EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project:	PepeMints
Website:	https://pepemints.vip
Platform:	BNB Network
Language:	Solidity
Date:	April 30th, 2023

Table of contents

`

Introduction

EtherAuthority was contracted by the PepeMints team to perform the Security audit of the PepeMints smart contract 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 April 30th, 2023.

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

- PepeMints is an auction token and high-yield Certificate of Deposit on the BNB network, using an Auction Lobby system to support the price and liquidity.
- PepeMints is a unique DeFi protocol that represents the next step in evolution from WhalesCandy.
- PepeMints is a PM token which has burn, transfer, stake, stakeInt, refStake, sendETH, reinvest, claimRewards, buyAndStake, updateDaily functionalities.
- PepeMints contract inherits IERC20, ReentrancyGuard standard smart contracts from the OpenZeppelin library and IUniswapV2Router02 standard smart contracts from the uniswap from github library .
- These OpenZeppelin contracts and uniswap library are considered community audited and time tested, and hence are not part of the audit scope.

Audit scope

Name	**Code Review and Security Analysis Report for PepeMints Smart Contract**
Platform	**BNB Network / Solidity**
File	PepeMints.sol
File MD5 Hash	BA089A2AD7740091597D0DA5B8634F5A
Updated File MD5 Hash	01887D19A7E3C8CEF5732EAB4DBBEDB7
Audit Date	April 30th, 2023
Revised Audit Date	May 1st, 2023

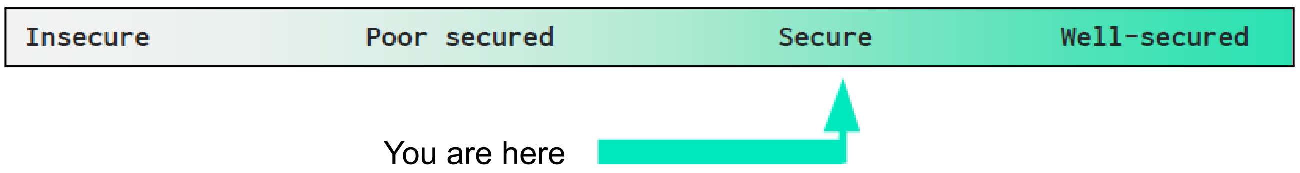
Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Tokenomics: ● Name: PepeMints.vip ● Symbol: PM ● Decimals: 18 ● 1 Day: 1 days	YES, This is valid.
Auction supply. ● Decreases by 1.5%/day. ● Daily amount and rate of decrease can be reset by admin.	YES, This is valid.
Team tokens ● 5% of the amount offered in each daily auction.	YES, This is valid.
● Forex Trading Fee: 12.50% ● 2.5% of lobby entries goto lottery pot. ● Buy Back Percent: 5% ● Tax Factor: 100% ● Percent to Receive On Sell: 90% ● Percent to Receive On Buy: 50% ● Daily Available Tokens: 16,420 ether ● Daily Available Tokens Decrease: 1.5%	YES, This is valid.
<u>Owner has control over following functions:</u> ● Current owner can transfer ownership of the contract to	YES, This is valid.

a new account. ● Buy and sell tax can be toggled on/off. ● Buy tax can be set by the admin to any amount. ● Sales tax can be set by the admin but is limited to 10%. ● Daily amount and rate of decrease can be reset by admin. ● Tax Factor value can be set by the owner. ● Set the percentage to be received when buying from PancakeSwap by the owner. ● Set the right amount of user stakes for UI by the owner. ● Set the right amount of overall StakedToken for UI by the owner.	

Audit Summary

According to the standard audit assessment, Customer`s solidity based smart contracts are **"Secured"**. This token contract does contain owner control, which does not make it 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 1 low and some very low level issues.

We confirm that 1 informational severity issue is fixed in the revised smart contract code.

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	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 not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

Overall Audit Result: PASSED

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 PepeMints Token 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 PepeMints Token.

The PepeMints Token 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 a PepeMints Token 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://pepemints.vip 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.

AS-IS overview

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	nonReentrant	modifier	Passed	No Issue
3	_nonReentrantBefore	write	Passed	No Issue
4	_nonReentrantAfter	write	Passed	No Issue
5	_reentrancyGuardEntered	internal	Passed	No Issue
6	transferOwnership	write	access only Owner	No Issue
7	claimOwnership	write	Passed	No Issue
8	onlyOwner	modifier	Passed	No Issue
9	toggleSellTaxOn	external	access only Owner	No Issue
10	toggleBuyTaxOn	external	access only Owner	No Issue
11	totalSupply	external	Passed	No Issue
12	balanceOf	external	Passed	No Issue
13	allowance	external	Passed	No Issue
14	approve	write	Passed	No Issue
15	increaseAllowance	external	Passed	No Issue
16	decreaseAllowance	external	Passed	No Issue
17	setDailyAvailableTokens	external	access only Owner	No Issue
18	setDailyAvailableTokensDecrease Percentage	external	access only Owner	No Issue
19	setDev	external	access only Owner	No Issue
20	setBuyBackContract	external	access only Owner	No Issue
21	setLaunchpad	external	access only Owner	No Issue
22	setSettingDone	external	access only Owner	No Issue
23	setLotterySharePercentage	external	access only Owner	No Issue
24	setDevAuctionBuyFee	external	access only Owner	removed
25	setDripBuyAuctionFee	external	access only Owner	No Issue
26	setBuyBackPercent	external	access only Owner	No Issue
27	setTaxFactor	external	access only Owner	No Issue
28	setPercentToReceiveOnSell	external	access only Owner	No Issue
29	setPercentToReceiveOnBuy	external	access only Owner	No Issue
30	setExcludedFromSellTaxReceiver	external	access only Owner	No Issue
31	isExcludedFromSellTaxSender	read	Passed	No Issue
32	setExcludedFromBuyTaxReceiver	external	access only Owner	No Issue
33	isExcludedFromBuyTaxReceiver	read	Passed	No Issue
34	transfer	write	Passed	No Issue
35	transferFrom	write	Passed	No Issue
36	_transfer	internal	Passed	No Issue
37	_mint	internal	Passed	No Issue
38	_devMint	external	access only Owner	No Issue
39	burn	external	Passed	No Issue
40	_burn	internal	Passed	No Issue
41	stake	external	Passed	No Issue

42	stakeInt	internal	Passed	No Issue
43	refStake	internal	Passed	No Issue
44	setUsersStakeNumber	external	Passed	No Issue
45	setOverallStakedToken	external	Passed	No Issue
46	setAuctionEntry_allDays	external	Passed	No Issue
47	addStakesUser	external	Passed	No Issue
48	setStakesUser	external	Passed	No Issue
49	thisDay	read	Passed	No Issue
50	getAmountFromLiq	read	Passed	No Issue
51	buyAndStake	external	Function input parameters lack of check	Refer Audit Findings
52	updateDaily	write	Passed	No Issue
53	_updateDailyAvailableTokens	internal	Passed	No Issue
54	burnAndBuyback	internal	Passed	No Issue
55	buyShareFromAuction	external	Passed	No Issue
56	calculateTokenPerShareOnDay	read	Passed	No Issue
57	claimTokenFromSharesAndStake	external	Passed	No Issue
58	claimRefTokensAndStake	external	Passed	No Issue
59	calcReward	read	Passed	No Issue
60	calcClaim	external	Passed	No Issue
61	_collect	internal	Passed	No Issue
62	claimRewards	write	Passed	Fixed
63	reinvest	write	Passed	No Issue
64	claimRewardsInRange	external	Passed	No Issue
65	reinvestInRange	external	Passed	No Issue
66	getUserStakesInRange	external	Passed	No Issue
67	getAuctionEntryInfo	external	Passed	No Issue
68	checkLottery	internal	Passed	Fixed
69	sendETH	internal	Passed	No Issue
70	setForexTradingFee	external	access onlyOwner	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.

