AUDITED BY
LOVELY INSPECTOR

SMART CONTRACT

SECURITY AUDIT

PANCAKE SWAP

inspector.lovely.finance

LOVELY INSPECTOR

TABLE OF CONTENTS

LOVELY INSPECTOR

DISCLAIMER

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

LOVELY INSPECTOR

AUDIT SCOPE

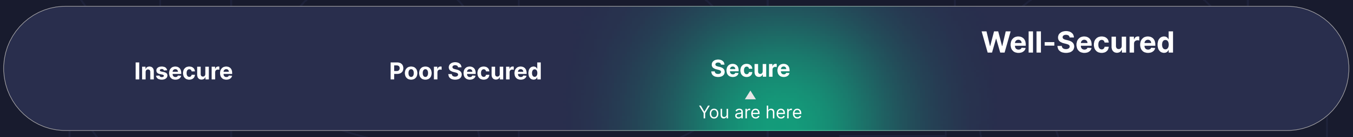
Name	Code Review and Security Analysis Report for PancakeSwap Contract
Platform	Ethereum / Solidity
File 1	MasterChef.sol
File 2	SyrupBar.sol
File 3	SousChef.sol
Github Commit Hash	a61313bf107c7f82e1a0f5736d815041fbf8cdff
Audit Date	October 27th, 2023

PROPOSED SMART CONTRACT FEATURES

Claimed Feature Detail	Our Observation
File 1 MasterChef.sol Owner Specifications: • Multiplier numbers can be set by the owner. • Add a new lp to the pool by the owner. • Update the given pool's CAKE allocation point by the owner • Set the migrator contract by the owner. • Update dev address by the previous dev.	Validated
File 2 SyrupBar.sol • Name: SyrupBar Token • Symbol: SYRUP Owner Specifications: • Mint a new token by the owner. • burn a token by the owner. • Safe cake transfer by the owner	Validated
File 3 SousChef.sol • SousChef is the chef of new tokens.	Validated

AUDIT SUMMARY

According to the standard audit assessment, the Customer`s solidity-based smart contracts are **“Secured”**. Also, these contracts contain owner control, which does not make them fully decentralized.



We used various tools like Slither, Solhin,t, and Remix IDE. At the same time, this finding is based on a 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 0 low, and 0 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.

LOVELY INSPECTOR

KEY TECHNICAL METRICS

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

Overall Audit Result: PASSED

LOVELY INSPECTOR

BUSINESS RISK ANALYSIS

CATEGORY	RESULT
● Buy Tax	0%
● Sell Tax	0%
● Cannot Buy	Not Detected
● Cannot Sell	Not Detected
● Max Tax	0%
● Modify Tax	Not Detected
● Fee Check	No
● Is Honeypot	Not Detected
● Trading Cooldown	Not Detected
● Can Pause Trade?	No
● Pause Transfer?	No
● Max Tax?	No
● Is it Anti-whale?	No
● Is Anti-bot?	Not Detected
● Is it a Blacklist?	Not Detected
● Blacklist Check	No
● Can Mint?	No
● Is it Proxy?	No
● Can Take Ownership?	Yes
● Hidden Owner?	Not Detected
● Self Destruction?	Not Detected
● Auditor Confidence	High

Overall Audit Result: PASSED

CODE QUALITY

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

The libraries in PancakeSwap 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 PancakeSwap Protocol.

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

Code parts are well commented on smart contracts..

DOCUMENTATION

We were given a PancakeSwap smart contract code in the form of a github web link. The hash of that code is mentioned above in the table.

As mentioned above, code parts are well commented on. And the logic is straightforward. 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://pancakeswap.finance which provided rich information about the project architecture and tokenomics.

USE OF DEPENDENCIES

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

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

LEVEL OF CRITICALITY

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

LOVELY INSPECTOR

AUDIT FINDINGS TABLE

	Total	Resolved	UnResolved	Acknowledged
High Severity Issues Found	0	0	0	0
Moderate Severity Issues Found	0	0	0	0
Medium Severity Issues	0	0	0	0
Low Severity Issues	0	0	0	0
Informational Observations	0	0	0	0

The PancakeSwap - Audit report identifies 0 issues with varying severity levels, discovered through manual review and static analysis techniques, alongside rigorous code reviews, highlighting the need for further investigation and vulnerability identification.

