Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Project Name: GuitarSwap Protocol
Platform: Binance Smart Chain
Website: www.guitarswap.exchange
Language: Solidity
Date: November 3rd, 2021

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 GuitarSwap team to perform the Security audit of the GuitarSwap 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 November 3rd, 2021.

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

GuitarSwap is a decentralized Automated Market Maker (AMM) platform. It lets users swap the digital assets in a decentralized way. Guitar Token is a BEP20 token smart contract running as the backbone of GuitarSwap ecosystem.

Audit scope

Name	Code Review and Security Analysis Report for GuitarSwap Protocol Smart Contracts
Platform	**BSC / Solidity**
File 1	GuitarToken.sol
File 1 MD5 Hash	B71057C1EBEADB606319B5910423DD77
File 2	GuitarSyrupBar.sol
File 2 MD5 Hash	271EB5D0B227358BCA0237C7A3F1F7AD
File 3	GuitarMasterChef.sol
File 3 MD5 Hash	5E9CF19CBE8C837B85F77EFD641DF3EC
Audit Date	November 3rd, 2021

Claimed Smart Contracts Features

Claimed Feature Detail	Our Observation
File 1: GuitarToken.sol • Name: GuitarSwap Token • Symbol: GUT • Decimals: 18 • Maximum Supply: 1 Billion GUT • Total Supply: 520 Million BUT	**YES, This is valid.**
File 2: GuitarSyrupBar.sol • Name: SyrupBar Token • Symbol: SYRUP • Decimals: 18	**YES, This is valid.**
File 3: GuitarMasterChef.sol • Bonus Multiplier: 1 • Maximum Emission Rate: 10,00,000 • Maximum Deposit Fee per pools: 20% • Guitar per Block: 7 • Pool Length: 10 • Total Allocation Points: 5333	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. These Protocol contracts do 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 4 low and some very low level issues.

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	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	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	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	Moderated
	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**

Email: audit@EtherAuthority.io

Code Quality

This audit scope has 3 smart contracts files. Smart contracts contains Libraries, Smart contracts, inherits and Interfaces. This is a compact and well written smart contract.

The libraries in GuitarSwap 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 GuitarSwap Protocol.

The GuitarSwap 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 **not** well commented on smart contracts.

Documentation

We were given a GuitarSwap Protocol smart contracts code in the form of a BSCScan 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 contract 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

GuitarToken.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mintFor	write	Require condition after mint	Refer Audit Findings
3	mint	write	Require condition after mint	Refer Audit Findings
4	delegates	external	Passed	No Issue
5	delegate	external	Passed	No Issue
6	delegateBySig	external	Passed	No Issue
7	getCurrentVotes	external	Passed	No Issue
8	getPriorVotes	external	Passed	No Issue
9	_delegate	internal	Passed	No Issue
10	_moveDelegates	internal	Passed	No Issue
11	_writeCheckpoint	internal	Passed	No Issue
12	safe32	internal	Passed	No Issue
13	getChainId	internal	Passed	No Issue
14	getOwner	external	Passed	No Issue
15	name	read	Passed	No Issue
16	decimals	read	Passed	No Issue
17	symbol	read	Passed	No Issue
18	totalSupply	read	Passed	No Issue
19	balanceOf	read	Passed	No Issue
20	transfer	write	Passed	No Issue
21	allowance	write	Passed	No Issue
22	approve	write	Passed	No Issue
23	transferFrom	write	Passed	No Issue
24	increaseAllowance	write	Passed	No Issue
25	decreaseAllowance	write	Passed	No Issue
26	mint	write	access only Owner	No Issue
27	_transfer	internal	Passed	No Issue
28	_mint	internal	Passed	No Issue
29	_burn	internal	Passed	No Issue
30	_approve	internal	Passed	No Issue
31	_burnFrom	internal	Passed	No Issue
32	owner	read	Passed	No Issue
33	onlyOwner	modifier	Passed	No Issue
34	renounceOwnership	write	access only Owner	No Issue
35	transferOwnership	write	access only Owner	No Issue

GuitarSyrupBar.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	mint	write	access only Owner	No Issue
3	burn	write	access only Owner	No Issue
4	safeGuitarTransfer	write	access only Owner	No Issue
5	delegates	external	Passed	No Issue
6	delegate	external	Passed	No Issue
7	delegateBySig	external	Passed	No Issue
8	getCurrentVotes	external	Passed	No Issue
9	getPriorVotes	external	Passed	No Issue
10	_delegate	internal	Passed	No Issue
11	_moveDelegates	internal	Passed	No Issue
12	_writeCheckpoint	internal	Passed	No Issue
13	safe32	internal	Passed	No Issue
14	getChainId	internal	Passed	No Issue
15	getOwner	external	Passed	No Issue
16	name	read	Passed	No Issue
17	decimals	read	Passed	No Issue
18	symbol	read	Passed	No Issue
19	totalSupply	read	Passed	No Issue
20	balanceOf	read	Passed	No Issue
21	transfer	write	Passed	No Issue
22	allowance	write	Passed	No Issue
23	approve	write	Passed	No Issue
24	transferFrom	write	Passed	No Issue
25	increaseAllowance	write	Passed	No Issue
26	decreaseAllowance	write	Passed	No Issue
27	mint	write	access only Owner	No Issue
28	_transfer	internal	Passed	No Issue
29	_mint	internal	Passed	No Issue
30	_burn	internal	Passed	No Issue
31	_approve	internal	Passed	No Issue
32	_burnFrom	internal	Passed	No Issue
33	owner	read	Passed	No Issue
34	onlyOwner	modifier	Passed	No Issue
35	renounceOwnership	write	access only Owner	No Issue
36	transferOwnership	write	access only Owner	No Issue

GuitarMasterChef.sol

Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	owner	read	Passed	No Issue
3	onlyOwner	modifier	Passed	No Issue
4	renounceOwnership	write	access only Owner	No Issue
5	transferOwnership	write	access only Owner	No Issue
6	updateMultiplier	write	access only Owner	No Issue
7	poolLength	external	Passed	No Issue
8	add	external	Critical operation lacks event log	Refer Audit Findings
9	set	external	Critical operation lacks event log	Refer Audit Findings
10	updateStakingPool	internal	Passed	No Issue
11	getMultiplier	write	Passed	No Issue
12	pendingGuitar	external	Passed	No Issue
13	massUpdatePools	write	Infinite loops possibility	Refer Audit Findings
14	updatePool	write	Critical operation lacks event log	Refer Audit Findings
15	deposit	external	Passed	No Issue
16	withdraw	external	Passed	No Issue
17	enterStaking	external	Passed	No Issue
18	leaveStaking	external	Passed	No Issue
19	emergencyWithdraw	external	Passed	No Issue
20	safeGuitarTransfer	internal	Passed	No Issue
21	dev	external	Passed	No Issue
22	setFeeAddress	external	Passed	No Issue
23	updateEmissionRate	external	Infinite loops possibility	Refer Audit Findings

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) Require condition after mint - **GuitarToken.sol**

```
function mintFor(address _to, uint256 _amount) public onlyOwner {
    _mint(_to, _amount);
    require(totalSupply() <= maxSupply, "reach max supply");
    _moveDelegates(address(0), _delegates[_to], _amount);
}

function mint(uint256 amount) public override onlyOwner returns (bool) {
    _mint(_msgSender(), amount);
    require(totalSupply() <= maxSupply, "reach max supply");
    return true;
}
```

Validation of the maximum limit for mint is executed after calling the mint function.

Resolution: We suggest validating for the maximum minting limit before calling the mint function.

(2) Critical operation lacks event log - **GuitarMasterChef.sol**

Missing event log for listed functions:

- add()
- set()
- updatePool()

Resolution: We suggest writing an event log for listed events.

(3) Infinite loops possibility - **GuitarMasterChef.sol**

As array elements will increase, then it will cost more and more gas. And eventually, it will stop all the functionality. After several hundreds of transactions, all those functions depending on it will stop. We suggest avoiding loops. For example, use mapping to store the array index. And query that data directly, instead of looping through all the elements to find an element.

Resolution: We suggest adjusting logic to replace loops with mapping or other code structure for the functions listed below:

- massUpdatePools() - poolInfo.length
- updateEmissionRate() - currentIndex

(4) GUT token can be added again in the pool - **GuitarMasterChef.sol**

GUT token which has been added while MasterChef deploy, can be added again in the pool.

Resolution: We suggest setting pool existence for the initial pool, so the same token cannot be added into the pool.

Status: Acknowledged by auditee.

Very Low / Informational / Best practices:

(1) Use latest solidity version - **GuitarToken.so, GuitarMasterChef.sol**

Using the latest solidity will prevent any compiler level bugs.

Resolution: We suggest using 0.8.9 which is the latest version.

(2) Warning: SPDX license identifier - **GuitarToken.so, GuitarMasterChef.sol, GuitarSyrupBar.sol**

GuitarToken.sol: Warning: SPDX license identifier not provided in source file.