Audit Findings

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

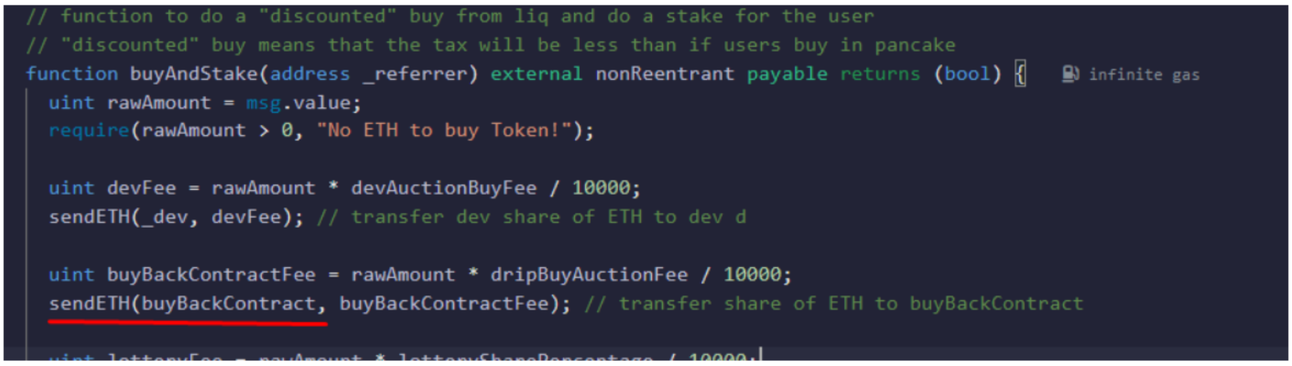
No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

(1) Function input parameters lack of check:



```
// function to do a "discounted" buy from liq and do a stake for the user
// "discounted" buy means that the tax will be less than if users buy in pancake
function buyAndStake(address _referrer) external nonReentrant payable returns (bool) {    infinite gas
  uint rawAmount = msg.value;
  require(rawAmount > 0, "No ETH to buy Token!");

  uint devFee = rawAmount * devAuctionBuyFee / 10000;
  sendETH(_dev, devFee); // transfer dev share of ETH to dev d

  uint buyBackContractFee = rawAmount * dripBuyAuctionFee / 10000;
  sendETH(buyBackContract, buyBackContractFee); // transfer share of ETH to buyBackContract

  uint lotteryFee = rawAmount * lotteryShareDercentage / 10000;
```

Variable validation is not performed in below functions:

- buyAndStake = buyBackContract

Resolution: We advise to put validation : int type variables should not be empty and > 0 & address type variables should not be address(0).

Very Low / Informational / Best practices:

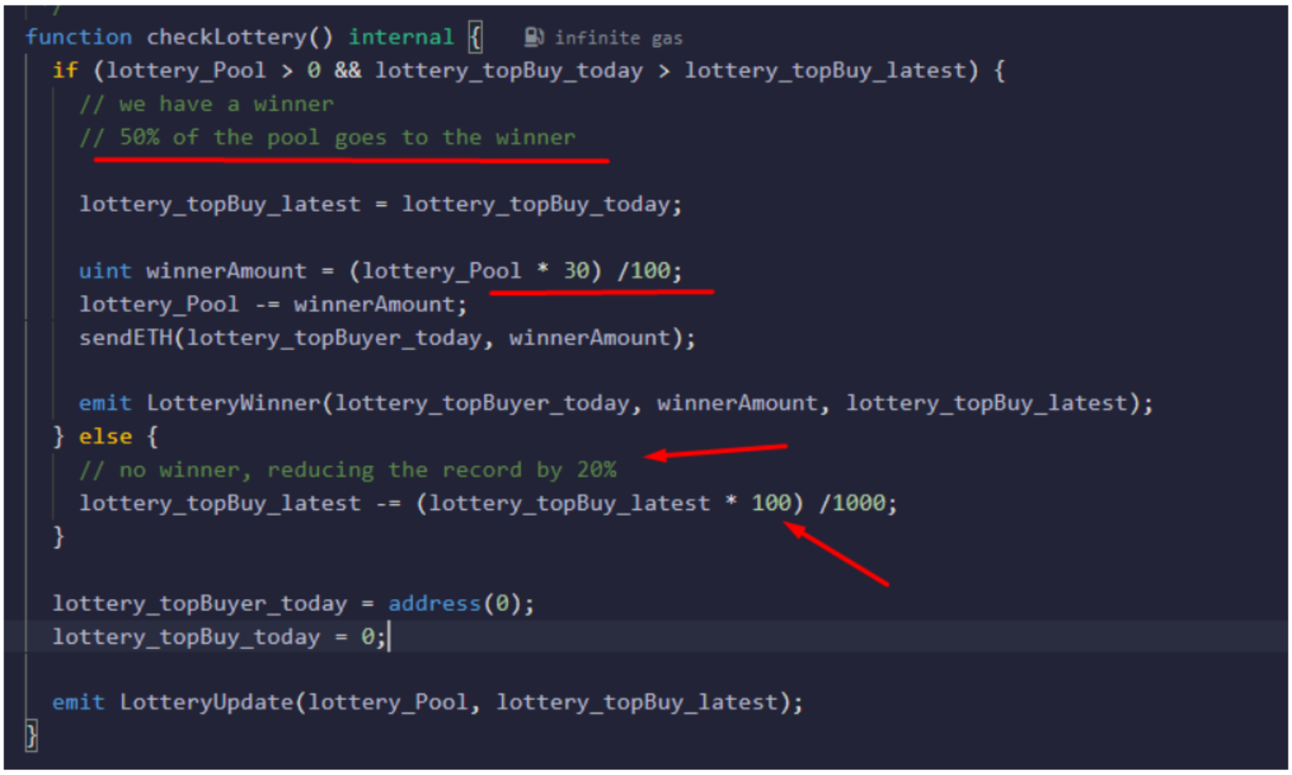
(1) SPDX license identifier missing:

SPDX license identifier not provided in source file.

Resolution: We suggest adding an SPDX license identifier.

(2) Code and comments mismatched:

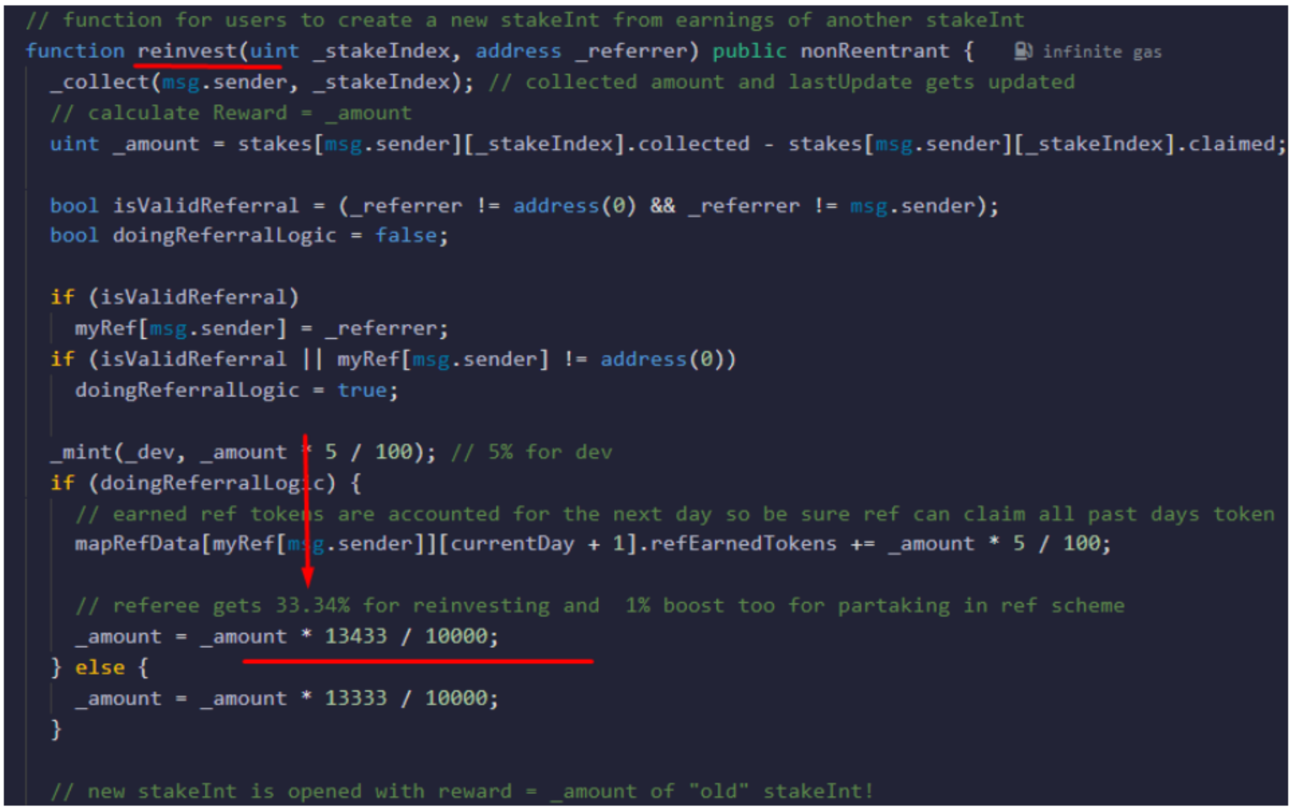
Function: checkLottery()



```
function checkLottery() internal {    infinite gas
  if (lottery_Pool > 0 && lottery_topBuy_today > lottery_topBuy_latest) {
    // we have a winner
    // 50% of the pool goes to the winner

    lottery_topBuy_latest = lottery_topBuy_today;

    uint winnerAmount = (lottery_Pool * 30) /100;
    lottery_Pool -= winnerAmount;
    sendETH(lottery_topBuyer_today, winnerAmount);

    emit LotteryWinner(lottery_topBuyer_today, winnerAmount, lottery_topBuy_latest);
  } else {
    // no winner, reducing the record by 20%
    lottery_topBuy_latest -= (lottery_topBuy_latest * 100) /1000;
  }

  lottery_topBuyer_today = address(0);
  lottery_topBuy_today = 0;

  emit LotteryUpdate(lottery_Pool, lottery_topBuy_latest);
}
```

In the checkLottery function , the comment says 50% goes to Winner , code calculates for 30% and according to the document Winner should get 33%. If there is no winner , the amount should decrease to 10% but code commented with 20%.