The smart contract is considered to **pass the audit,** as of the audit date, if no high severity or moderate severity issues are found.

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	No Low severity vulnerabilities were found.
Very Low / Informational / Best practices:	No Very Low severity vulnerabilities were found.

CENTRALIZATION

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

MasterChef.sol

- updateMultiplier: Multiplier number can be set by the owner.
- add: Add a new lp to the pool by the owner.
- set: Update the given pool's CAKE allocation point by the owner.
- setMigrator: Set the migrator contract by the owner.
- dev: Update dev address by the previous dev.

SyrupBar.sol

- mint: Mint a new token by the owner.
- burn: burn a token by the owner.
- safeCakeTransfer: Safe cake transfer by the owner.

BEP20.sol

mint: Mint a new token by the owner.

Ownable.sol

- **renounce Ownership: Deleting ownership will leave the contract without an owner, removing any owner-only functionality.**
- **transferOwnership: The current owner can transfer ownership of the contract to a new account.**

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

CONCLUSION

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

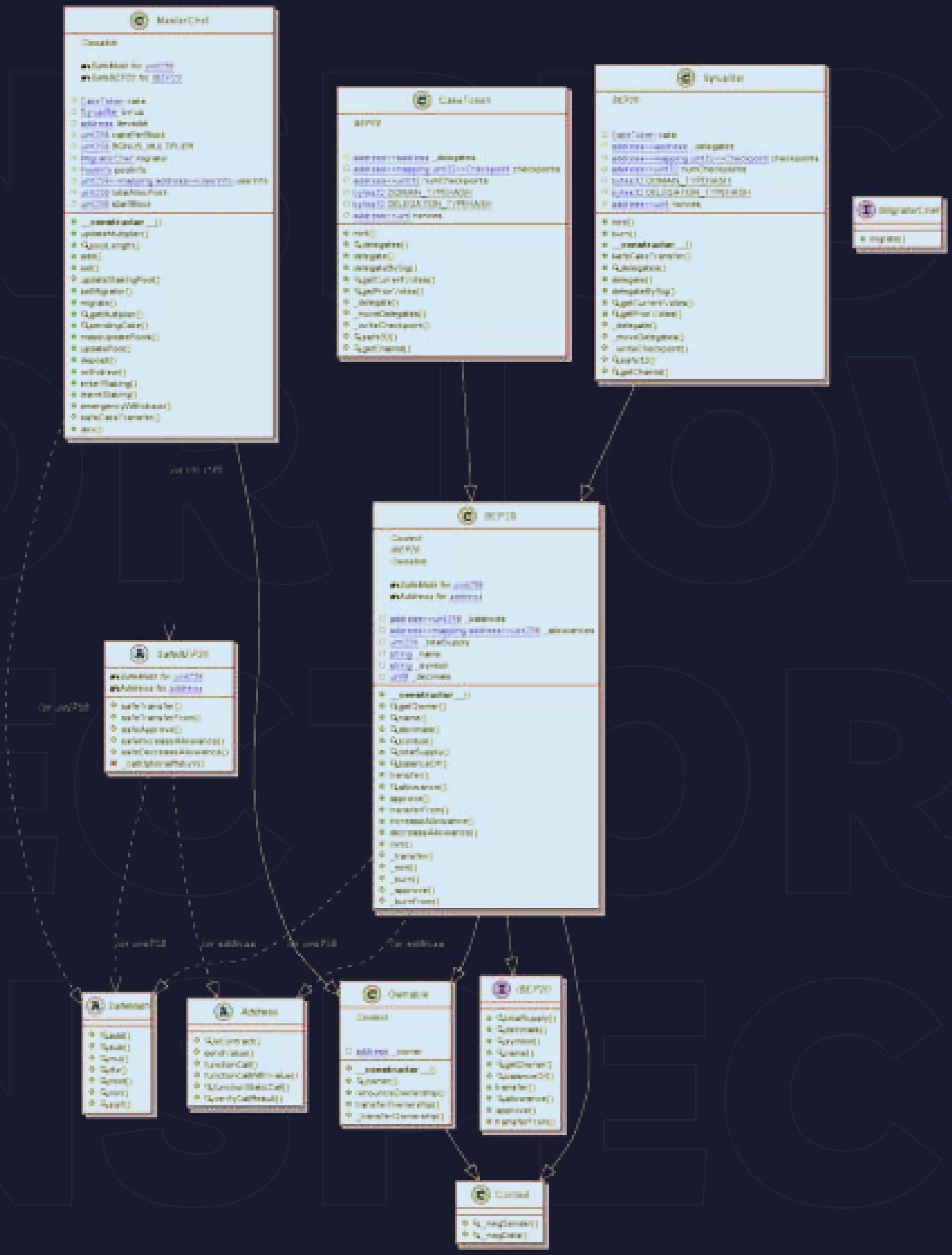
Security state of the reviewed smart contract, based on standard audit procedure scope, is **“Secured”.**

