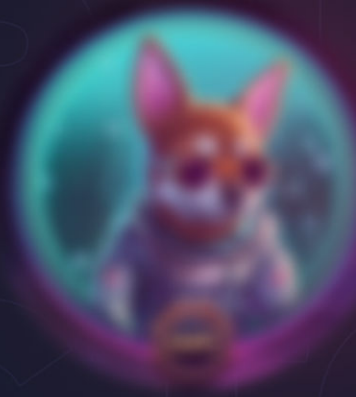
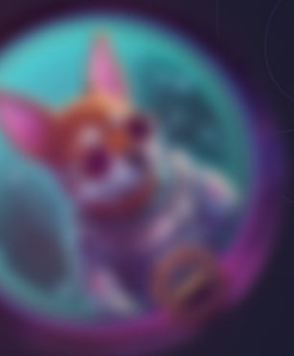
INSPECTOR LOVELY



SMART CONTRACT

SECURITY AUDIT

ASTROPUP COIN

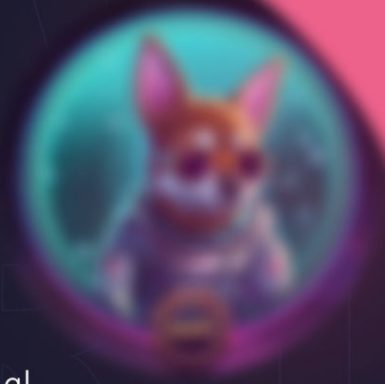


inspector.lovely.finance

INSPECTOR LOVELY

TABLE OF CONTENTS

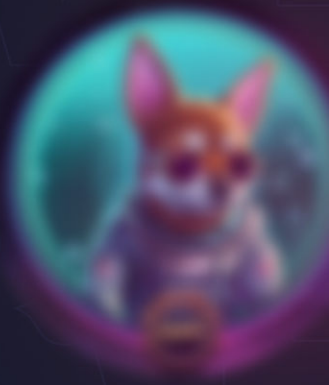
INSPECTOR LOVELY



DISCLAIMER

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws of the project's smart contract. Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it. Before making any judgments, you have to conduct your own independent research. We will discuss this in more depth in the following disclaimer - please read it fully. DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. Scan and verify report's presence in the GitHub repository by a qr-code on the title page. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Inspector Lovely and its affiliates shall not be held responsible to you or anyone else, nor shall Inspector Lovely provide any guarantee or representation to any person with regard to the accuracy or integrity of the report. Without any terms, warranties or other conditions other than as set forth in that exclusion and Inspector Lovely excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills). The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Inspector Lovely disclaims all responsibility and responsibilities and no claim against Inspector Lovely is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent). Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

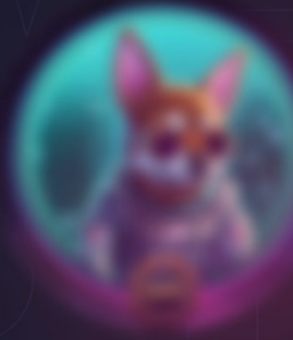
INSPECTOR LOVELY



AUDIT SCOPE

Name	Code Review and Security Analysis Report for AstroPup Coin Smart Contract
Platform	Ethereum/Solidity
File	AstroPup.sol
File MD5 Hash	143C43BE0550CD5E639572F2FF8C4D77
Smart Contract Code	0×96eAfff5BedF18566B18fCe71C2323b69C795623
Audit Date	May 9th, 2023
Revision Date	May 10th, 2023

INSPECTOR LOVELY



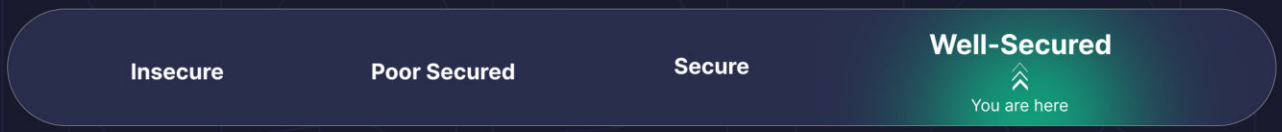
PROPOSED SMART CONTRACT FEATURES

Claimed Feature Detail	Our Observation
Tokenomics: • Name: AstroPup Coin • Symbol: ASPC • Decimals: 18 • The burn rate is set to 1% • Total Supply: 69 Billion	Validated
Owner Specification: • Triggers stopped state • Returns to normal state • Set a burn percentage • Current owner can transfer ownership of the contract • Deleting ownership will leave the contract without an owner, removing any owner-only functionality	The ownership is renounced and thus the smart contract is 100% decentralized

INSPECTOR LOVELY

AUDIT SUMMARY

According to the standard audit assessment, Customer`s solidity based smart contracts are **“Well Secured”**. The project owner also completed the KYC with Ether Authority which can be verified at here.