Function: reinvest()



```
// function for users to create a new stakeInt from earnings of another stakeInt
function reinvest(uint _stakeIndex, address _referrer) public nonReentrant {    infinite gas
  _collect(msg.sender, _stakeIndex); // collected amount and lastUpdate gets updated
  // calculate Reward = _amount
  uint _amount = stakes[msg.sender][_stakeIndex].collected - stakes[msg.sender][_stakeIndex].claimed;

  bool isValidReferral = (_referrer != address(0) && _referrer != msg.sender);
  bool doingReferralLogic = false;

  if (isValidReferral)
    myRef[msg.sender] = _referrer;
  if (isValidReferral || myRef[msg.sender] != address(0))
    doingReferralLogic = true;

  _mint(_dev, _amount * 5 / 100); // 5% for dev
  if (doingReferralLogic) {
    // earned ref tokens are accounted for the next day so be sure ref can claim all past days token
    mapRefData[myRef[msg.sender]][currentDay + 1].refEarnedTokens += _amount * 5 / 100;

    // referee gets 33.34% for reinvesting and  1% boost too for partaking in ref scheme
    _amount = _amount * 13433 / 10000;
  } else {
    _amount = _amount * 13333 / 10000;
  }

  // new stakeInt is opened with reward = _amount of "old" stakeInt!
```

In the reinvest function , code comment is 33.34% for reinvesting and in the document it's mentioned 33%. Also this comment is wrong "referee gets 1% boost too for partaking in ref scheme".

Resolution: We suggest correcting all the percentages and comments according to the document.

Status: This is fixed in the revised smart contract code.

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:

PepeMints.sol

- toggleSellTaxOn: Sell tax can be toggled on/off Sell tax can be toggled on/off.
- toggleBuyTaxOn: Buy tax can be toggled on/off Buy tax can be toggled on/off.
- setDailyAvailableTokens: Daily available token value can be set by the owner.
- setDailyAvailableTokensDecreasePercentage: Daily available token percentage can be set by the owner.
- setDev: Dev addresses can be set by the owner.
- setBuyBackContract: Buy Back Contract address can be set by the owner.
- setLaunchpad: Launchpad address can be set by the owner.
- setSettingDone: Setting status can be set by the owner.
- setLotterySharePercentage: Lottery share percentage can be set by the owner.
- setDripBuyAuctionFee: Drip Buy Auction fee can be set by the owner.
- setBuyBackPercent: Buyback Percentage can be set by the owner.
- setTaxFactor: Tax Factor value can be set by the owner.
- setPercentToReceiveOnSell: Set the percentage to be received when buying from PancakeSwap by the owner.
- setPercentToReceiveOnBuy: Set the percentage to be received when buying to PancakeSwap by the owner.
- setExcludedFromSellTaxReceiver: Set address to be in- or excluded from Tax when received by the owner.
- setExcludedFromBuyTaxReceiver: Set address to be in- or excluded from Tax when sender by the owner.
- _devMint: Dev mint by the owner.
- setUsersStakeNumber: Set the right amount of user stakes for UI by the owner.
- setOverallStakedToken: Set the right amount of overall StakedToken for UI by the owner.
- setAuctionEntry_allDays: Set the right amount of auctionEntry_allDays for UI by the owner.

- addStakesUser: Add a staker user address by the owner.
- setStakesUser: Set a staker user address by the owner.
- setForexTradingFee: Set the fee that goes to forex trading passive income with each auction entry by the owner.

BoringOwnable.sol

- transferOwnership: Current owner can transfer ownership of the contract to a new account.

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 1 low issue and 2 informational severity issues in the token smart contract. We confirm that 1 informational severity issue is fixed in the revised smart contract 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".**

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).

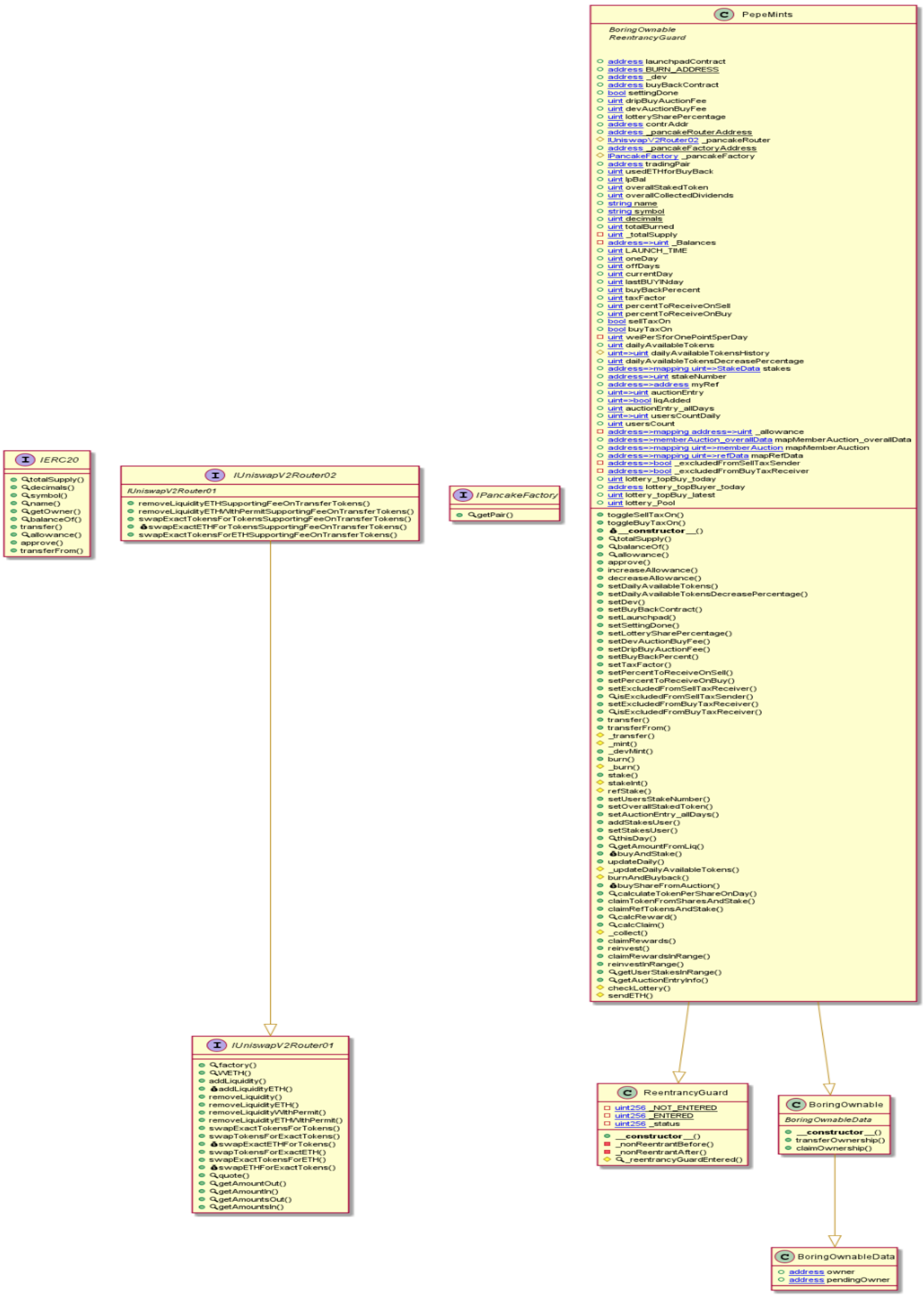
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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.