LOVELY INSPECTOR

ADDENDUM

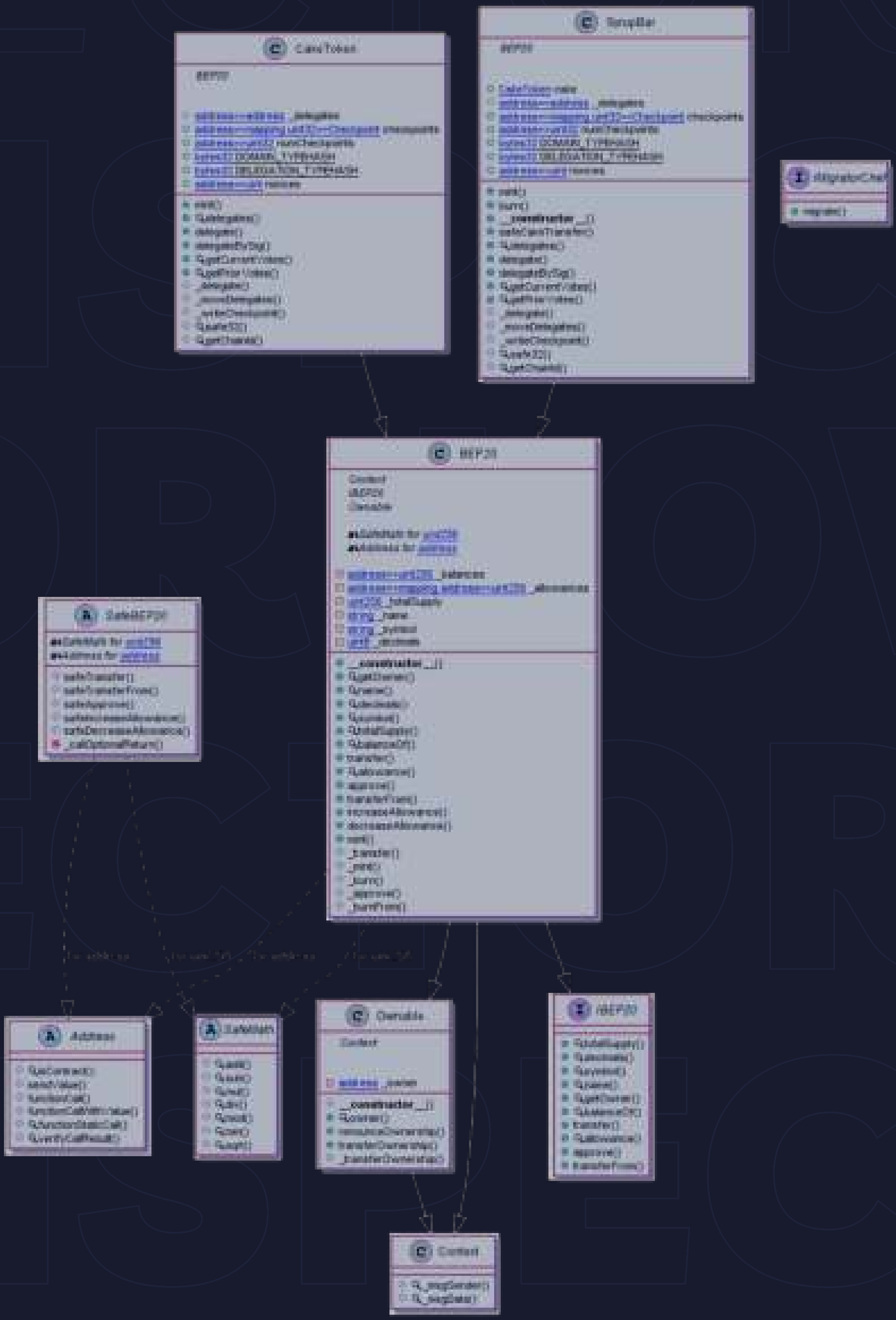
Code Flow Diagram - PancakeSwap

MasterChef Diagram



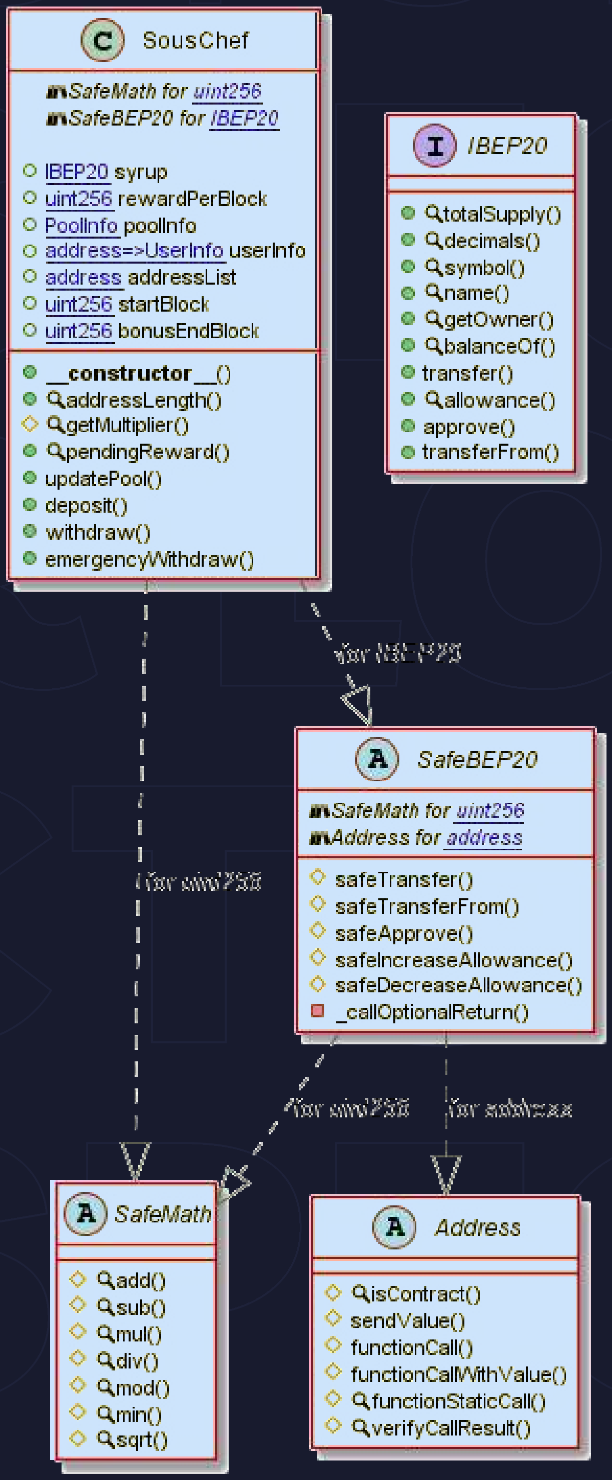
LOVELY INSPECTOR

SyrupBar Diagram



LOVELY INSPECTOR

SousChef Diagram