Resolution: We suggest adding SPDX-License-Identifier.

(3) Multiple pragma - **GuitarToken.so, GuitarMasterChef.sol, GuitarSyrupBar.sol**

There are multiple pragmas with different compiler versions.

Resolution: We suggest using only one pragma and removing the other.

(4) Warning - **GuitarSyrupBar.sol**

GuitarSyrupBar.sol:1152:5: Warning: Documentation tag on non-public state variables will be disallowed in 0.7.0. You will need to use the @dev tag explicitly. /// @notice A record of each account's delegate.

Resolution: We suggest using 0.8.9 which is the latest version.

Centralization

These smart contracts have 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:

- mintFor: The GuitarToken owner can mint tokens to `_to`. Must only be called by the owner (MasterChef).
- mint: The GuitarToken owner can mint tokens himself.
- mint: The GuitarSyrupBar owner can mint tokens to `_to`. Must only be called by the owner (MasterChef).
- burn: The GuitarSyrupBar owner can burn any one's token amount.
- safeGuitarTransfer: The GuitarSyrupBar owner can Safe guitar transfer function, just in case if rounding error causes pool to not have enough GUTs.
- updateMultiplier: The GuitarMasterChef owner can update the multiplier number.
- add: The GuitarMasterChef owner can add a new lp to the pool.
- set: The GuitarMasterChef owner can update the given pool's GUT allocation point.
- updateEmissionRate: The GuitarMasterChef owner can update emission rate.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts, but did not find any critical severity issues. 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.

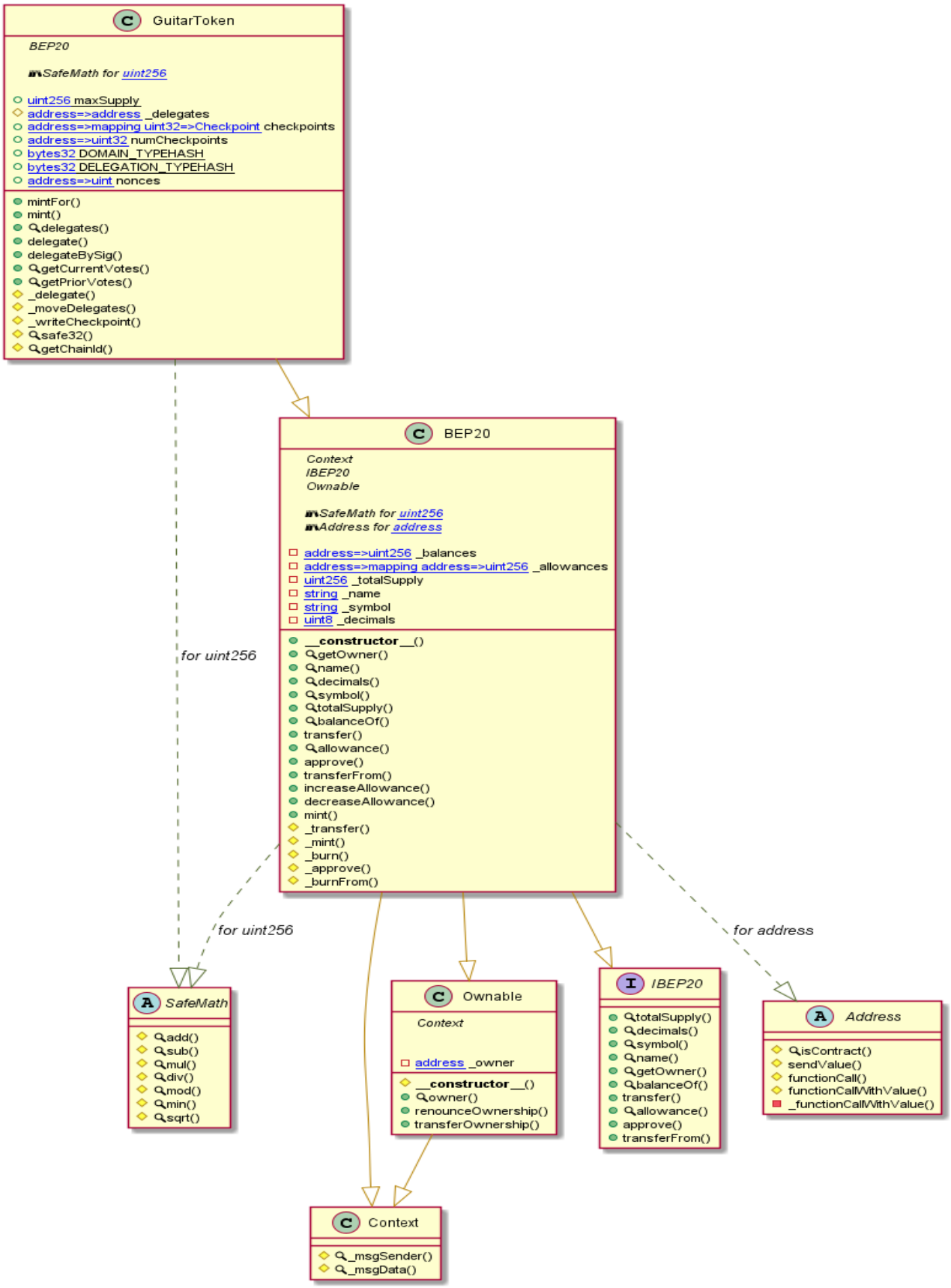
Technical Disclaimer

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.