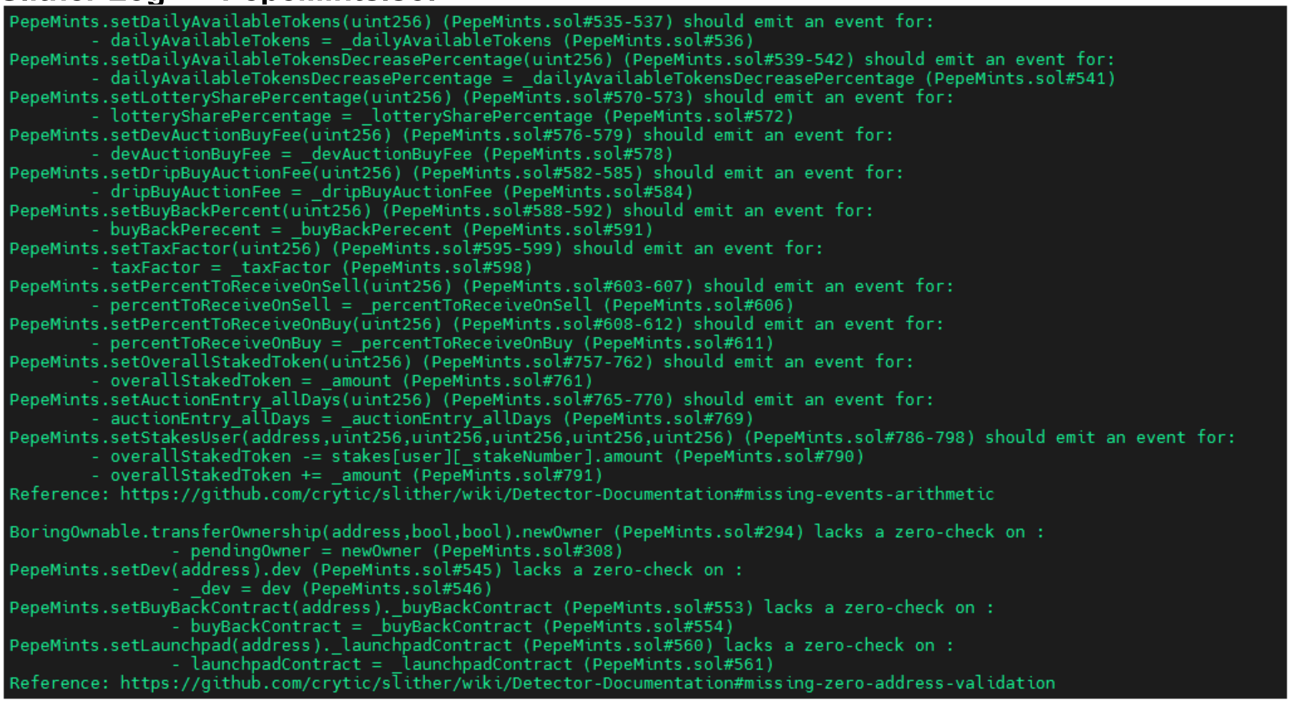
Appendix

Code Flow Diagram - PepeMints Token

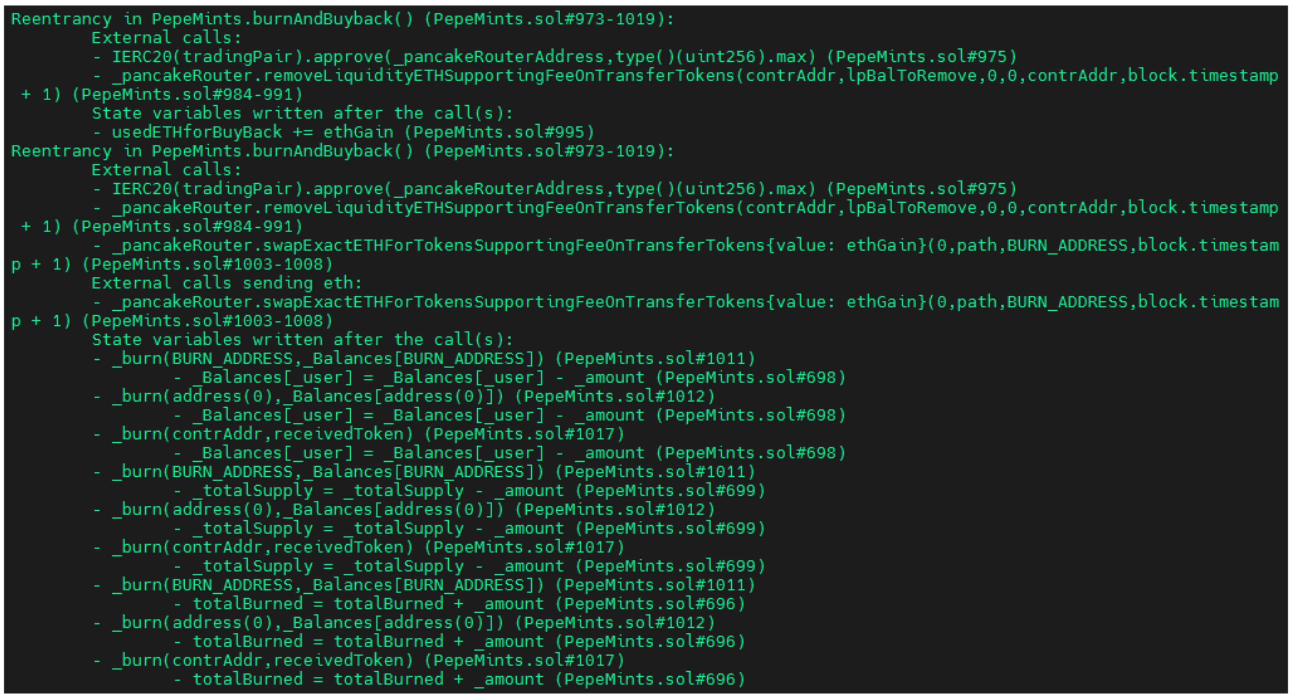


Slither Results Log

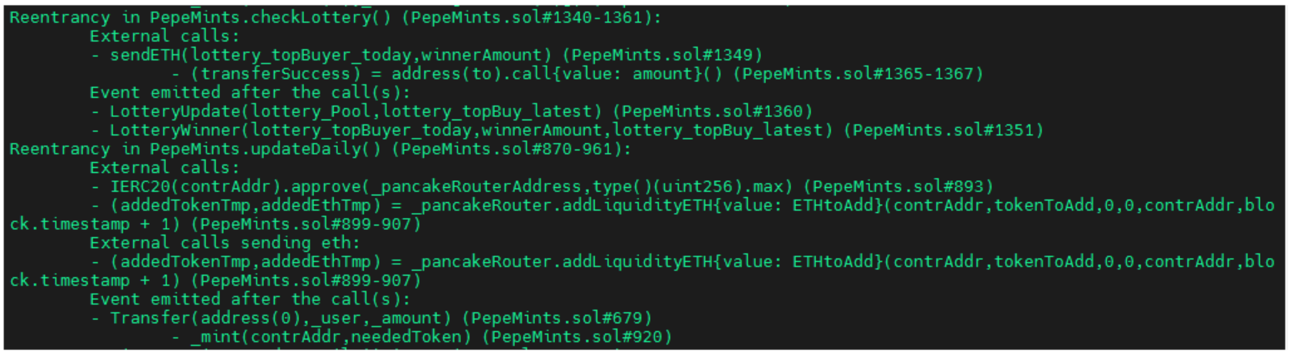
Slither Log >> PepeMints.sol



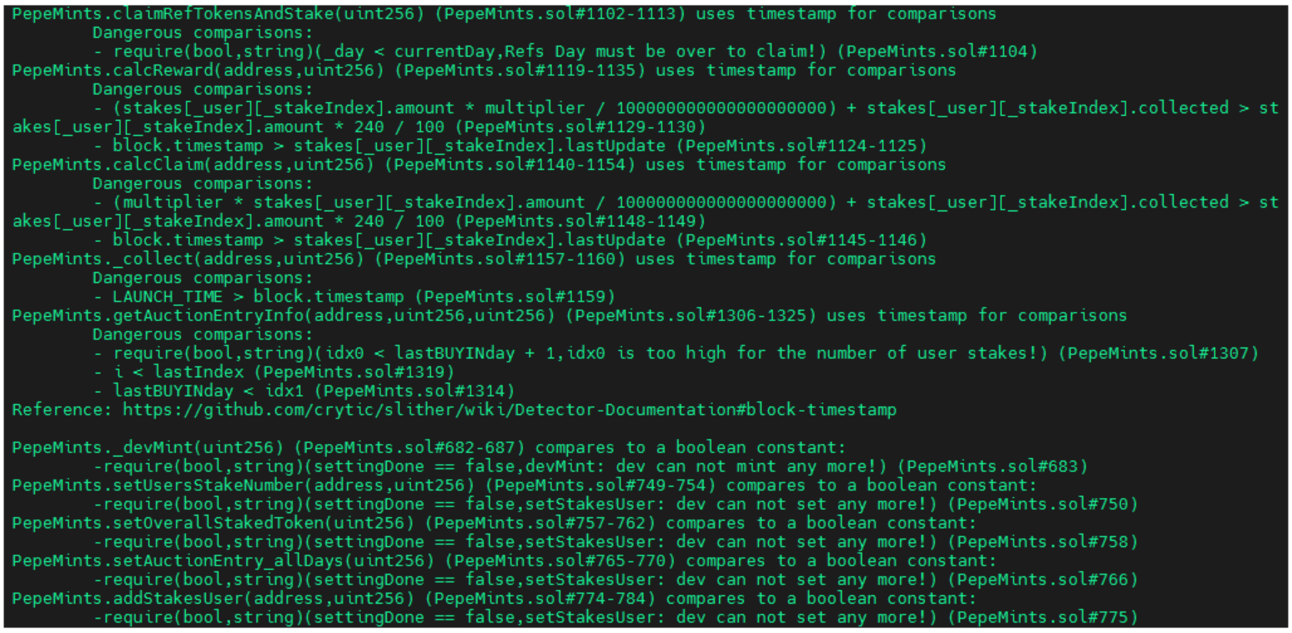
```
PepeMints.setDailyAvailableTokens(uint256) (PepeMints.sol#535-537) should emit an event for:
        - dailyAvailableTokens = _dailyAvailableTokens (PepeMints.sol#536)
PepeMints.setDailyAvailableTokensDecreasePercentage(uint256) (PepeMints.sol#539-542) should emit an event for:
        - dailyAvailableTokensDecreasePercentage = _dailyAvailableTokensDecreasePercentage (PepeMints.sol#541)
PepeMints.setLotterySharePercentage(uint256) (PepeMints.sol#570-573) should emit an event for:
        - lotterySharePercentage = _lotterySharePercentage (PepeMints.sol#572)
PepeMints.setDevAuctionBuyFee(uint256) (PepeMints.sol#576-579) should emit an event for:
        - devAuctionBuyFee = _devAuctionBuyFee (PepeMints.sol#578)
PepeMints.setDripBuyAuctionFee(uint256) (PepeMints.sol#582-585) should emit an event for:
        - dripBuyAuctionFee = _dripBuyAuctionFee (PepeMints.sol#584)
PepeMints.setBuyBackPercent(uint256) (PepeMints.sol#588-592) should emit an event for:
        - buyBackPerecent = _buyBackPerecent (PepeMints.sol#591)
PepeMints.setTaxFactor(uint256) (PepeMints.sol#595-599) should emit an event for:
        - taxFactor = _taxFactor (PepeMints.sol#598)
PepeMints.setPercentToReceiveOnSell(uint256) (PepeMints.sol#603-607) should emit an event for:
        - percentToReceiveOnSell = _percentToReceiveOnSell (PepeMints.sol#606)
PepeMints.setPercentToReceiveOnBuy(uint256) (PepeMints.sol#608-612) should emit an event for:
        - percentToReceiveOnBuy = _percentToReceiveOnBuy (PepeMints.sol#611)
PepeMints.setOverallStakedToken(uint256) (PepeMints.sol#757-762) should emit an event for:
        - overallStakedToken = _amount (PepeMints.sol#761)
PepeMints.setAuctionEntry_allDays(uint256) (PepeMints.sol#765-770) should emit an event for:
        - auctionEntry_allDays = _auctionEntry_allDays (PepeMints.sol#769)
PepeMints.setStakesUser(address,uint256,uint256,uint256,uint256,uint256) (PepeMints.sol#786-798) should emit an event for:
        - overallStakedToken -= stakes[user][_stakeNumber].amount (PepeMints.sol#790)
        - overallStakedToken += _amount (PepeMints.sol#791)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

BoringOwnable.transferOwnership(address,bool,bool).newOwner (PepeMints.sol#294) lacks a zero-check on :
                - pendingOwner = newOwner (PepeMints.sol#308)
PepeMints.setDev(address).dev (PepeMints.sol#545) lacks a zero-check on :
                - _dev = dev (PepeMints.sol#546)
PepeMints.setBuyBackContract(address)._buyBackContract (PepeMints.sol#553) lacks a zero-check on :
                - buyBackContract = _buyBackContract (PepeMints.sol#554)
PepeMints.setLaunchpad(address)._launchpadContract (PepeMints.sol#560) lacks a zero-check on :
                - launchpadContract = _launchpadContract (PepeMints.sol#561)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```



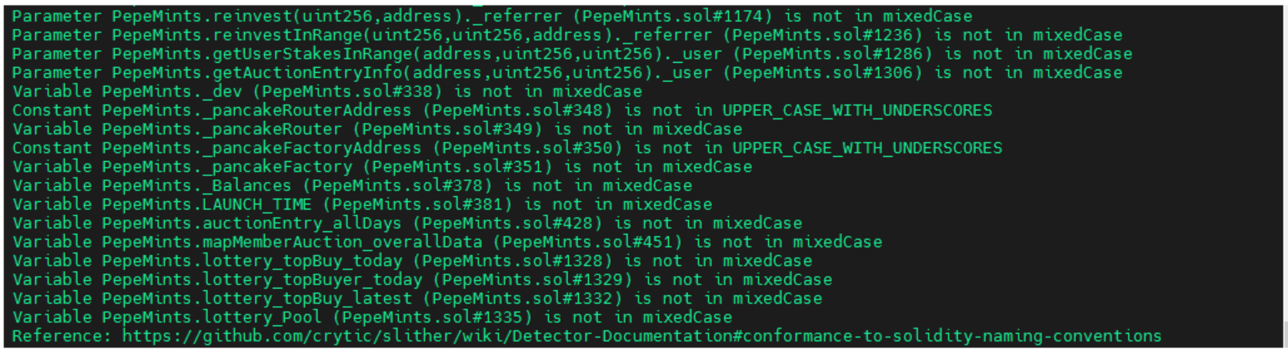
```
Reentrancy in PepeMints.burnAndBuyback() (PepeMints.sol#973-1019):
        External calls:
        - IERC20(tradingPair).approve(_pancakeRouterAddress,type()(uint256).max) (PepeMints.sol#975)
        - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAddr,lpBalToRemove,0,0,contrAddr,block.timestamp