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 2 low, and some very low level issues.

We confirm that All severity issues are solved in the revised smart contract.

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.

INSPECTOR LOVELY

KEY TECHNICAL METRICS

MAIN CATEGORY	SUBCATEGORY	RESULT
Contract Programming	Solidity version is not specified	Passed
	Solidity version is too old	Passed
	Integer overflow/underflow	Passed
	Function input parameters lack check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	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	Passed
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 is not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

INSPECTOR LOVELY

CODE QUALITY

This audit scope has 1 smart contract. Smart contract contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in the AstroPup Coin 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 AstroPup Coin.

The AstroPup Coin team has not provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are well commented on in the smart contracts. Ethereum's NatSpec commenting style is used, which is a good thing.

DOCUMENTATION

We were given an AstroPup Coin smart contract code in the form of a file. The hash of that code is mentioned above in the table.

As mentioned above, code parts are well commented. So it is 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.

Another source of information was its official website: https://www.astropupcoin.com which provided rich information about the project architecture and tokenomics.

USE OF DEPENDENCIES

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

Apart from libraries, its functions are not used in external smart contract calls.

INSPECTOR LOVELY

AS-IS OVERVIEW

Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
1	constructor	write	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	balanceOf	read	Passed	No Issue
7	transfer	write	Passed	No Issue
8	allowance	read	Passed	No Issue
9	approve	write	Passed	No Issue
10	transferFrom	write	Passed	No Issue
11	increaseAllowance	write	Passed	No Issue
12	decreaseAllowance	write	Passed	No Issue
13	_transfer	internal	Passed	No Issue
14	_mint	internal	Passed	No Issue
15	_burn	internal	Passed	No Issue
16	_approve	internal	Passed	No Issue
17	_spendAllowance	internal	Passed	No Issue
18	_beforeTokenTransfer	internal	Passed	No Issue
19	_afterTokenTransfer	internal	Passed	No Issue
20	whenNotPaused	modifier	Passed	No Issue
21	whenPaused	modifier	Passed	No Issue
22	paused	read	Passed	No Issue
23	_requireNotPaused	internal	Passed	No Issue
24	_requirePaused	internal	Passed	No Issue
25	_pause	internal	Passed	No Issue
26	_unpause	internal	Passed	No Issue
27	onlyOwner	modifier	Passed	No Issue
28	owner	read	Passed	No Issue
29	_checkOwner	internal	Passed	No Issue

AS-IS OVERVIEW

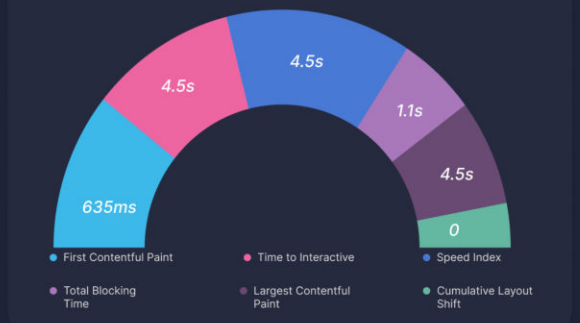
Functions

SL.	FUNCTIONS	TYPE	OBSERVATION	CONCLUSION
30	renounceOwnership	write	access only Owner	No Issue
31	transferOwnership	write	access only Owner	No Issue
32	_transferOwnership	internal	Passed	No Issue
33	burn	write	Passed	No Issue
34	burnFrom	write	Passed	No Issue
35	pause	write	access only Owner	No Issue
36	unpause	write	access only Owner	No Issue
37	setBurnPercentage	write	Passed	Fixed
38	_beforeTokenTransfer	internal	Passed	No Issue
39	_transfer	internal	Passed	Fixed