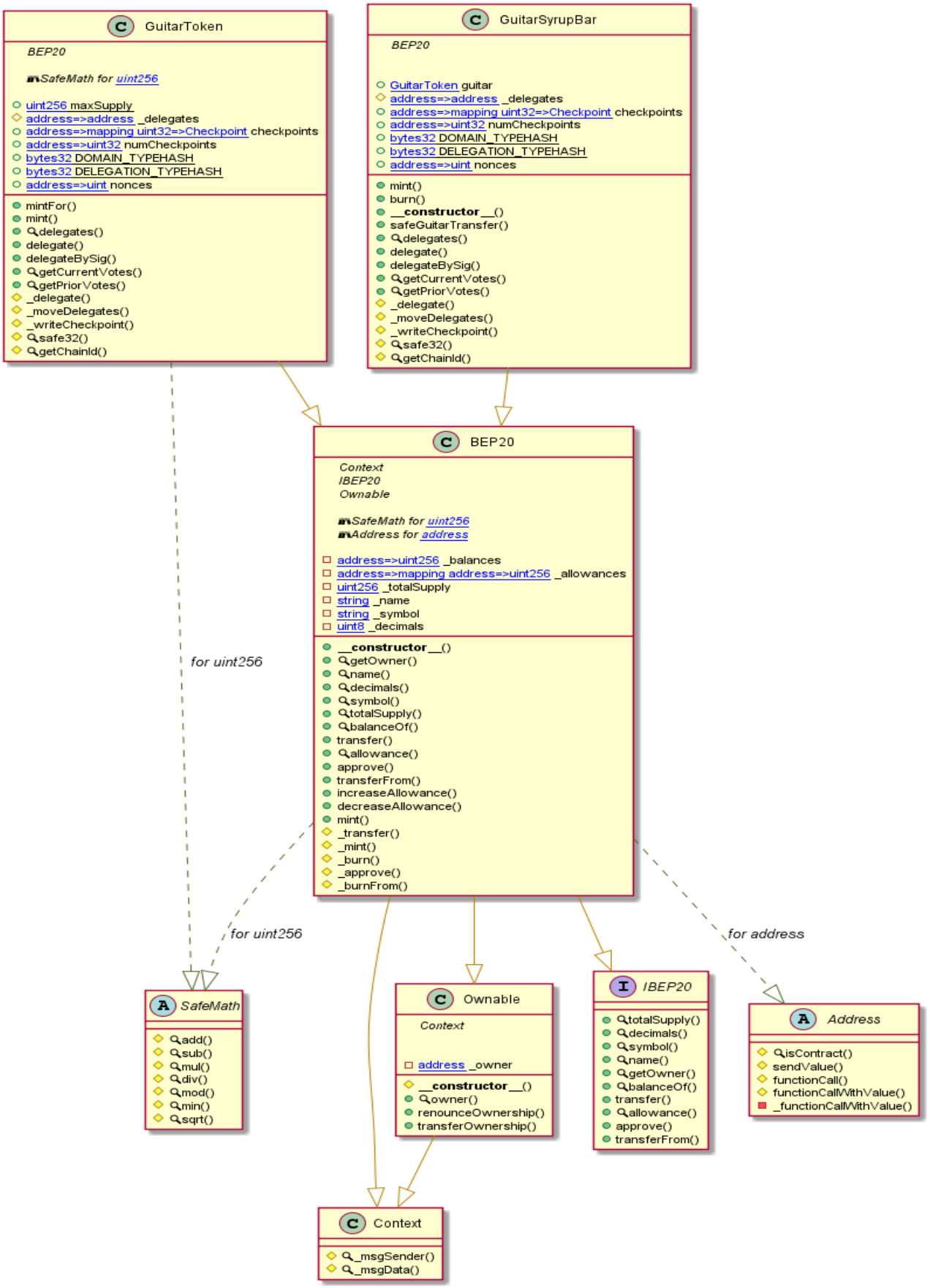
Appendix

Code Flow Diagram - GuitarSwap Protocol

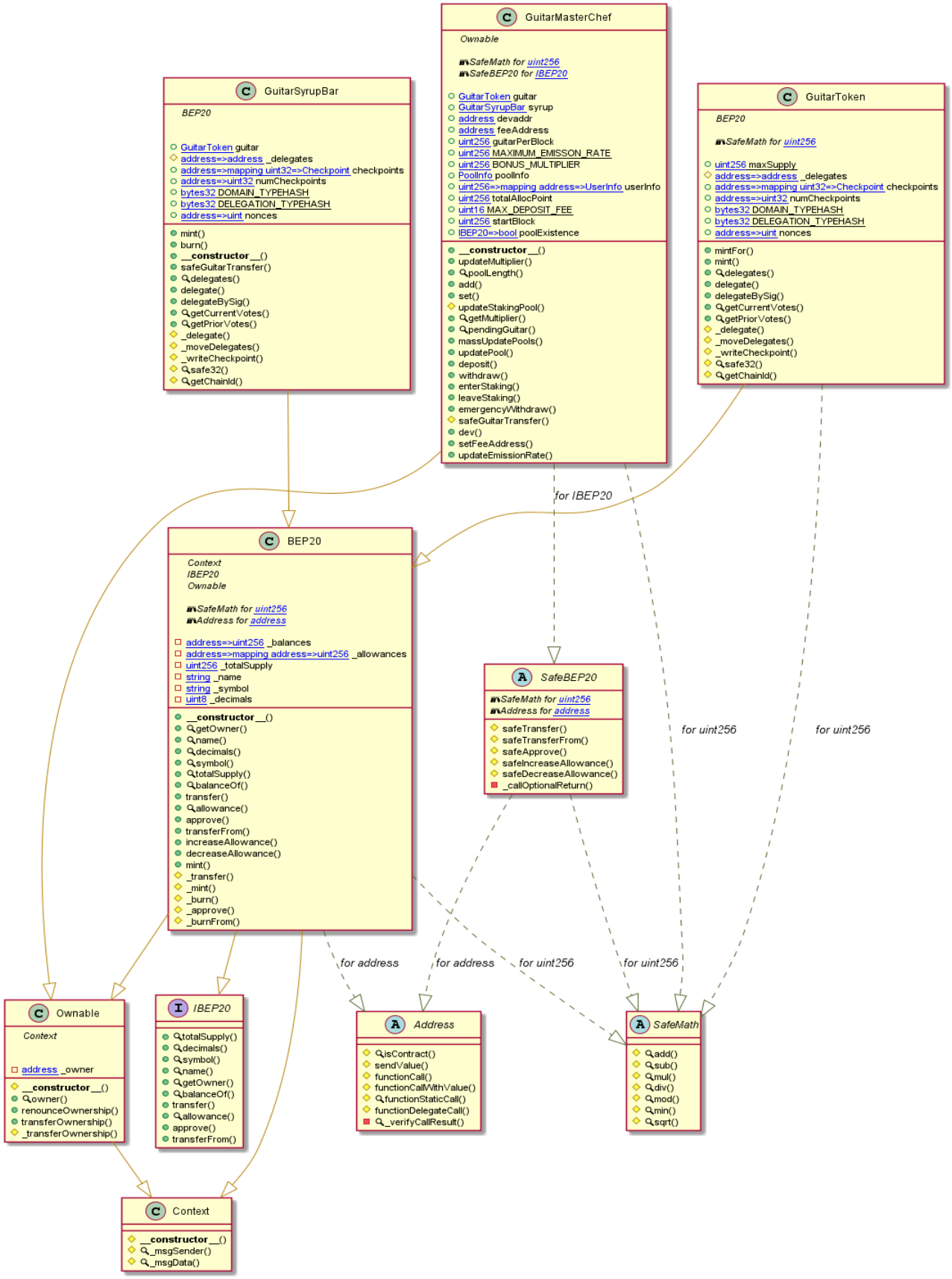
GuitarToken Diagram



GuitarSyrupBar Diagram



GuitarMasterChef Diagram



Slither Results Log

Slither log >> GuitarToken.sol

```
INFO:Detectors:
GuitarToken._writeCheckpoint(address,uint32,uint256,uint256) (GuitarToken.sol#1071-1089) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (GuitarToken.sol#1081)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
BEP20.allowance(address,address).owner (GuitarToken.sol#656) shadows:
        - Ownable.owner() (GuitarToken.sol#61-63) (function)
BEP20._approve(address,address,uint256).owner (GuitarToken.sol#828) shadows:
        - Ownable.owner() (GuitarToken.sol#61-63) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GuitarToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (GuitarToken.sol#937-978) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (GuitarToken.sol#976)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (GuitarToken.sol#405-416) uses assembly
        - INLINE ASM (GuitarToken.sol#412-414)
Address._functionCallWithValue(address,bytes,uint256,string) (GuitarToken.sol#513-539) uses assembly
        - INLINE ASM (GuitarToken.sol#531-534)
GuitarToken.getChainId() (GuitarToken.sol#1096-1100) uses assembly
        - INLINE ASM (GuitarToken.sol#1098)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['>0.6.6', '>=0.4.0', '>=0.6.0<0.8.0', '>=0.6.6']
        - >=0.6.0<0.8.0 (GuitarToken.sol#5)
        - >=0.6.0<0.8.0 (GuitarToken.sol#29)
        - >=0.4.0 (GuitarToken.sol#96)
        - >=0.4.0 (GuitarToken.sol#193)
        - >=0.6.6 (GuitarToken.sol#382)
        - >=0.4.0 (GuitarToken.sol#542)
        - >0.6.6 (GuitarToken.sol#855)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (GuitarToken.sol#513-539) is never used and should be removed
Address.functionCall(address,bytes) (GuitarToken.sol#460-462) is never used and should be removed
```

```
Address.functionCall(address,bytes,string) (GuitarToken.sol#470-476) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (GuitarToken.sol#489-495) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (GuitarToken.sol#503-511) is never used and should be removed
Address.isContract(address) (GuitarToken.sol#405-416) is never used and should be removed
Address.sendValue(address,uint256) (GuitarToken.sol#434-440) is never used and should be removed
BEP20._burn(address,uint256) (GuitarToken.sol#806-812) is never used and should be removed
BEP20._burnFrom(address,uint256) (GuitarToken.sol#845-852) is never used and should be removed
Context._msgData() (GuitarToken.sol#23-26) is never used and should be removed
SafeMath.div(uint256,uint256) (GuitarToken.sol#297-299) is never used and should be removed
SafeMath.div(uint256,uint256,string) (GuitarToken.sol#313-323) is never used and should be removed
SafeMath.min(uint256,uint256) (GuitarToken.sol#362-364) is never used and should be removed
SafeMath.mod(uint256,uint256) (GuitarToken.sol#337-339) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (GuitarToken.sol#353-360) is never used and should be removed
SafeMath.mul(uint256,uint256) (GuitarToken.sol#271-283) is never used and should be removed
SafeMath.sqrt(uint256) (GuitarToken.sol#367-378) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (GuitarToken.sol#5) is too complex
Pragma version>=0.6.0<0.8.0 (GuitarToken.sol#29) is too complex
Pragma version>=0.4.0 (GuitarToken.sol#96) allows old versions
Pragma version>=0.4.0 (GuitarToken.sol#193) allows old versions
Pragma version>=0.6.6 (GuitarToken.sol#382) allows old versions
Pragma version>=0.4.0 (GuitarToken.sol#542) allows old versions
Pragma version>0.6.6 (GuitarToken.sol#855) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (GuitarToken.sol#434-440):
        - (success) = recipient.call{value: amount}() (GuitarToken.sol#438)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (GuitarToken.sol#513-539):
        - (success,returndata) = target.call{value: weiValue}(data) (GuitarToken.sol#522)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter GuitarToken.mintFor(address,uint256)._to (GuitarToken.sol#862) is not in mixedCase
Parameter GuitarToken.mintFor(address,uint256)._amount (GuitarToken.sol#862) is not in mixedCase
Constant GuitarToken.maxSupply (GuitarToken.sol#860) is not in UPPER_CASE_WITH_UNDERSCORES
Variable GuitarToken._delegates (GuitarToken.sol#880) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
```

```
INFO:Detectors:
Parameter GuitarToken.mintFor(address,uint256)._to (GuitarToken.sol#862) is not in mixedCase
Parameter GuitarToken.mintFor(address,uint256)._amount (GuitarToken.sol#862) is not in mixedCase
Constant GuitarToken.maxSupply (GuitarToken.sol#860) is not in UPPER_CASE_WITH_UNDERSCORES
Variable GuitarToken._delegates (GuitarToken.sol#880) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (GuitarToken.sol#24)" inContext (GuitarToken.sol#18-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (GuitarToken.sol#80-83)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (GuitarToken.sol#89-93)
decimals() should be declared external:
        - BEP20.decimals() (GuitarToken.sol#615-617)
symbol() should be declared external:
        - BEP20.symbol() (GuitarToken.sol#622-624)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (GuitarToken.sol#648-651)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (GuitarToken.sol#656-658)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (GuitarToken.sol#667-670)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (GuitarToken.sol#684-696)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (GuitarToken.sol#710-713)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (GuitarToken.sol#729-736)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (GuitarToken.sol#746-749)
        - GuitarToken.mint(uint256) (GuitarToken.sol#868-872)
mintFor(address,uint256) should be declared external:
        - GuitarToken.mintFor(address,uint256) (GuitarToken.sol#862-866)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:GuitarToken.sol analyzed (7 contracts with 75 detectors), 51 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> GuitarSyrupBar.sol