 + 1) (PepeMints.sol#984-991)
        State variables written after the call(s):
        - usedETHforBuyBack += ethGain (PepeMints.sol#995)
Reentrancy in PepeMints.burnAndBuyback() (PepeMints.sol#973-1019):
        External calls:
        - IERC20(tradingPair).approve(_pancakeRouterAddress,type()(uint256).max) (PepeMints.sol#975)
        - _pancakeRouter.removeLiquidityETHSupportingFeeOnTransferTokens(contrAddr,lpBalToRemove,0,0,contrAddr,block.timestamp
 + 1) (PepeMints.sol#984-991)
        - _pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethGain}(0,path,BURN_ADDRESS,block.timestam
p + 1) (PepeMints.sol#1003-1008)
        External calls sending eth:
        - _pancakeRouter.swapExactETHForTokensSupportingFeeOnTransferTokens{value: ethGain}(0,path,BURN_ADDRESS,block.timestam
p + 1) (PepeMints.sol#1003-1008)
        State variables written after the call(s):
        - _burn(BURN_ADDRESS,_Balances[BURN_ADDRESS]) (PepeMints.sol#1011)
                - _Balances[_user] = _Balances[_user] - _amount (PepeMints.sol#698)
        - _burn(address(0),_Balances[address(0)]) (PepeMints.sol#1012)
                - _Balances[_user] = _Balances[_user] - _amount (PepeMints.sol#698)
        - _burn(contrAddr,receivedToken) (PepeMints.sol#1017)
                - _Balances[_user] = _Balances[_user] - _amount (PepeMints.sol#698)
        - _burn(BURN_ADDRESS,_Balances[BURN_ADDRESS]) (PepeMints.sol#1011)
                - _totalSupply = _totalSupply - _amount (PepeMints.sol#699)
        - _burn(address(0),_Balances[address(0)]) (PepeMints.sol#1012)
                - _totalSupply = _totalSupply - _amount (PepeMints.sol#699)
        - _burn(contrAddr,receivedToken) (PepeMints.sol#1017)
                - _totalSupply = _totalSupply - _amount (PepeMints.sol#699)
        - _burn(BURN_ADDRESS,_Balances[BURN_ADDRESS]) (PepeMints.sol#1011)
                - totalBurned = totalBurned + _amount (PepeMints.sol#696)
        - _burn(address(0),_Balances[address(0)]) (PepeMints.sol#1012)
                - totalBurned = totalBurned + _amount (PepeMints.sol#696)
        - _burn(contrAddr,receivedToken) (PepeMints.sol#1017)
                - totalBurned = totalBurned + _amount (PepeMints.sol#696)
```



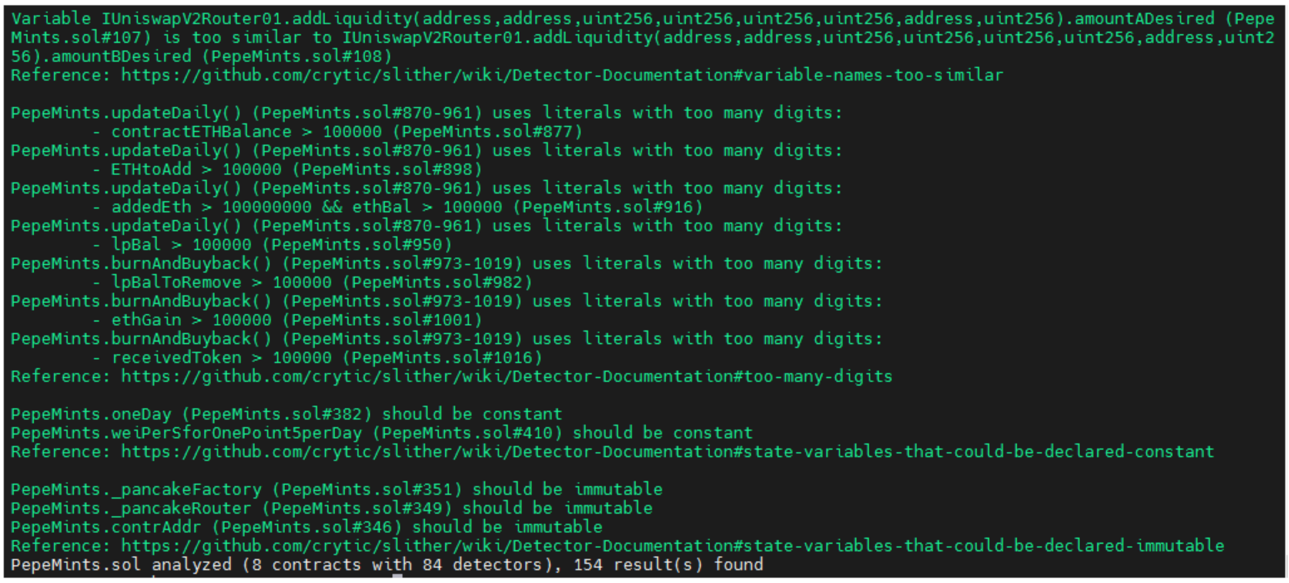
```
Reentrancy in PepeMints.checkLottery() (PepeMints.sol#1340-1361):
        External calls:
        - sendETH(lottery_topBuyer_today,winnerAmount) (PepeMints.sol#1349)
                - (transferSuccess) = address(to).call{value: amount}() (PepeMints.sol#1365-1367)
        Event emitted after the call(s):
        - LotteryUpdate(lottery_Pool,lottery_topBuy_latest) (PepeMints.sol#1360)
        - LotteryWinner(lottery_topBuyer_today,winnerAmount,lottery_topBuy_latest) (PepeMints.sol#1351)
Reentrancy in PepeMints.updateDaily() (PepeMints.sol#870-961):
        External calls:
        - IERC20(contrAddr).approve(_pancakeRouterAddress,type()(uint256).max) (PepeMints.sol#893)
        - (addedTokenTmp,addedEthTmp) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,blo
ck.timestamp + 1) (PepeMints.sol#899-907)
        External calls sending eth:
        - (addedTokenTmp,addedEthTmp) = _pancakeRouter.addLiquidityETH{value: ETHtoAdd}(contrAddr,tokenToAdd,0,0,contrAddr,blo
ck.timestamp + 1) (PepeMints.sol#899-907)
        Event emitted after the call(s):
        - Transfer(address(0),_user,_amount) (PepeMints.sol#679)
                - _mint(contrAddr,neededToken) (PepeMints.sol#920)
```