INSPECTOR LOVELY

PROJECT WEBSITE PERFORMANCE AUDIT

Performance Metrics



Browser Timings

Redirect Duration **0ms**	Connection Duration **396ms**	Backend Duration **108ms**
Time to First Byte **504ms**	First Paint **636ms**	DOM Interactive Time **653ms**
DOM Content Loaded **667ms**	Onload Time **978ms**	Fully Loaded Time **4.9s**

Grade

D	Performance 40%	Structure 90%

Web Vitals

LCP	TBT	CLS
4.5s	1.1ms	0

Top Issues

IMPECT	AUDIT	
High	Avoid enormous network payloads (LCP)	Total size was 3.55MB

URL	SIZE
https://cdn-static-e.dora.run/dora_runner/main.dart.a7ab3d8f5494e9ac.js	1.11MB
https://cdn-imgs.dora.run/design/HLK0tRLZN7vG6gLuGkicun.png/w/2048/h/2048/format/webp?project=563353	854KB
https://cdn-imgs.dora.run/design/lLq1pYBOb1SF7KP7BP4BgQ.png/w/2048/h/2048/format/webp?project=563353	369KB
https://cdn-static-e.dora.run/dora_runner/loading.6330c237a27f80f1.gif	250KB
https://cdn-imgs.dora.run/design/ENx2YaVP5zjKg6QFC1iXXJ.png/w/2048/h/2048/format/webp?project=563353	174KB
https://cdn-static-e.dora.run/fonts/v4/Inter-600.ttf	143KB
https://cdn-static-e.dora.run/fonts/v4/Inter-500.ttf	142KB
https://cdn-static-e.dora.run/fonts/v4/Inter-regular.ttf	132KB
https://cdn-imgs.dora.run/design/I7BMGDbFcqzGxbaNnQZtYB.png/format/webp?project=563353	104KB
https://www.googletagmanager.com/gtag/js?id=G-JPSGKY82PS	80.9KB

Level of Criticality

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 a significant impact on smart contract execution, e.g. public access to crucial
Med	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.

INSPECTOR LOVELY

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. Used Wrong symbol to compare:

```
44      function _transfer(
45          address from,
46          address to,
47          uint256 amount
48      ) internal override {
49          uint256 burnAmount = amount.mul(burnPercentage).div(100);
50          uint256 remainingAmount = amount.sub(burnAmount);
51          super._transfer(from, to, remainingAmount);
52          if (burnAmount ] 0) {
53              super._burn(from, burnAmount);
54          }
55      }
56  }
57
```

Used wrong symbol to compare greater than(>) in the if statement of _transfer function to compare burn amount.

Resolution: We suggest using the > symbol.

Status: This is fixed in the revised code.

(2) Not able to set Burn Percentage:

```
26      }
27
28      function setBurnPercentage(uint256 newBurnPercentage) public onlyOwner {
29          require(
30              newBurnPercentage [= 100,
31              "Burn percentage must be between 0 and 100."
32          );
33          burnPercentage = newBurnPercentage;
34      }
35
```

Used the wrong symbol to compare less than(<) in the required statement.

Resolution: We suggest using < symbol.

Status: This is fixed in the revised code.

VERY LOW / INFORMATIONAL / BEST PRACTICES:

1. SafeMath Library:

SafeMath Library is used in this contract code, but the compiler version is greater than or equal to 0.8.0, Then it will be not required to use, solidity automatically handles overflow/underflow.

Resolution: Remove the SafeMath library and use normal math operators, It will improve code size, and less gas consumption.

Status: This is fixed in the revised code.

2. Please use the latest compiler version when deploying contract:

This is not a severe issue, but we suggest using the latest compiler version at the time of contract deployment, which is 0.8.19 at the time of this audit. Using the latest compiler version is always recommended which prevents any compiler level issues

Resolution: We suggest using the latest compiler version 0.8.19.

Status: This is fixed in the revised code.

INSPECTOR LOVELY

CENTRALIZATION

The ownership of the token smart contract is renounced with this transaction hash. And thus the token smart contract has no centralized risk and the contract is 100% decentralized.

To make the smart contract 100% decentralized, we suggest renouncing ownership in the smart contract once its function is completed.

CONCLUSION

We were given a contract code in the form of a file and we have used all possible tests based on given objects as files. We have observed 2 low severity issues and 2 informational severity issues in the token smart contract. We confirm that all severity issues are solved in the revised code. **So, it's good to go for the mainnet deployment.**

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.