```
INFO:Detectors:
GuitarSyrupBar.safeGuitarTransfer(address,uint256) (GuitarSyrupBar.sol#1137-1144) ignores return value by guitar.transfer(_to,guitarBal)
(GuitarSyrupBar.sol#1140)
GuitarSyrupBar.safeGuitarTransfer(address,uint256) (GuitarSyrupBar.sol#1137-1144) ignores return value by guitar.transfer(_to,_amount) (G
uitarSyrupBar.sol#1142)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
GuitarToken._writeCheckpoint(address,uint32,uint256,uint256) (GuitarSyrupBar.sol#1077-1095) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (GuitarSyrupBar.sol#1087)
GuitarSyrupBar._writeCheckpoint(address,uint32,uint256,uint256) (GuitarSyrupBar.sol#1345-1363) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (GuitarSyrupBar.sol#1355)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
BEP20.allowance(address,address).owner (GuitarSyrupBar.sol#661) shadows:
        - Ownable.owner() (GuitarSyrupBar.sol#66-68) (function)
BEP20._approve(address,address,uint256).owner (GuitarSyrupBar.sol#833) shadows:
        - Ownable.owner() (GuitarSyrupBar.sol#66-68) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GuitarToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (GuitarSyrupBar.sol#943-984) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (GuitarSyrupBar.sol#982)
GuitarSyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (GuitarSyrupBar.sol#1211-1252) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,GUT::delegateBySig: signature expired) (GuitarSyrupBar.sol#1250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (GuitarSyrupBar.sol#410-421) uses assembly
        - INLINE ASM (GuitarSyrupBar.sol#417-419)
Address._functionCallWithValue(address,bytes,uint256,string) (GuitarSyrupBar.sol#518-544) uses assembly
        - INLINE ASM (GuitarSyrupBar.sol#536-539)
GuitarToken.getChainId() (GuitarSyrupBar.sol#1102-1106) uses assembly
        - INLINE ASM (GuitarSyrupBar.sol#1104)
GuitarSyrupBar.getChainId() (GuitarSyrupBar.sol#1370-1374) uses assembly
        - INLINE ASM (GuitarSyrupBar.sol#1372)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
```

```
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.6.12', '>0.6.6', '>=0.4.0', '>=0.6.0<0.8.0', '>=0.6.6']
        - >=0.6.0<0.8.0 (GuitarSyrupBar.sol#11)
        - >=0.6.0<0.8.0 (GuitarSyrupBar.sol#34)
        - >=0.4.0 (GuitarSyrupBar.sol#101)
        - >=0.4.0 (GuitarSyrupBar.sol#198)
        - >=0.6.6 (GuitarSyrupBar.sol#387)
        - >=0.4.0 (GuitarSyrupBar.sol#547)
        - >0.6.6 (GuitarSyrupBar.sol#860)
        - 0.6.12 (GuitarSyrupBar.sol#1111)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (GuitarSyrupBar.sol#518-544) is never used and should be removed
Address.functionCall(address,bytes) (GuitarSyrupBar.sol#465-467) is never used and should be removed
Address.functionCall(address,bytes,string) (GuitarSyrupBar.sol#475-481) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (GuitarSyrupBar.sol#494-500) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (GuitarSyrupBar.sol#508-516) is never used and should be removed
Address.isContract(address) (GuitarSyrupBar.sol#410-421) is never used and should be removed
Address.sendValue(address,uint256) (GuitarSyrupBar.sol#439-445) is never used and should be removed
BEP20._burnFrom(address,uint256) (GuitarSyrupBar.sol#850-857) is never used and should be removed
Context._msgData() (GuitarSyrupBar.sol#28-31) is never used and should be removed
SafeMath.div(uint256,uint256) (GuitarSyrupBar.sol#302-304) is never used and should be removed
SafeMath.div(uint256,uint256,string) (GuitarSyrupBar.sol#318-328) is never used and should be removed
SafeMath.min(uint256,uint256) (GuitarSyrupBar.sol#367-369) is never used and should be removed
SafeMath.mod(uint256,uint256) (GuitarSyrupBar.sol#342-344) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (GuitarSyrupBar.sol#358-365) is never used and should be removed
SafeMath.mul(uint256,uint256) (GuitarSyrupBar.sol#276-288) is never used and should be removed
SafeMath.sqrt(uint256) (GuitarSyrupBar.sol#372-383) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (GuitarSyrupBar.sol#11) is too complex
Pragma version>=0.6.0<0.8.0 (GuitarSyrupBar.sol#34) is too complex
Pragma version>=0.4.0 (GuitarSyrupBar.sol#101) allows old versions
Pragma version>=0.4.0 (GuitarSyrupBar.sol#198) allows old versions
Pragma version>=0.6.6 (GuitarSyrupBar.sol#387) allows old versions
Pragma version>=0.4.0 (GuitarSyrupBar.sol#547) allows old versions
Pragma version>0.6.6 (GuitarSyrupBar.sol#860) allows old versions
```

```
Pragma version>=0.4.0 (GuitarSyrupBar.sol#547) allows old versions
Pragma version>0.6.6 (GuitarSyrupBar.sol#860) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (GuitarSyrupBar.sol#439-445):
        - (success) = recipient.call{value: amount}() (GuitarSyrupBar.sol#443)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (GuitarSyrupBar.sol#518-544):
        - (success,returndata) = target.call{value: weiValue}(data) (GuitarSyrupBar.sol#527)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter GuitarToken.mintFor(address,uint256)._to (GuitarSyrupBar.sol#868) is not in mixedCase
Parameter GuitarToken.mintFor(address,uint256)._amount (GuitarSyrupBar.sol#868) is not in mixedCase
Constant GuitarToken.maxSupply (GuitarSyrupBar.sol#865) is not in UPPER_CASE_WITH_UNDERSCORES
Variable GuitarToken._delegates (GuitarSyrupBar.sol#886) is not in mixedCase
Parameter GuitarSyrupBar.mint(address,uint256)._to (GuitarSyrupBar.sol#1117) is not in mixedCase
Parameter GuitarSyrupBar.mint(address,uint256)._amount (GuitarSyrupBar.sol#1117) is not in mixedCase
Parameter GuitarSyrupBar.burn(address,uint256)._from (GuitarSyrupBar.sol#1122) is not in mixedCase
Parameter GuitarSyrupBar.burn(address,uint256)._amount (GuitarSyrupBar.sol#1122) is not in mixedCase
Parameter GuitarSyrupBar.safeGuitarTransfer(address,uint256)._to (GuitarSyrupBar.sol#1137) is not in mixedCase
Parameter GuitarSyrupBar.safeGuitarTransfer(address,uint256)._amount (GuitarSyrupBar.sol#1137) is not in mixedCase
Variable GuitarSyrupBar._delegates (GuitarSyrupBar.sol#1153) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (GuitarSyrupBar.sol#29)" inContext (GuitarSyrupBar.sol#23-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (GuitarSyrupBar.sol#85-88)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (GuitarSyrupBar.sol#94-98)
```