```
PepeMints.claimRefTokensAndStake(uint256) (PepeMints.sol#1102-1113) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_day < currentDay,Refs Day must be over to claim!) (PepeMints.sol#1104)
PepeMints.calcReward(address,uint256) (PepeMints.sol#1119-1135) uses timestamp for comparisons
        Dangerous comparisons:
        - (stakes[_user][_stakeIndex].amount * multiplier / 100000000000000000000) + stakes[_user][_stakeIndex].collected > st
akes[_user][_stakeIndex].amount * 240 / 100 (PepeMints.sol#1129-1130)
        - block.timestamp > stakes[_user][_stakeIndex].lastUpdate (PepeMints.sol#1124-1125)
PepeMints.calcClaim(address,uint256) (PepeMints.sol#1140-1154) uses timestamp for comparisons
        Dangerous comparisons:
        - (multiplier * stakes[_user][_stakeIndex].amount / 100000000000000000000) + stakes[_user][_stakeIndex].collected > st
akes[_user][_stakeIndex].amount * 240 / 100 (PepeMints.sol#1148-1149)
        - block.timestamp > stakes[_user][_stakeIndex].lastUpdate (PepeMints.sol#1145-1146)
PepeMints._collect(address,uint256) (PepeMints.sol#1157-1160) uses timestamp for comparisons
        Dangerous comparisons:
        - LAUNCH_TIME > block.timestamp (PepeMints.sol#1159)
PepeMints.getAuctionEntryInfo(address,uint256,uint256) (PepeMints.sol#1306-1325) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(idx0 < lastBUYINday + 1,idx0 is too high for the number of user stakes!) (PepeMints.sol#1307)
        - i < lastIndex (PepeMints.sol#1319)
        - lastBUYINday < idx1 (PepeMints.sol#1314)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

PepeMints._devMint(uint256) (PepeMints.sol#682-687) compares to a boolean constant:
        -require(bool,string)(settingDone == false,devMint: dev can not mint any more!) (PepeMints.sol#683)
PepeMints.setUsersStakeNumber(address,uint256) (PepeMints.sol#749-754) compares to a boolean constant:
        -require(bool,string)(settingDone == false,setStakesUser: dev can not set any more!) (PepeMints.sol#750)
PepeMints.setOverallStakedToken(uint256) (PepeMints.sol#757-762) compares to a boolean constant:
        -require(bool,string)(settingDone == false,setStakesUser: dev can not set any more!) (PepeMints.sol#758)
PepeMints.setAuctionEntry_allDays(uint256) (PepeMints.sol#765-770) compares to a boolean constant:
        -require(bool,string)(settingDone == false,setStakesUser: dev can not set any more!) (PepeMints.sol#766)
PepeMints.addStakesUser(address,uint256) (PepeMints.sol#774-784) compares to a boolean constant:
        -require(bool,string)(settingDone == false,setStakesUser: dev can not set any more!) (PepeMints.sol#775)
```



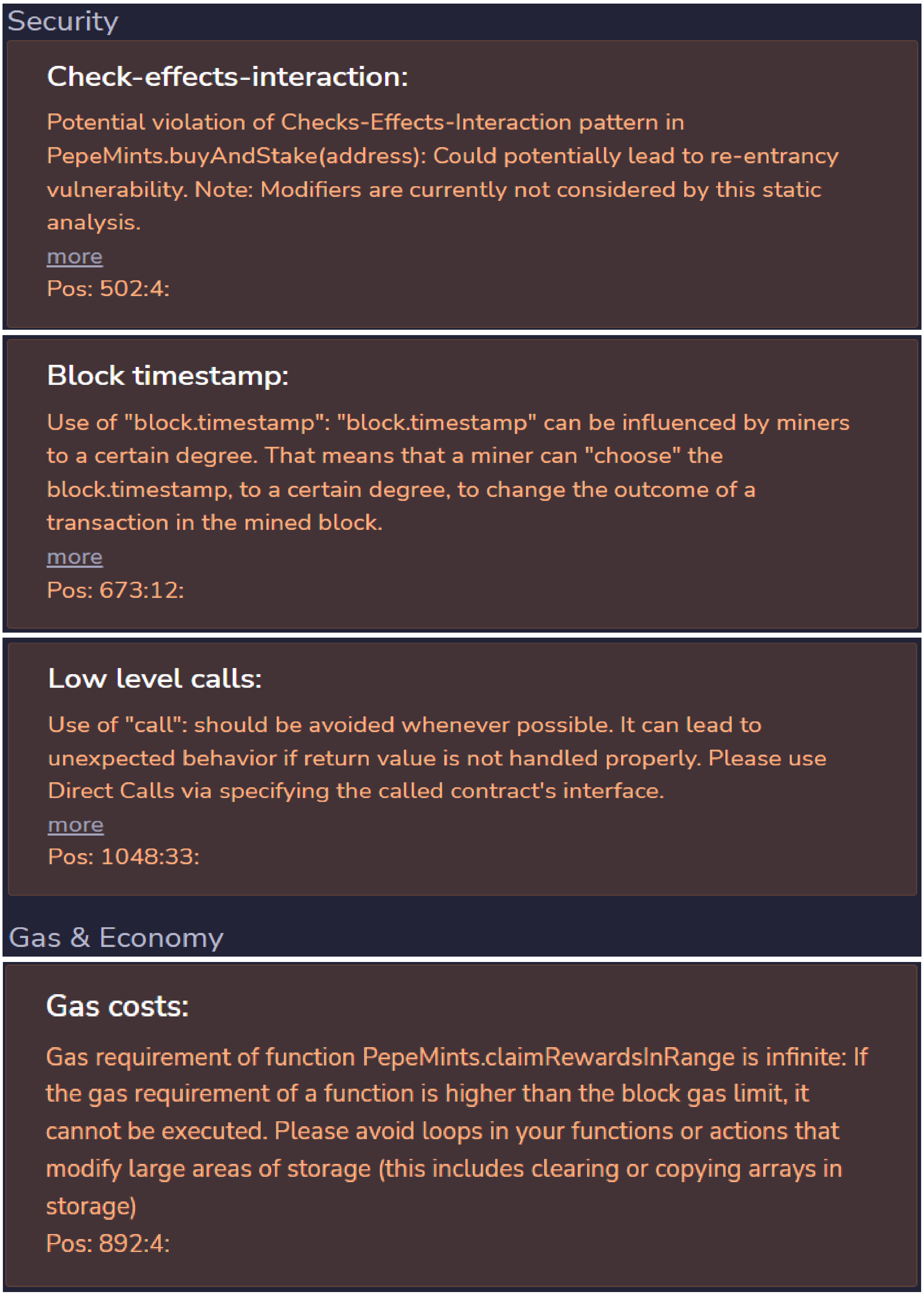
```
Parameter PepeMints.reinvest(uint256,address)._referrer (PepeMints.sol#1174) is not in mixedCase
Parameter PepeMints.reinvestInRange(uint256,uint256,address)._referrer (PepeMints.sol#1236) is not in mixedCase
Parameter PepeMints.getUserStakesInRange(address,uint256,uint256)._user (PepeMints.sol#1286) is not in mixedCase
Parameter PepeMints.getAuctionEntryInfo(address,uint256,uint256)._user (PepeMints.sol#1306) is not in mixedCase
Variable PepeMints._dev (PepeMints.sol#338) is not in mixedCase
Constant PepeMints._pancakeRouterAddress (PepeMints.sol#348) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PepeMints._pancakeRouter (PepeMints.sol#349) is not in mixedCase
Constant PepeMints._pancakeFactoryAddress (PepeMints.sol#350) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PepeMints._pancakeFactory (PepeMints.sol#351) is not in mixedCase
Variable PepeMints._Balances (PepeMints.sol#378) is not in mixedCase
Variable PepeMints.LAUNCH_TIME (PepeMints.sol#381) is not in mixedCase
Variable PepeMints.auctionEntry_allDays (PepeMints.sol#428) is not in mixedCase
Variable PepeMints.mapMemberAuction_overallData (PepeMints.sol#451) is not in mixedCase
Variable PepeMints.lottery_topBuy_today (PepeMints.sol#1328) is not in mixedCase
Variable PepeMints.lottery_topBuyer_today (PepeMints.sol#1329) is not in mixedCase
Variable PepeMints.lottery_topBuy_latest (PepeMints.sol#1332) is not in mixedCase
Variable PepeMints.lottery_Pool (PepeMints.sol#1335) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```



```
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Pepe
Mints.sol#107) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint2
56).amountBDesired (PepeMints.sol#108)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PepeMints.updateDaily() (PepeMints.sol#870-961) uses literals with too many digits:
        - contractETHBalance > 100000 (PepeMints.sol#877)
PepeMints.updateDaily() (PepeMints.sol#870-961) uses literals with too many digits:
        - ETHtoAdd > 100000 (PepeMints.sol#898)
PepeMints.updateDaily() (PepeMints.sol#870-961) uses literals with too many digits:
        - addedEth > 100000000 && ethBal > 100000 (PepeMints.sol#916)
PepeMints.updateDaily() (PepeMints.sol#870-961) uses literals with too many digits:
        - lpBal > 100000 (PepeMints.sol#950)
PepeMints.burnAndBuyback() (PepeMints.sol#973-1019) uses literals with too many digits:
        - lpBalToRemove > 100000 (PepeMints.sol#982)
PepeMints.burnAndBuyback() (PepeMints.sol#973-1019) uses literals with too many digits:
        - ethGain > 100000 (PepeMints.sol#1001)
PepeMints.burnAndBuyback() (PepeMints.sol#973-1019) uses literals with too many digits:
        - receivedToken > 100000 (PepeMints.sol#1016)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

PepeMints.oneDay (PepeMints.sol#382) should be constant
PepeMints.weiPerSforOnePoint5perDay (PepeMints.sol#410) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

PepeMints._pancakeFactory (PepeMints.sol#351) should be immutable
PepeMints._pancakeRouter (PepeMints.sol#349) should be immutable
PepeMints.contrAddr (PepeMints.sol#346) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
PepeMints.sol analyzed (8 contracts with 84 detectors), 154 result(s) found
```