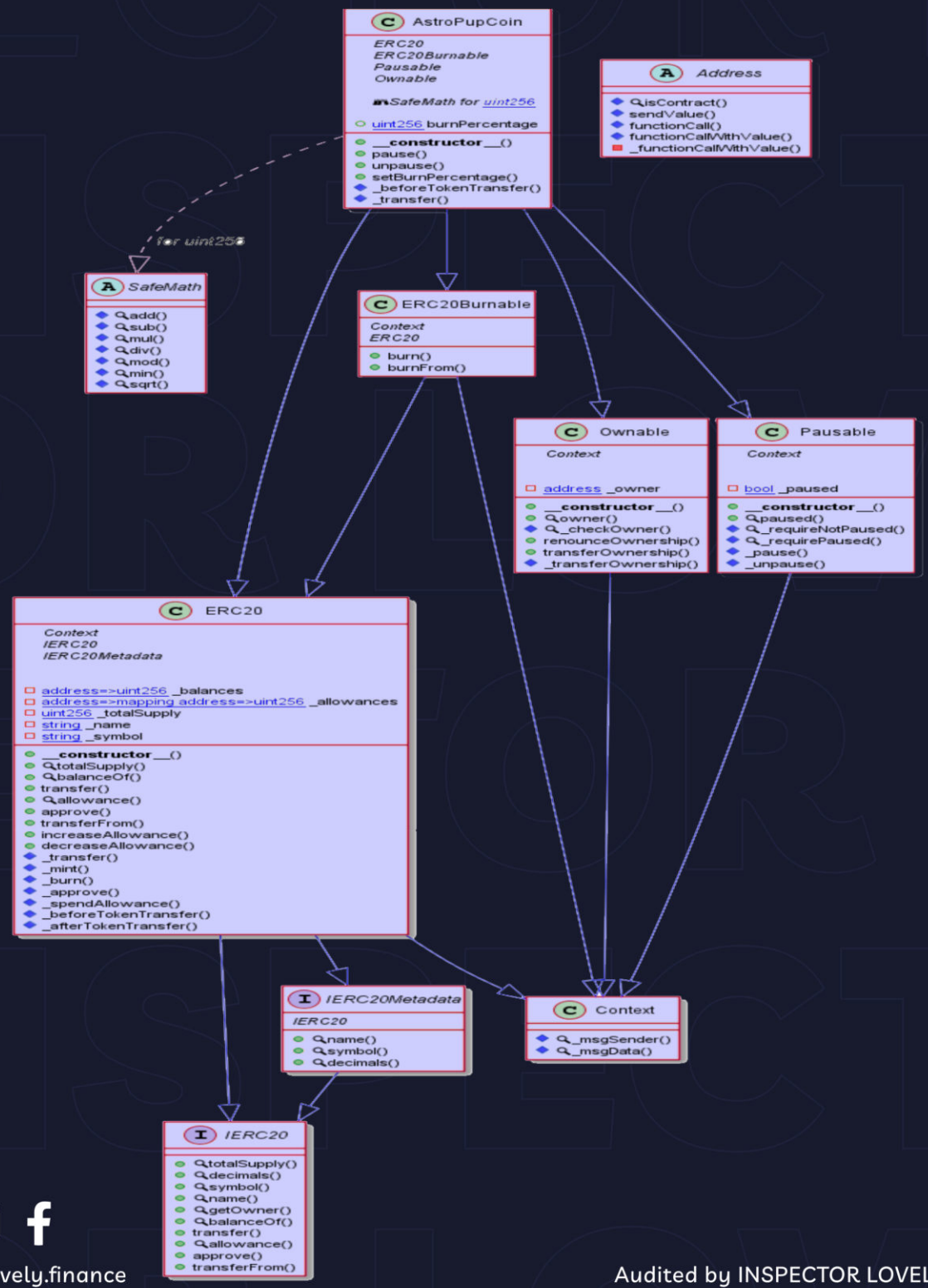
The audit report contains all found security vulnerabilities and other issues in the reviewed code.

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

INSPECTOR LOVELY

ADDENDUM

Code Flow Diagram - AstroPup Coin



INSPECTOR LOVELY

SECURITY ASSESSMENT REPORT

Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details, and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report with warnings and errors, allowing developers to quickly prototype and fix issues.

We did the analysis of the project altogether. Below are the results.