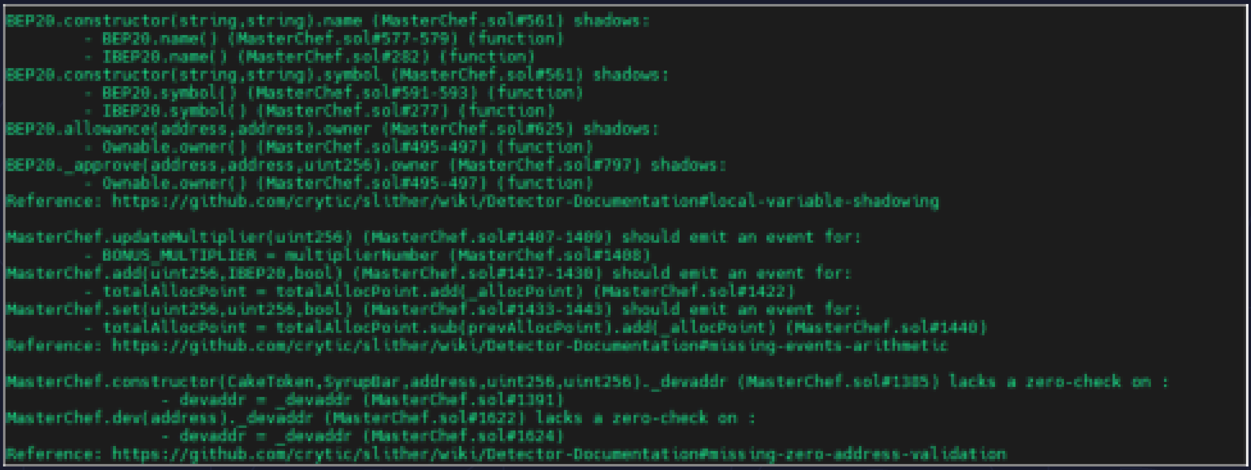


SECURITY ASSESSMENT REPORT

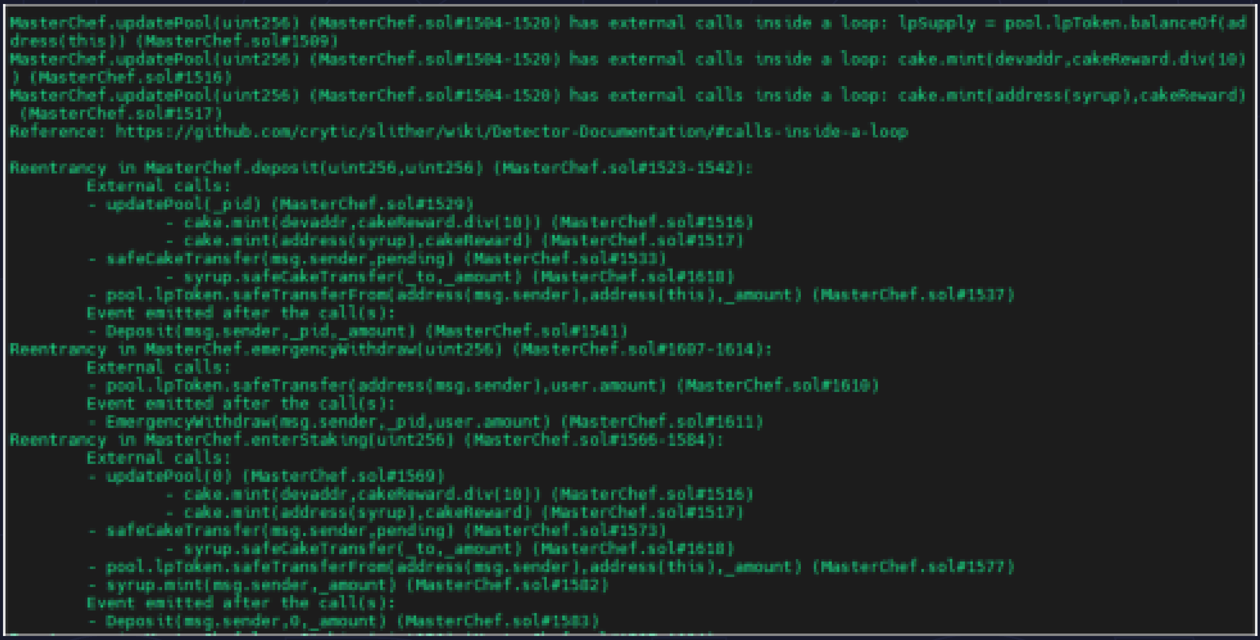
Slither is a Solidity static analysis framework that uses vulnerability detectors, displays contract details and provides an API for writing custom analyses. It helps developers identify vulnerabilities, improve code comprehension, and prototype custom analyses quickly. The analysis includes a report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