```
decimals() should be declared external:
        - BEP20.decimals() (GuitarSyrupBar.sol#620-622)
symbol() should be declared external:
        - BEP20.symbol() (GuitarSyrupBar.sol#627-629)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (GuitarSyrupBar.sol#653-656)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (GuitarSyrupBar.sol#661-663)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (GuitarSyrupBar.sol#672-675)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (GuitarSyrupBar.sol#689-701)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (GuitarSyrupBar.sol#715-718)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (GuitarSyrupBar.sol#734-741)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (GuitarSyrupBar.sol#751-754)
        - GuitarToken.mint(uint256) (GuitarSyrupBar.sol#874-878)
mintFor(address,uint256) should be declared external:
        - GuitarToken.mintFor(address,uint256) (GuitarSyrupBar.sol#868-872)
mint(address,uint256) should be declared external:
        - GuitarSyrupBar.mint(address,uint256) (GuitarSyrupBar.sol#1117-1120)
burn(address,uint256) should be declared external:
        - GuitarSyrupBar.burn(address,uint256) (GuitarSyrupBar.sol#1122-1125)
safeGuitarTransfer(address,uint256) should be declared external:
        - GuitarSyrupBar.safeGuitarTransfer(address,uint256) (GuitarSyrupBar.sol#1137-1144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:GuitarSyrupBar.sol analyzed (8 contracts with 75 detectors), 65 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Slither log >> GuitarMasterChef.sol

```
INFO:Detectors:
GuitarSyrupBar.safeGuitarTransfer(address,uint256) (GuitarMasterChef.sol#1296-1303) ignores return value by guitar.transfer(_to,guitarBal
) (GuitarMasterChef.sol#1299)
GuitarSyrupBar.safeGuitarTransfer(address,uint256) (GuitarMasterChef.sol#1296-1303) ignores return value by guitar.transfer(_to,_amount)
(GuitarMasterChef.sol#1301)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
GuitarMasterChef.pendingGuitar(uint256,address) (GuitarMasterChef.sol#1750-1772) performs a multiplication on the result of a division:
        -guitarReward = multiplier.mul(guitarPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (GuitarMasterChef.sol#1763-1766)
        -accGuitarPerShare = accGuitarPerShare.add(guitarReward.mul(1e12).div(pool.lpSupply)) (GuitarMasterChef.sol#1767-1769)
GuitarMasterChef.updatePool(uint256) (GuitarMasterChef.sol#1783-1804) performs a multiplication on the result of a division:
        -guitarReward = multiplier.mul(guitarPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (GuitarMasterChef.sol#1794-1797)
        -pool.accGuitarPerShare = pool.accGuitarPerShare.add(guitarReward.mul(1e12).div(pool.lpSupply)) (GuitarMasterChef.sol#1800-1802)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
GuitarToken._writeCheckpoint(address,uint32,uint256,uint256) (GuitarMasterChef.sol#1236-1254) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (GuitarMasterChef.sol#1246)
GuitarSyrupBar._writeCheckpoint(address,uint32,uint256,uint256) (GuitarMasterChef.sol#1504-1522) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (GuitarMasterChef.sol#1514)
GuitarMasterChef.updatePool(uint256) (GuitarMasterChef.sol#1783-1804) uses a dangerous strict equality:
        - pool.lpSupply == 0 || totalAllocPoint == 0 (GuitarMasterChef.sol#1789)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in GuitarMasterChef.add(uint256,IBEP20,uint16,bool) (GuitarMasterChef.sol#1667-1698):
        External calls:
        - massUpdatePools() (GuitarMasterChef.sol#1680)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        State variables written after the call(s):
        - poolExistence[_lpToken] = true (GuitarMasterChef.sol#1686)
        - poolInfo.push(PoolInfo(_lpToken,0,_allocPoint,lastRewardBlock,0,_depositFeeBP)) (GuitarMasterChef.sol#1687-1696)
        - updateStakingPool() (GuitarMasterChef.sol#1697)
                - poolInfo[0].allocPoint = points (GuitarMasterChef.sol#1736)
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (GuitarMasterChef.sol#1685)
        - updateStakingPool() (GuitarMasterChef.sol#1697)
                - totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (GuitarMasterChef.sol#1733-1735)
Reentrancy in GuitarMasterChef.deposit(uint256,uint256) (GuitarMasterChef.sol#1807-1847):
```

```
        External calls:
        - updatePool(_pid) (GuitarMasterChef.sol#1815)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1824)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (GuitarMasterChef.sol#1829-1833)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (GuitarMasterChef.sol#1839)
        State variables written after the call(s):
        - pool.lpSupply = pool.lpSupply.add(_amount).sub(depositFee) (GuitarMasterChef.sol#1843)
        - user.amount = user.amount.add(_amount).sub(depositFee) (GuitarMasterChef.sol#1842)
        - user.rewardDebt = user.amount.mul(pool.accGuitarPerShare).div(1e12) (GuitarMasterChef.sol#1845)
Reentrancy in GuitarMasterChef.emergencyWithdraw(uint256) (GuitarMasterChef.sol#1937-1945):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (GuitarMasterChef.sol#1941)
        State variables written after the call(s):
        - user.amount = 0 (GuitarMasterChef.sol#1943)
        - user.rewardDebt = 0 (GuitarMasterChef.sol#1944)
Reentrancy in GuitarMasterChef.enterStaking(uint256) (GuitarMasterChef.sol#1874-1910):
        External calls:
        - updatePool(0) (GuitarMasterChef.sol#1877)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1885)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (GuitarMasterChef.sol#1890-1894)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (GuitarMasterChef.sol#1900)
        State variables written after the call(s):
        - pool.lpSupply = pool.lpSupply.add(_amount).sub(depositFee) (GuitarMasterChef.sol#1904)
        - user.amount = user.amount.add(_amount).sub(depositFee) (GuitarMasterChef.sol#1903)
        - user.rewardDebt = user.amount.mul(pool.accGuitarPerShare).div(1e12) (GuitarMasterChef.sol#1906)
Reentrancy in GuitarMasterChef.leaveStaking(uint256) (GuitarMasterChef.sol#1913-1934):
        External calls:
        - updatePool(0) (GuitarMasterChef.sol#1918)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1923)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
```

```
        State variables written after the call(s):
        - pool.lpSupply = pool.lpSupply.sub(_amount) (GuitarMasterChef.sol#1927)
        - user.amount = user.amount.sub(_amount) (GuitarMasterChef.sol#1926)
Reentrancy in GuitarMasterChef.leaveStaking(uint256) (GuitarMasterChef.sol#1913-1934):
        External calls:
        - updatePool(0) (GuitarMasterChef.sol#1918)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1923)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (GuitarMasterChef.sol#1928)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accGuitarPerShare).div(1e12) (GuitarMasterChef.sol#1930)
Reentrancy in GuitarMasterChef.set(uint256,uint256,uint16,bool) (GuitarMasterChef.sol#1701-1723):
        External calls:
        - massUpdatePools() (GuitarMasterChef.sol#1712)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        State variables written after the call(s):
        - poolInfo[_pid].allocPoint = _allocPoint (GuitarMasterChef.sol#1715)
        - poolInfo[_pid].depositFeeBP = _depositFeeBP (GuitarMasterChef.sol#1716)
        - updateStakingPool() (GuitarMasterChef.sol#1721)
                - poolInfo[0].allocPoint = points (GuitarMasterChef.sol#1736)
        - totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint) (GuitarMasterChef.sol#1718-1720)
        - updateStakingPool() (GuitarMasterChef.sol#1721)
                - totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (GuitarMasterChef.sol#1733-1735)
Reentrancy in GuitarMasterChef.updateEmissionRate(uint256) (GuitarMasterChef.sol#1966-1971):
        External calls:
        - massUpdatePools() (GuitarMasterChef.sol#1968)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        State variables written after the call(s):
        - guitarPerBlock = _guitarPerBlock (GuitarMasterChef.sol#1969)
Reentrancy in GuitarMasterChef.updatePool(uint256) (GuitarMasterChef.sol#1783-1804):
        External calls:
        - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
        - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        State variables written after the call(s):
```

```
        - pool.accGuitarPerShare = pool.accGuitarPerShare.add(guitarReward.mul(1e12).div(pool.lpSupply)) (GuitarMasterChef.sol#1800-1802)
        - pool.lastRewardBlock = block.number (GuitarMasterChef.sol#1803)
Reentrancy in GuitarMasterChef.withdraw(uint256,uint256) (GuitarMasterChef.sol#1850-1871):
        External calls:
        - updatePool(_pid) (GuitarMasterChef.sol#1857)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1862)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
        State variables written after the call(s):
        - pool.lpSupply = pool.lpSupply.sub(_amount) (GuitarMasterChef.sol#1866)
        - user.amount = user.amount.sub(_amount) (GuitarMasterChef.sol#1865)
Reentrancy in GuitarMasterChef.withdraw(uint256,uint256) (GuitarMasterChef.sol#1850-1871):
        External calls:
        - updatePool(_pid) (GuitarMasterChef.sol#1857)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1862)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (GuitarMasterChef.sol#1867)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accGuitarPerShare).div(1e12) (GuitarMasterChef.sol#1869)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
GuitarMasterChef.add(uint256,IBEP20,uint16,bool) (GuitarMasterChef.sol#1667-1698) ignores return value by _lpToken.balanceOf(address(this
)) (GuitarMasterChef.sol#1677)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
BEP20.allowance(address,address).owner (GuitarMasterChef.sol#820) shadows:
        - Ownable.owner() (GuitarMasterChef.sol#264-266) (function)
BEP20._approve(address,address,uint256).owner (GuitarMasterChef.sol#992) shadows:
        - Ownable.owner() (GuitarMasterChef.sol#264-266) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GuitarMasterChef.constructor(GuitarToken,GuitarSyrupBar,address,address,uint256,uint256)._devaddr (GuitarMasterChef.sol#1624) lacks a zer
o-check on :
                - devaddr = _devaddr (GuitarMasterChef.sol#1631)
GuitarMasterChef.constructor(GuitarToken,GuitarSyrupBar,address,address,uint256,uint256)._feeAddress (GuitarMasterChef.sol#1625) lacks a
```

```
zero-check on :
                - feeAddress = _feeAddress (GuitarMasterChef.sol#1632)
GuitarMasterChef.dev(address)._devaddr (GuitarMasterChef.sol#1953) lacks a zero-check on :
                - devaddr = _devaddr (GuitarMasterChef.sol#1955)
GuitarMasterChef.setFeeAddress(address)._feeAddress (GuitarMasterChef.sol#1959) lacks a zero-check on :
                - feeAddress = _feeAddress (GuitarMasterChef.sol#1961)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in GuitarMasterChef.deposit(uint256,uint256) (GuitarMasterChef.sol#1807-1847):
        External calls:
        - updatePool(_pid) (GuitarMasterChef.sol#1815)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1824)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (GuitarMasterChef.sol#1829-1833)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (GuitarMasterChef.sol#1839)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (GuitarMasterChef.sol#1846)
Reentrancy in GuitarMasterChef.emergencyWithdraw(uint256) (GuitarMasterChef.sol#1937-1945):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (GuitarMasterChef.sol#1941)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,user.amount) (GuitarMasterChef.sol#1942)
Reentrancy in GuitarMasterChef.enterStaking(uint256) (GuitarMasterChef.sol#1874-1910):
        External calls:
        - updatePool(0) (GuitarMasterChef.sol#1877)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1885)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (GuitarMasterChef.sol#1890-1894)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (GuitarMasterChef.sol#1900)
        - syrup.mint(msg.sender,_amount) (GuitarMasterChef.sol#1908)
        Event emitted after the call(s):
        - Deposit(msg.sender,0,_amount) (GuitarMasterChef.sol#1909)