Solidity Static Analysis

PepeMints.sol



Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in PepeMints.buyAndStake(address): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 502:4:

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: 673:12:

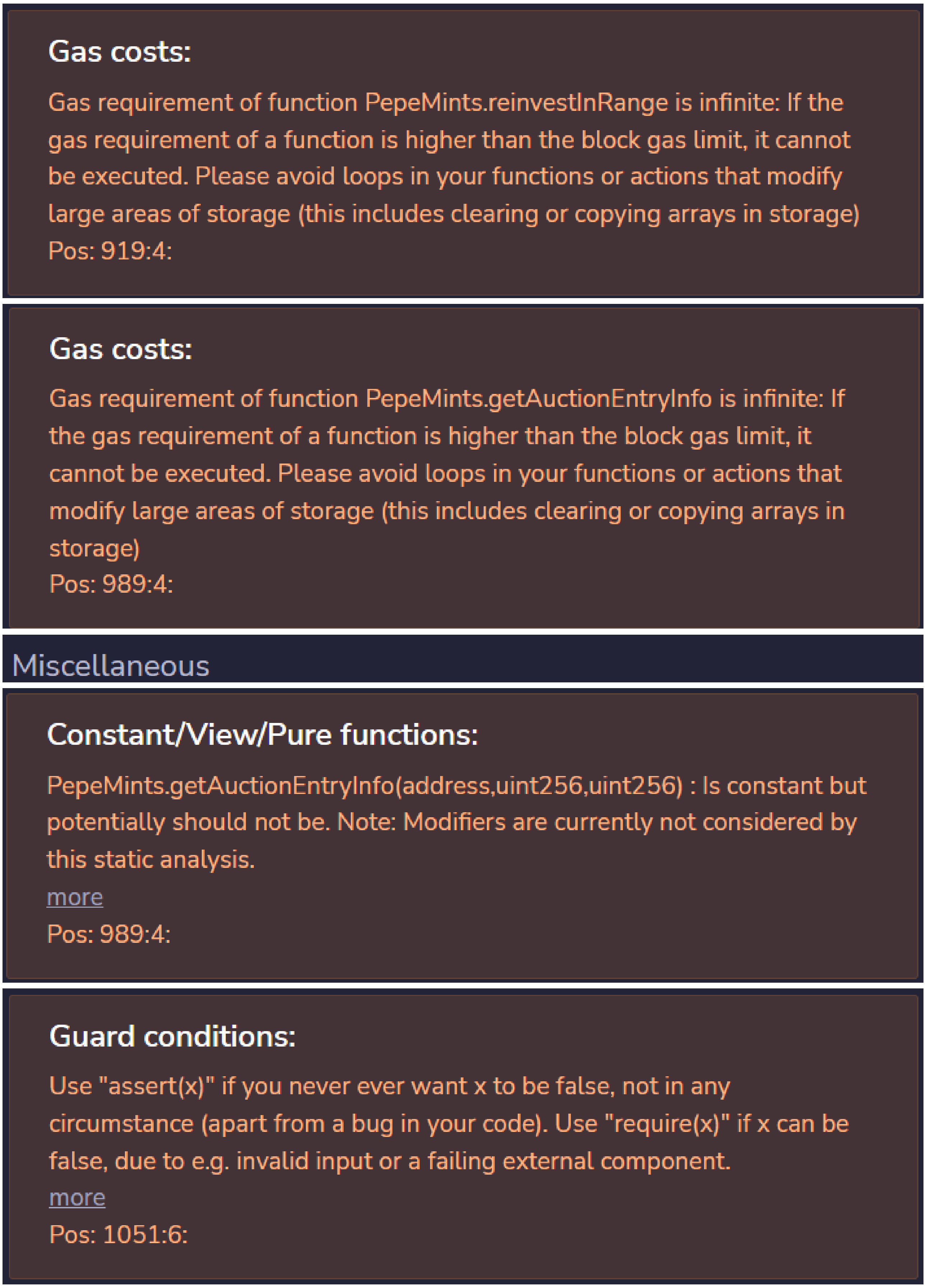
Low level calls:

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: 1048:33:

Gas & Economy

Gas costs:

Gas requirement of function PepeMints.claimRewardsInRange 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: 892:4:



Gas costs:

Gas requirement of function PepeMints.reinvestInRange 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: 919:4:

Gas costs:

Gas requirement of function PepeMints.getAuctionEntryInfo 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: 989:4:

Miscellaneous

Constant/View/Pure functions:

