gas requirement of function AstroBirdzDividendTracker.processAccount is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1515:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1491:12:

Miscellaneous

Constant/View/Pure functions:

IterableMapping.set(struct IterableMapping.Map,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 32:4:

Constant/View/Pure functions:

AstroBirdzDividendTracker.recoverERC20(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1278:4:

Constant/View/Pure functions:

AstroBirdzDividendTracker.bytesToAddress(bytes) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1427:4:

Similar variable names:

AstroBirdzDividendTracker.getAccount(address) : Variables have very similar names "lastClaimTime" and "lastClaimTimes". Note: Modifiers are currently not considered by this static analysis.
Pos: 1380:44:

Similar variable names:

AstroBirdzDividendTracker.canAutoClaim(uint256) : Variables have very similar names "lastClaimTime" and "lastClaimTimes". Note: Modifiers are currently not considered by this static analysis.
Pos: 1406:12:

Similar variable names:

AstroBirdzDividendTracker.canAutoClaim(uint256) : Variables have very similar names "lastClaimTime" and "lastClaimTimes". Note: Modifiers are currently not considered by this static analysis.
Pos: 1410:34:

No return:

AstroBirdzDividendTracker.bytesToAddress(bytes): Defines a return type but never explicitly returns a value.
Pos: 1427:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1322:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1326:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1422:28:

Solhint Linter

AstroBirdsV2.sol

```
AstroBirdsV2.sol:1222:18: Error: Parse error: missing ';' at '{'
AstroBirdsV2.sol:1260:18: Error: Parse error: missing ';' at '{'
AstroBirdsV2.sol:1366:18: Error: Parse error: missing ';' at '{'
AstroBirdsV2.sol:1507:18: Error: Parse error: missing ';' at '{'
```

AstroBirdzDividendTracker.sol

```
AstroBirdzDividendTracker.sol:502:18: Error: Parse error: missing ';'
at '{'
AstroBirdzDividendTracker.sol:899:22: Error: Parse error: missing ';'
at '{'
AstroBirdzDividendTracker.sol:943:18: Error: Parse error: missing ';'
at '{'
AstroBirdzDividendTracker.sol:976:18: Error: Parse error: missing ';'
at '{'
AstroBirdzDividendTracker.sol:1025:18: Error: Parse error: missing
';' at '{'
```

Software analysis result:

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

Email: audit@EtherAuthority.io