Slither Log >> AstroPup.sol

```
AstroPupCoin.setBurnPercentage(uint256) (AstroPup.sol#734-737) should emit an event for:
        - burnPercentage = newBurnPercentage (AstroPup.sol#736)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Address.isContract(address) (AstroPup.sol#87-94) uses assembly
        - INLINE ASM (AstroPup.sol#90-92)
Address._functionCallWithValue(address,bytes,uint256,string) (AstroPup.sol#133-155) uses assembly
        - INLINE ASM (AstroPup.sol#147-150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address._functionCallWithValue(address,bytes,uint256,string) (AstroPup.sol#133-155) is never used and should be removed
Address.functionCall(address,bytes) (AstroPup.sol#103-105) is never used and should be removed
Address.functionCall(address,bytes,string) (AstroPup.sol#107-113) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (AstroPup.sol#115-121) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (AstroPup.sol#123-131) is never used and should be removed
Address.isContract(address) (AstroPup.sol#87-94) is never used and should be removed
Address.sendValue(address,uint256) (AstroPup.sol#96-101) is never used and should be removed
Context._msgData() (AstroPup.sol#192-195) is never used and should be removed
ERC20._mint(address,uint256) (AstroPup.sol#419-432) is never used and should be removed
SafeMath.add(uint256,uint256) (AstroPup.sol#8-13) is never used and should be removed
SafeMath.min(uint256,uint256) (AstroPup.sol#69-71) is never used and should be removed
SafeMath.mod(uint256,uint256) (AstroPup.sol#56-58) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (AstroPup.sol#60-67) is never used and should be removed
SafeMath.sqrt(uint256) (AstroPup.sol#73-84) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (AstroPup.sol#2) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (AstroPup.sol#96-101):
        - (success) = recipient.call{value: amount}() (AstroPup.sol#99)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (AstroPup.sol#133-155):
        - (success,returndata) = target.call{value: weiValue}(data) (AstroPup.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Redundant expression "this (AstroPup.sol#193)" inContext (AstroPup.sol#187-196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

AstroPupCoin (AstroPup.sol#716-753) does not implement functions:
        - IERC20Metadata.decimals() (AstroPup.sol#212)
        - IERC20.getOwner() (AstroPup.sol#166)
        - IERC20Metadata.name() (AstroPup.sol#202)
        - IERC20Metadata.symbol() (AstroPup.sol#207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
AstroPup.sol analyzed (10 contracts with 84 detectors), 23 result(s) found
```

INSPECTOR LOVELY

SOLIDITY STATIC ANALYSIS

AstroPup.sol

Gas & Economy

Gas costs:

Gas requirement of function AstroPupCoin.pause 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: 20:0:

Gas costs:

Gas requirement of function AstroPupCoin.unpause 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: 24:0:

Miscellaneous

Constant/View/Pure functions:

AstroPupCoin._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: 33:0:

Constant/View/Pure functions:

AstroPupCoin._transfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 41:0:

INSPECTOR LOVELY

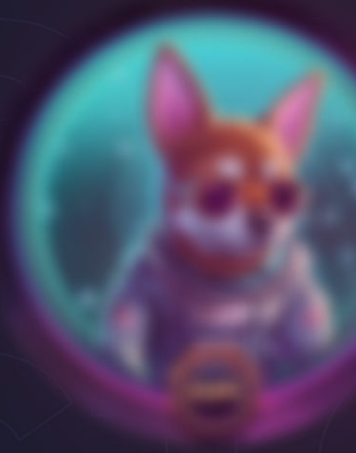
COMPLIANCE ANALYSIS

AstroPup.sol

```
AstroPup.sol:2:1: Error: Compiler version ^0.8.0 does not satisfy the r
semver requirement
AstroPup.sol:15:1: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0
```

Software analysis result:

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



INSPECTOR LOVELY

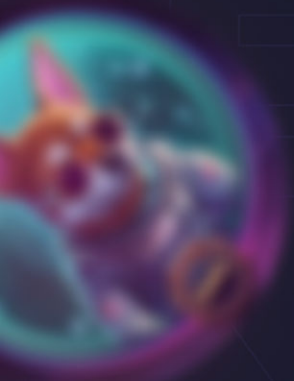
INSPECTOR LOVELY

INFO

Website: Inspector.lovely.finance

Telegram community: t.me/inspectorlovely

Twitter: twitter.com/InspectorLovely



inspector.lovely.finance