Reentrancy in GuitarMasterChef.leaveStaking(uint256) (GuitarMasterChef.sol#1913-1934):
        External calls:
```

```
        - updatePool(0) (GuitarMasterChef.sol#1918)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1923)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (GuitarMasterChef.sol#1928)
        - syrup.burn(msg.sender,_amount) (GuitarMasterChef.sol#1932)
        Event emitted after the call(s):
        - Withdraw(msg.sender,0,_amount) (GuitarMasterChef.sol#1933)
Reentrancy in GuitarMasterChef.updateEmissionRate(uint256) (GuitarMasterChef.sol#1966-1971):
        External calls:
        - massUpdatePools() (GuitarMasterChef.sol#1968)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        Event emitted after the call(s):
        - UpdateEmissionRate(msg.sender,_guitarPerBlock) (GuitarMasterChef.sol#1970)
Reentrancy in GuitarMasterChef.withdraw(uint256,uint256) (GuitarMasterChef.sol#1850-1871):
        External calls:
        - updatePool(_pid) (GuitarMasterChef.sol#1857)
                - guitar.mintFor(devaddr,guitarReward.div(10)) (GuitarMasterChef.sol#1798)
                - guitar.mintFor(address(syrup),guitarReward) (GuitarMasterChef.sol#1799)
        - safeGuitarTransfer(msg.sender,pending) (GuitarMasterChef.sol#1862)
                - syrup.safeGuitarTransfer(_to,_amount) (GuitarMasterChef.sol#1949)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (GuitarMasterChef.sol#1867)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (GuitarMasterChef.sol#1870)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
GuitarToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (GuitarMasterChef.sol#1102-1143) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,CAKE::delegateBySig: signature expired) (GuitarMasterChef.sol#1141)
GuitarSyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (GuitarMasterChef.sol#1370-1411) uses timestamp for compariso
ns
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,GUT::delegateBySig: signature expired) (GuitarMasterChef.sol#1409)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (GuitarMasterChef.sol#435-444) uses assembly
```

```
        - INLINE ASM (GuitarMasterChef.sol#442)
Address._verifyCallResult(bool,bytes,string) (GuitarMasterChef.sol#580-597) uses assembly
        - INLINE ASM (GuitarMasterChef.sol#589-592)
GuitarToken.getChainId() (GuitarMasterChef.sol#1261-1265) uses assembly
        - INLINE ASM (GuitarMasterChef.sol#1263)
GuitarSyrupBar.getChainId() (GuitarMasterChef.sol#1529-1533) uses assembly
        - INLINE ASM (GuitarMasterChef.sol#1531)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
GuitarMasterChef.nonDuplicated(IBEP20) (GuitarMasterChef.sol#1660-1663) compares to a boolean constant:
        -require(bool,string)(poolExistence[_lpToken] == false,nonDuplicated: duplicated) (GuitarMasterChef.sol#1661)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.6.12', '>0.6.6', '>=0.4.0', '>=0.6.2<0.8.0', '^0.6.0']
        - >=0.4.0 (GuitarMasterChef.sol#9)
        - >=0.4.0 (GuitarMasterChef.sol#201)
        - >=0.4.0 (GuitarMasterChef.sol#232)
        - >=0.4.0 (GuitarMasterChef.sol#310)
        - >=0.6.2<0.8.0 (GuitarMasterChef.sol#412)
        - ^0.6.0 (GuitarMasterChef.sol#604)
        - >=0.4.0 (GuitarMasterChef.sol#706)
        - >0.6.6 (GuitarMasterChef.sol#1019)
        - 0.6.12 (GuitarMasterChef.sol#1270)
        - 0.6.12 (GuitarMasterChef.sol#1538)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (GuitarMasterChef.sol#488-490) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (GuitarMasterChef.sol#513-515) is never used and should be removed
Address.functionDelegateCall(address,bytes) (GuitarMasterChef.sol#562-564) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (GuitarMasterChef.sol#572-578) is never used and should be removed
Address.functionStaticCall(address,bytes) (GuitarMasterChef.sol#538-540) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (GuitarMasterChef.sol#548-554) is never used and should be removed
Address.sendValue(address,uint256) (GuitarMasterChef.sol#462-468) is never used and should be removed
BEP20._burnFrom(address,uint256) (GuitarMasterChef.sol#1009-1016) is never used and should be removed
Context._msgData() (GuitarMasterChef.sol#222-225) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (GuitarMasterChef.sol#646-660) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (GuitarMasterChef.sol#671-681) is never used and should be removed
```

```
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (GuitarMasterChef.sol#662-669) is never used and should be removed
SafeMath.min(uint256,uint256) (GuitarMasterChef.sol#178-180) is never used and should be removed
SafeMath.mod(uint256,uint256) (GuitarMasterChef.sol#153-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (GuitarMasterChef.sol#169-176) is never used and should be removed
SafeMath.sqrt(uint256) (GuitarMasterChef.sol#183-194) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.4.0 (GuitarMasterChef.sol#9) allows old versions
Pragma version>=0.4.0 (GuitarMasterChef.sol#201) allows old versions
Pragma version>=0.4.0 (GuitarMasterChef.sol#232) allows old versions
Pragma version>=0.4.0 (GuitarMasterChef.sol#310) allows old versions
Pragma version>=0.6.2<0.8.0 (GuitarMasterChef.sol#412) is too complex
Pragma version^0.6.0 (GuitarMasterChef.sol#604) allows old versions
Pragma version>=0.4.0 (GuitarMasterChef.sol#706) allows old versions
Pragma version>0.6.6 (GuitarMasterChef.sol#1019) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (GuitarMasterChef.sol#462-468):
        - (success) = recipient.call{value: amount}() (GuitarMasterChef.sol#466)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (GuitarMasterChef.sol#523-530):
        - (success,returndata) = target.call{value: value}(data) (GuitarMasterChef.sol#528)
Low level call in Address.functionStaticCall(address,bytes,string) (GuitarMasterChef.sol#548-554):
        - (success,returndata) = target.staticcall(data) (GuitarMasterChef.sol#552)
Low level call in Address.functionDelegateCall(address,bytes,string) (GuitarMasterChef.sol#572-578):
        - (success,returndata) = target.delegatecall(data) (GuitarMasterChef.sol#576)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter GuitarToken.mintFor(address,uint256)._to (GuitarMasterChef.sol#1027) is not in mixedCase
Parameter GuitarToken.mintFor(address,uint256)._amount (GuitarMasterChef.sol#1027) is not in mixedCase
Constant GuitarToken.maxSupply (GuitarMasterChef.sol#1024) is not in UPPER_CASE_WITH_UNDERSCORES
Variable GuitarToken._delegates (GuitarMasterChef.sol#1045) is not in mixedCase
Parameter GuitarSyrupBar.mint(address,uint256)._to (GuitarMasterChef.sol#1276) is not in mixedCase
Parameter GuitarSyrupBar.mint(address,uint256)._amount (GuitarMasterChef.sol#1276) is not in mixedCase
Parameter GuitarSyrupBar.burn(address,uint256)._from (GuitarMasterChef.sol#1281) is not in mixedCase
Parameter GuitarSyrupBar.burn(address,uint256)._amount (GuitarMasterChef.sol#1281) is not in mixedCase
```

```
Parameter GuitarSyrupBar.safeGuitarTransfer(address,uint256)._to (GuitarMasterChef.sol#1296) is not in mixedCase
Parameter GuitarSyrupBar.safeGuitarTransfer(address,uint256)._amount (GuitarMasterChef.sol#1296) is not in mixedCase
Variable GuitarSyrupBar._delegates (GuitarMasterChef.sol#1312) is not in mixedCase
Parameter GuitarMasterChef.add(uint256,IBEP20,uint16,bool)._allocPoint (GuitarMasterChef.sol#1668) is not in mixedCase
Parameter GuitarMasterChef.add(uint256,IBEP20,uint16,bool)._lpToken (GuitarMasterChef.sol#1669) is not in mixedCase
Parameter GuitarMasterChef.add(uint256,IBEP20,uint16,bool)._depositFeeBP (GuitarMasterChef.sol#1670) is not in mixedCase
Parameter GuitarMasterChef.add(uint256,IBEP20,uint16,bool)._withUpdate (GuitarMasterChef.sol#1671) is not in mixedCase
Parameter GuitarMasterChef.set(uint256,uint256,uint16,bool)._pid (GuitarMasterChef.sol#1702) is not in mixedCase
Parameter GuitarMasterChef.set(uint256,uint256,uint16,bool)._allocPoint (GuitarMasterChef.sol#1703) is not in mixedCase
Parameter GuitarMasterChef.set(uint256,uint256,uint16,bool)._depositFeeBP (GuitarMasterChef.sol#1704) is not in mixedCase
Parameter GuitarMasterChef.set(uint256,uint256,uint16,bool)._withUpdate (GuitarMasterChef.sol#1705) is not in mixedCase
Parameter GuitarMasterChef.getMultiplier(uint256,uint256)._from (GuitarMasterChef.sol#1741) is not in mixedCase
Parameter GuitarMasterChef.getMultiplier(uint256,uint256)._to (GuitarMasterChef.sol#1741) is not in mixedCase
Parameter GuitarMasterChef.pendingGuitar(uint256,address)._pid (GuitarMasterChef.sol#1750) is not in mixedCase
Parameter GuitarMasterChef.pendingGuitar(uint256,address)._user (GuitarMasterChef.sol#1750) is not in mixedCase
Parameter GuitarMasterChef.updatePool(uint256)._pid (GuitarMasterChef.sol#1783) is not in mixedCase
Parameter GuitarMasterChef.deposit(uint256,uint256)._pid (GuitarMasterChef.sol#1808) is not in mixedCase
Parameter GuitarMasterChef.deposit(uint256,uint256)._amount (GuitarMasterChef.sol#1809) is not in mixedCase
Parameter GuitarMasterChef.withdraw(uint256,uint256)._pid (GuitarMasterChef.sol#1850) is not in mixedCase
Parameter GuitarMasterChef.withdraw(uint256,uint256)._amount (GuitarMasterChef.sol#1850) is not in mixedCase
Parameter GuitarMasterChef.enterStaking(uint256)._amount (GuitarMasterChef.sol#1874) is not in mixedCase
Parameter GuitarMasterChef.leaveStaking(uint256)._amount (GuitarMasterChef.sol#1913) is not in mixedCase
Parameter GuitarMasterChef.emergencyWithdraw(uint256)._pid (GuitarMasterChef.sol#1937) is not in mixedCase
Parameter GuitarMasterChef.safeGuitarTransfer(address,uint256)._to (GuitarMasterChef.sol#1948) is not in mixedCase
Parameter GuitarMasterChef.safeGuitarTransfer(address,uint256)._amount (GuitarMasterChef.sol#1948) is not in mixedCase
Parameter GuitarMasterChef.dev(address)._devaddr (GuitarMasterChef.sol#1953) is not in mixedCase
Parameter GuitarMasterChef.setFeeAddress(address)._feeAddress (GuitarMasterChef.sol#1959) is not in mixedCase
Parameter GuitarMasterChef.updateEmissionRate(uint256)._guitarPerBlock (GuitarMasterChef.sol#1966) is not in mixedCase
Variable GuitarMasterChef.BONUS_MULTIPLIER (GuitarMasterChef.sol#1597) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (GuitarMasterChef.sol#223)" inContext (GuitarMasterChef.sol#213-226)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (GuitarMasterChef.sol#283-286)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (GuitarMasterChef.sol#292-294)
```

```
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (GuitarMasterChef.sol#292-294)
decimals() should be declared external:
        - BEP20.decimals() (GuitarMasterChef.sol#779-781)
symbol() should be declared external:
        - BEP20.symbol() (GuitarMasterChef.sol#786-788)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (GuitarMasterChef.sol#812-815)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (GuitarMasterChef.sol#820-822)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (GuitarMasterChef.sol#831-834)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (GuitarMasterChef.sol#848-860)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (GuitarMasterChef.sol#874-877)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (GuitarMasterChef.sol#893-900)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (GuitarMasterChef.sol#910-913)
        - GuitarToken.mint(uint256) (GuitarMasterChef.sol#1033-1037)
mintFor(address,uint256) should be declared external:
        - GuitarToken.mintFor(address,uint256) (GuitarMasterChef.sol#1027-1031)
mint(address,uint256) should be declared external:
        - GuitarSyrupBar.mint(address,uint256) (GuitarMasterChef.sol#1276-1279)
burn(address,uint256) should be declared external:
        - GuitarSyrupBar.burn(address,uint256) (GuitarMasterChef.sol#1281-1284)
safeGuitarTransfer(address,uint256) should be declared external:
        - GuitarSyrupBar.safeGuitarTransfer(address,uint256) (GuitarMasterChef.sol#1296-1303)
updateMultiplier(uint256) should be declared external:
        - GuitarMasterChef.updateMultiplier(uint256) (GuitarMasterChef.sol#1651-1653)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:GuitarMasterChef.sol analyzed (10 contracts with 75 detectors), 121 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
root@server:/chetan/gaza/mycontracts#
```