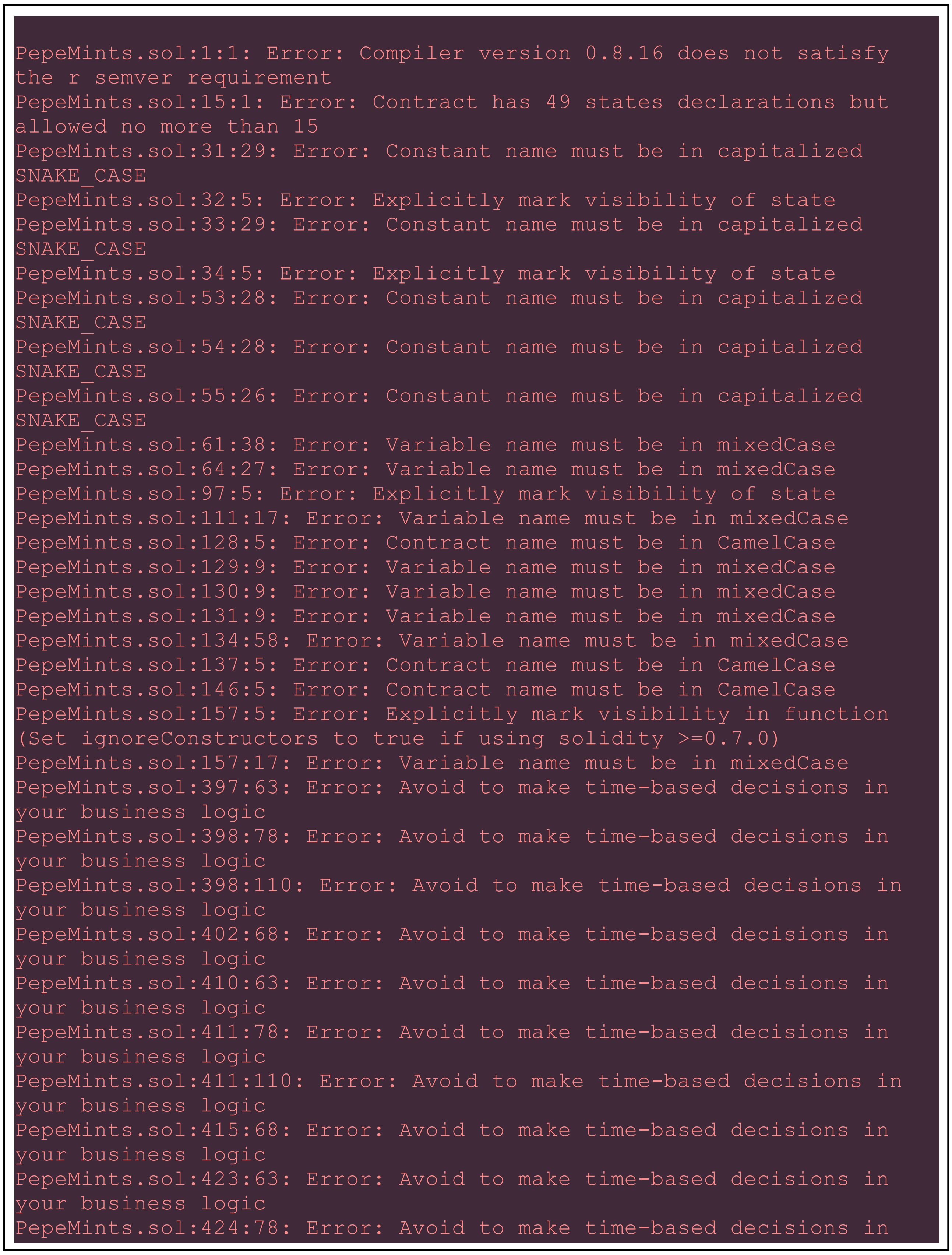
PepeMints.getAuctionEntryInfo(address,uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 989:4:

Guard conditions:

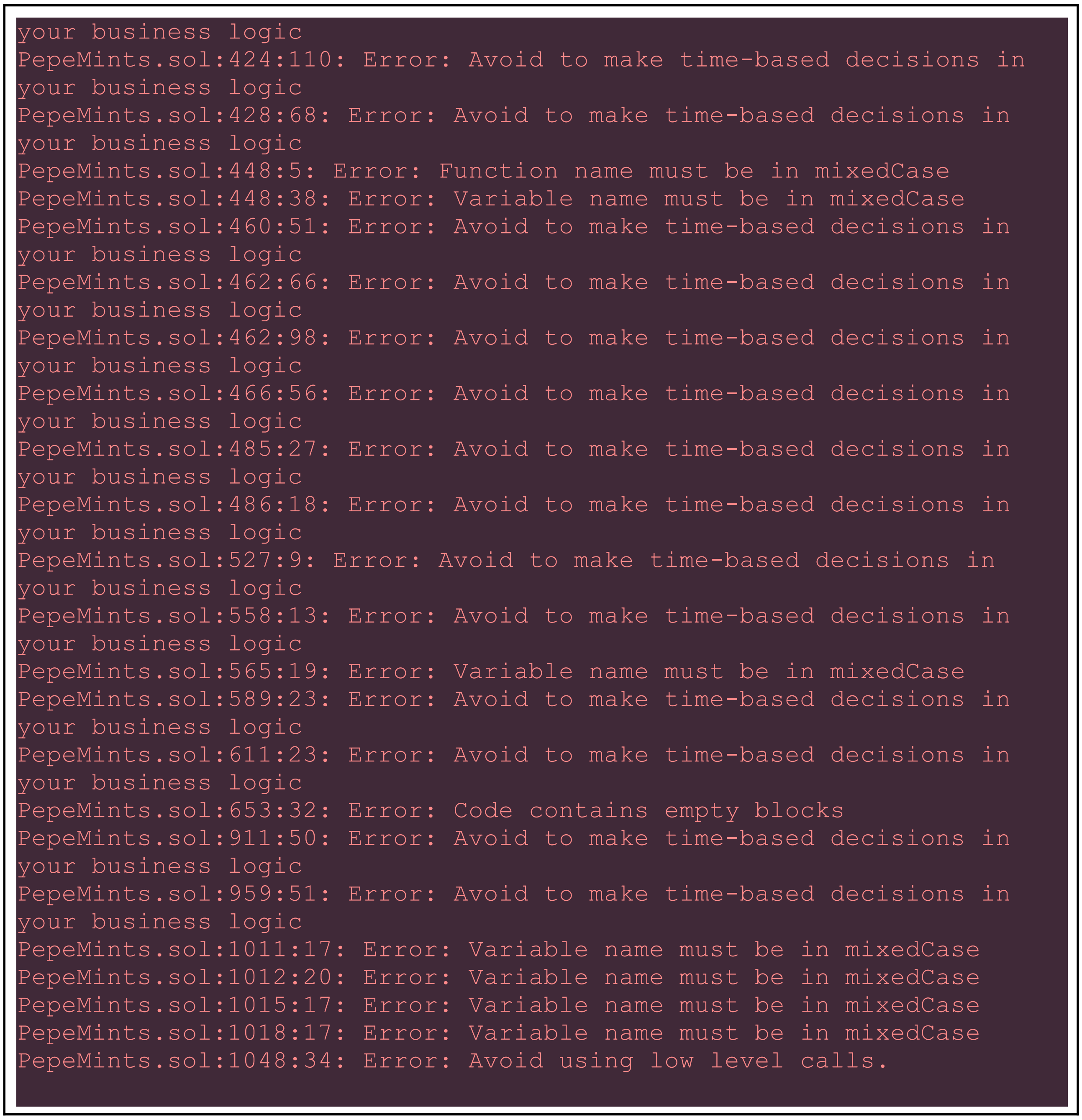
Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1051:6:

Solhint Linter

PepeMints.sol



```
PepeMints.sol:1:1: Error: Compiler version 0.8.16 does not satisfy
the r semver requirement
PepeMints.sol:15:1: Error: Contract has 49 states declarations but
allowed no more than 15
PepeMints.sol:31:29: Error: Constant name must be in capitalized
SNAKE_CASE
PepeMints.sol:32:5: Error: Explicitly mark visibility of state
PepeMints.sol:33:29: Error: Constant name must be in capitalized
SNAKE_CASE
PepeMints.sol:34:5: Error: Explicitly mark visibility of state
PepeMints.sol:53:28: Error: Constant name must be in capitalized
SNAKE_CASE
PepeMints.sol:54:28: Error: Constant name must be in capitalized
SNAKE_CASE
PepeMints.sol:55:26: Error: Constant name must be in capitalized
SNAKE_CASE
PepeMints.sol:61:38: Error: Variable name must be in mixedCase
PepeMints.sol:64:27: Error: Variable name must be in mixedCase
PepeMints.sol:97:5: Error: Explicitly mark visibility of state
PepeMints.sol:111:17: Error: Variable name must be in mixedCase
PepeMints.sol:128:5: Error: Contract name must be in CamelCase
PepeMints.sol:129:9: Error: Variable name must be in mixedCase
PepeMints.sol:130:9: Error: Variable name must be in mixedCase
PepeMints.sol:131:9: Error: Variable name must be in mixedCase
PepeMints.sol:134:58: Error: Variable name must be in mixedCase
PepeMints.sol:137:5: Error: Contract name must be in CamelCase
PepeMints.sol:146:5: Error: Contract name must be in CamelCase
PepeMints.sol:157:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
PepeMints.sol:157:17: Error: Variable name must be in mixedCase
PepeMints.sol:397:63: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:398:78: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:398:110: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:402:68: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:410:63: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:411:78: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:411:110: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:415:68: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:423:63: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:424:78: Error: Avoid to make time-based decisions in
```



```
your business logic
PepeMints.sol:424:110: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:428:68: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:448:5: Error: Function name must be in mixedCase
PepeMints.sol:448:38: Error: Variable name must be in mixedCase
PepeMints.sol:460:51: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:462:66: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:462:98: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:466:56: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:485:27: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:486:18: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:527:9: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:558:13: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:565:19: Error: Variable name must be in mixedCase
PepeMints.sol:589:23: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:611:23: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:653:32: Error: Code contains empty blocks
PepeMints.sol:911:50: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:959:51: Error: Avoid to make time-based decisions in
your business logic
PepeMints.sol:1011:17: Error: Variable name must be in mixedCase
PepeMints.sol:1012:20: Error: Variable name must be in mixedCase
PepeMints.sol:1015:17: Error: Variable name must be in mixedCase
PepeMints.sol:1018:17: Error: Variable name must be in mixedCase
PepeMints.sol:1048:34: Error: Avoid using low level calls.
```

Software analysis result:

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

EtherAuthority