Solidity Static Analysis

GuitarToken.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 513:4:

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: 412:8:

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: 531:16:

Gas & Economy

Gas costs:

Gas requirement of function BEP20.transferOwnership 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: 89:4:

Gas costs:

Gas requirement of function GuitarToken.transferOwnership 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: 89:4:

Gas costs:

Gas requirement of function BEP20.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: 608:4:

Gas costs:

Gas requirement of function BEP20.transferFrom 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: 684:4:

Gas costs:

Gas requirement of function GuitarToken.transferFrom 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: 684:4:

Gas costs:

Gas requirement of function BEP20.increaseAllowance 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: 710:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 107:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 236:4:

Constant/View/Pure functions:

SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 297:4:

Constant/View/Pure functions:

SafeMath.mod(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 337:4:

Similar variable names:

BEP20._burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 846:23:

Similar variable names:

BEP20._burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 848:12:

Similar variable names:

BEP20._burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 850:24:

Similar variable names:

BEP20._burnFrom(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 850:51:

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: 370:24:

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: 373:20:

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: 373:21:

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: 1026:36:

GuitarSyrupBar.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 510:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in GuitarSyrupBar.safeGuitarTransfer(address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1129:4:

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: 409:8:

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: 1364:8:

Block timestamp:

Use of "now": "now" does not mean current time. "now" is an alias for "block.timestamp". "block.timestamp" can be influenced by miners to a certain degree, be careful.
more
Pos: 1242:16:

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: 974:16:

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: 435:27:

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: 519:50:

Gas & Economy

Gas costs:

Gas requirement of function BEP20.transferOwnership 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: 86:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 104:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 233:4:

Constant/View/Pure functions:

SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 294:4:

Similar variable names:

GuitarSyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) : Variables have very similar names "nonce" and "nonces". Note: Modifiers are currently not considered by this static analysis.
Pos: 1241:16:

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: 1292:36:

GuitarMasterChef.sol

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 517:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeBEP20.safeApprove(contract IBEP20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 640:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeBEP20.safeIncreaseAllowance(contract IBEP20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 656:4:

Gas & Economy

Gas costs:

Gas requirement of function BEP20.transferOwnership 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: 286:4:

Gas costs:

Gas requirement of function GuitarMasterChef.set 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: 1695:4:

Gas costs:

Gas requirement of function GuitarMasterChef.getMultiplier 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: 1735:4:

Gas costs:

Gas requirement of function GuitarMasterChef.pendingGuitar 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: 1744:4:

Gas costs:

Gas requirement of function GuitarMasterChef.updateEmissionRate 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: 1960:4:

ERC

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 315:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 46:4:

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: 1655:8:

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: 1667:8:

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: 1701:8:

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: 1805:8:

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: 1848:8:

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: 1849:8:

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: 1910:8:

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: 1911:8:

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: 183:21:

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: 1185:36:

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: 1453:36:

Solhint Linter

GuitarToken.sol

```
GuitarToken.sol:6:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
GuitarToken.sol:29:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
GuitarToken.sol:96:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarToken.sol:193:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarToken.sol:221:25: Error: Use double quotes for string literals
GuitarToken.sol:237:26: Error: Use double quotes for string literals
GuitarToken.sol:280:29: Error: Use double quotes for string literals
GuitarToken.sol:298:26: Error: Use double quotes for string literals
GuitarToken.sol:338:26: Error: Use double quotes for string literals
GuitarToken.sol:382:1: Error: Compiler version >=0.6.6 does not
satisfy the r semver requirement
GuitarToken.sol:435:50: Error: Use double quotes for string literals
GuitarToken.sol:438:58: Error: Use double quotes for string literals
GuitarToken.sol:439:26: Error: Use double quotes for string literals
GuitarToken.sol:461:43: Error: Use double quotes for string literals
GuitarToken.sol:494:59: Error: Use double quotes for string literals
GuitarToken.sol:509:49: Error: Use double quotes for string literals
GuitarToken.sol:519:37: Error: Use double quotes for string literals
GuitarToken.sol:542:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarToken.sol:693:59: Error: Use double quotes for string literals
GuitarToken.sol:733:69: Error: Use double quotes for string literals
GuitarToken.sol:770:39: Error: Use double quotes for string literals
GuitarToken.sol:771:42: Error: Use double quotes for string literals
GuitarToken.sol:773:59: Error: Use double quotes for string literals
GuitarToken.sol:788:40: Error: Use double quotes for string literals
GuitarToken.sol:807:40: Error: Use double quotes for string literals
GuitarToken.sol:809:61: Error: Use double quotes for string literals
GuitarToken.sol:832:38: Error: Use double quotes for string literals
GuitarToken.sol:833:40: Error: Use double quotes for string literals
GuitarToken.sol:850:60: Error: Use double quotes for string literals
GuitarToken.sol:855:1: Error: Compiler version >0.6.6 does not
satisfy the r semver requirement
GuitarToken.sol:858:31: Error: Use double quotes for string literals
GuitarToken.sol:858:51: Error: Use double quotes for string literals
GuitarToken.sol:860:29: Error: Constant name must be in capitalized
SNAKE_CASE
GuitarToken.sol:976:17: Error: Avoid to make time-based decisions in
your business logic
GuitarToken.sol:1098:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

GuitarSyrupBar.sol

```
GuitarSyrupBar.sol:3:1: Error: Compiler version >=0.6.0 <0.8.0 does not
satisfy the r semver requirement
GuitarSyrupBar.sol:26:1: Error: Compiler version >=0.6.0 <0.8.0 does
not satisfy the r semver requirement
GuitarSyrupBar.sol:93:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarSyrupBar.sol:190:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarSyrupBar.sol:218:25: Error: Use double quotes for string literals
GuitarSyrupBar.sol:234:26: Error: Use double quotes for string literals
GuitarSyrupBar.sol:277:29: Error: Use double quotes for string literals
GuitarSyrupBar.sol:295:26: Error: Use double quotes for string literals
GuitarSyrupBar.sol:335:26: Error: Use double quotes for string literals
GuitarSyrupBar.sol:379:1: Error: Compiler version >=0.6.6 does not
satisfy the r semver requirement
GuitarSyrupBar.sol:432:50: Error: Use double quotes for string literals
GuitarSyrupBar.sol:435:58: Error: Use double quotes for string literals
GuitarSyrupBar.sol:436:26: Error: Use double quotes for string literals
GuitarSyrupBar.sol:458:43: Error: Use double quotes for string literals
GuitarSyrupBar.sol:491:59: Error: Use double quotes for string literals
GuitarSyrupBar.sol:506:49: Error: Use double quotes for string literals
GuitarSyrupBar.sol:516:37: Error: Use double quotes for string literals
GuitarSyrupBar.sol:539:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarSyrupBar.sol:690:59: Error: Use double quotes for string literals
GuitarSyrupBar.sol:730:69: Error: Use double quotes for string literals
GuitarSyrupBar.sol:767:39: Error: Use double quotes for string literals
GuitarSyrupBar.sol:768:42: Error: Use double quotes for string literals
GuitarSyrupBar.sol:770:59: Error: Use double quotes for string literals
GuitarSyrupBar.sol:785:40: Error: Use double quotes for string literals
GuitarSyrupBar.sol:804:40: Error: Use double quotes for string literals
GuitarSyrupBar.sol:806:61: Error: Use double quotes for string literals
GuitarSyrupBar.sol:829:38: Error: Use double quotes for string literals
GuitarSyrupBar.sol:830:40: Error: Use double quotes for string literals
GuitarSyrupBar.sol:847:60: Error: Use double quotes for string literals
GuitarSyrupBar.sol:852:1: Error: Compiler version >0.6.6 does not
satisfy the r semver requirement
GuitarSyrupBar.sol:855:31: Error: Use double quotes for string literals
GuitarSyrupBar.sol:855:51: Error: Use double quotes for string literals
GuitarSyrupBar.sol:857:29: Error: Constant name must be in capitalized
SNAKE_CASE
GuitarSyrupBar.sol:974:17: Error: Avoid to make time-based decisions in
your business logic
GuitarSyrupBar.sol:1096:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
GuitarSyrupBar.sol:1103:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
GuitarSyrupBar.sol:1107:34: Error: Use double quotes for string
literals
GuitarSyrupBar.sol:1107:52: Error: Use double quotes for string
literals
GuitarSyrupBar.sol:1242:17: Error: Avoid to make time-based decisions
in your business logic
```

```
GuitarSyrupBar.sol:1364:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
```

GuitarMasterChef.sol

```
GuitarMasterChef.sol:3:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarMasterChef.sol:31:25: Error: Use double quotes for string
literals
GuitarMasterChef.sol:47:26: Error: Use double quotes for string
literals
GuitarMasterChef.sol:90:29: Error: Use double quotes for string
literals
GuitarMasterChef.sol:108:26: Error: Use double quotes for string
literals
GuitarMasterChef.sol:148:26: Error: Use double quotes for string
literals
GuitarMasterChef.sol:195:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarMasterChef.sol:210:28: Error: Code contains empty blocks
GuitarMasterChef.sol:226:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarMasterChef.sol:266:41: Error: Use double quotes for string
literals
GuitarMasterChef.sol:294:41: Error: Use double quotes for string
literals
GuitarMasterChef.sol:304:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarMasterChef.sol:406:1: Error: Compiler version >=0.6.2 <0.8.0 does
not satisfy the r semver requirement
GuitarMasterChef.sol:598:1: Error: Compiler version ^0.6.0 does not
satisfy the r semver requirement
GuitarMasterChef.sol:700:1: Error: Compiler version >=0.4.0 does not
satisfy the r semver requirement
GuitarMasterChef.sol:851:59: Error: Use double quotes for string
literals
GuitarMasterChef.sol:891:69: Error: Use double quotes for string
literals
GuitarMasterChef.sol:928:39: Error: Use double quotes for string
literals
GuitarMasterChef.sol:929:42: Error: Use double quotes for string
literals
GuitarMasterChef.sol:931:59: Error: Use double quotes for string
literals
GuitarMasterChef.sol:946:40: Error: Use double quotes for string
literals
GuitarMasterChef.sol:965:40: Error: Use double quotes for string
literals
GuitarMasterChef.sol:967:61: Error: Use double quotes for string
literals
GuitarMasterChef.sol:990:38: Error: Use double quotes for string
literals
GuitarMasterChef.sol:991:40: Error: Use double quotes for string
literals
```

```
GuitarMasterChef.sol:1008:60: Error: Use double quotes for string
literals
GuitarMasterChef.sol:1013:1: Error: Compiler version >0.6.6 does not
satisfy the r semver requirement
GuitarMasterChef.sol:1016:31: Error: Use double quotes for string
literals
GuitarMasterChef.sol:1016:51: Error: Use double quotes for string
literals
GuitarMasterChef.sol:1018:29: Error: Constant name must be in
capitalized SNAKE_CASE
GuitarMasterChef.sol:1135:17: Error: Avoid to make time-based decisions
in your business logic
GuitarMasterChef.sol:1257:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
GuitarMasterChef.sol:1264:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
GuitarMasterChef.sol:1268:34: Error: Use double quotes for string
literals
GuitarMasterChef.sol:1268:52: Error: Use double quotes for string
literals
GuitarMasterChef.sol:1403:17: Error: Avoid to make time-based decisions
in your business logic
GuitarMasterChef.sol:1525:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
GuitarMasterChef.sol:1532:1: Error: Compiler version 0.6.12 does not
satisfy the r semver requirement
GuitarMasterChef.sol:1591:20: Error: Variable name must be in mixedCase
```

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