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

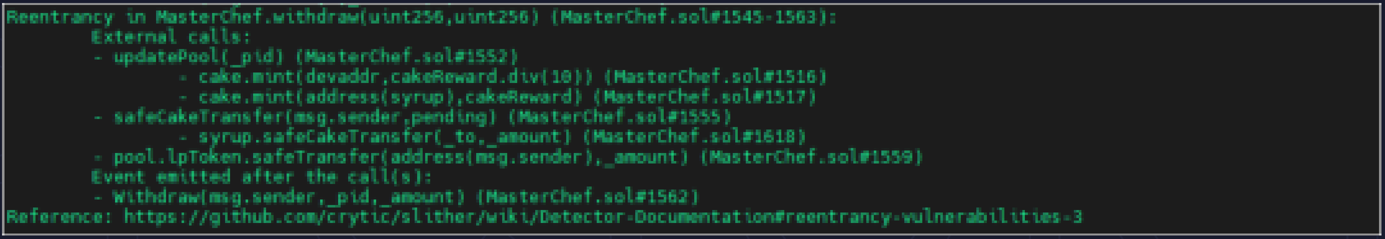
Slither log >> MasterChef.sol



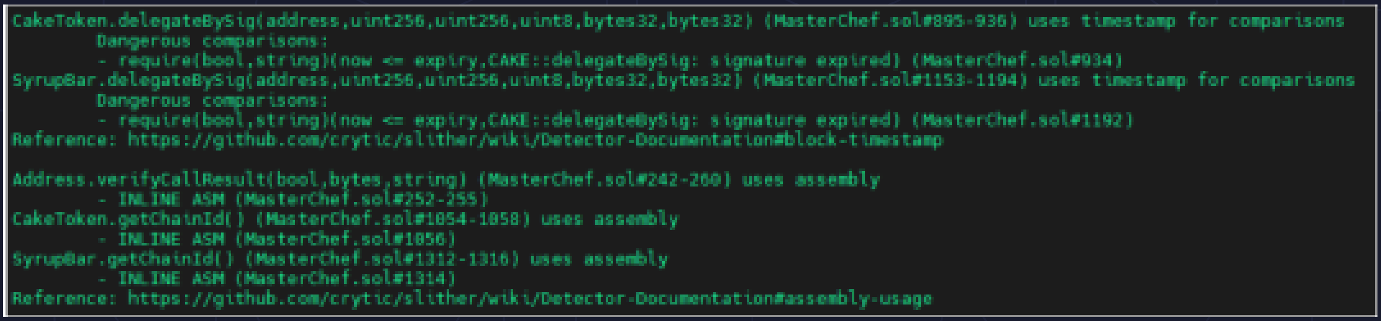
```
BEP20.constructor(string,string).name (MasterChef.sol#561) shadows:
        - BEP20.name() (MasterChef.sol#577-579) (function)
        - IBEP20.name() (MasterChef.sol#282) (function)
BEP20.constructor(string,string).symbol (MasterChef.sol#561) shadows:
        - BEP20.symbol() (MasterChef.sol#591-593) (function)
        - IBEP20.symbol() (MasterChef.sol#277) (function)
BEP20.allowance(address,address).owner (MasterChef.sol#625) shadows:
        - Ownable.owner() (MasterChef.sol#495-497) (function)
BEP20._approve(address,address,uint256).owner (MasterChef.sol#797) shadows:
        - Ownable.owner() (MasterChef.sol#495-497) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

MasterChef.updateMultiplier(uint256) (MasterChef.sol#1407-1409) should emit an event for:
        - BONUS_MULTIPLIER = multiplierNumber (MasterChef.sol#1408)
MasterChef.add(uint256,IBEP20,bool) (MasterChef.sol#1417-1430) should emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (MasterChef.sol#1422)
MasterChef.set(uint256,uint256,bool) (MasterChef.sol#1433-1443) should emit an event for:
        - totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint) (MasterChef.sol#1440)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

MasterChef.constructor(CakeToken,SyrupBar,address,uint256,uint256)._devaddr (MasterChef.sol#1385) lacks a zero-check on :
                - devaddr = _devaddr (MasterChef.sol#1391)
MasterChef.dev(address)._devaddr (MasterChef.sol#1622) lacks a zero-check on :
                - devaddr = _devaddr (MasterChef.sol#1624)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```



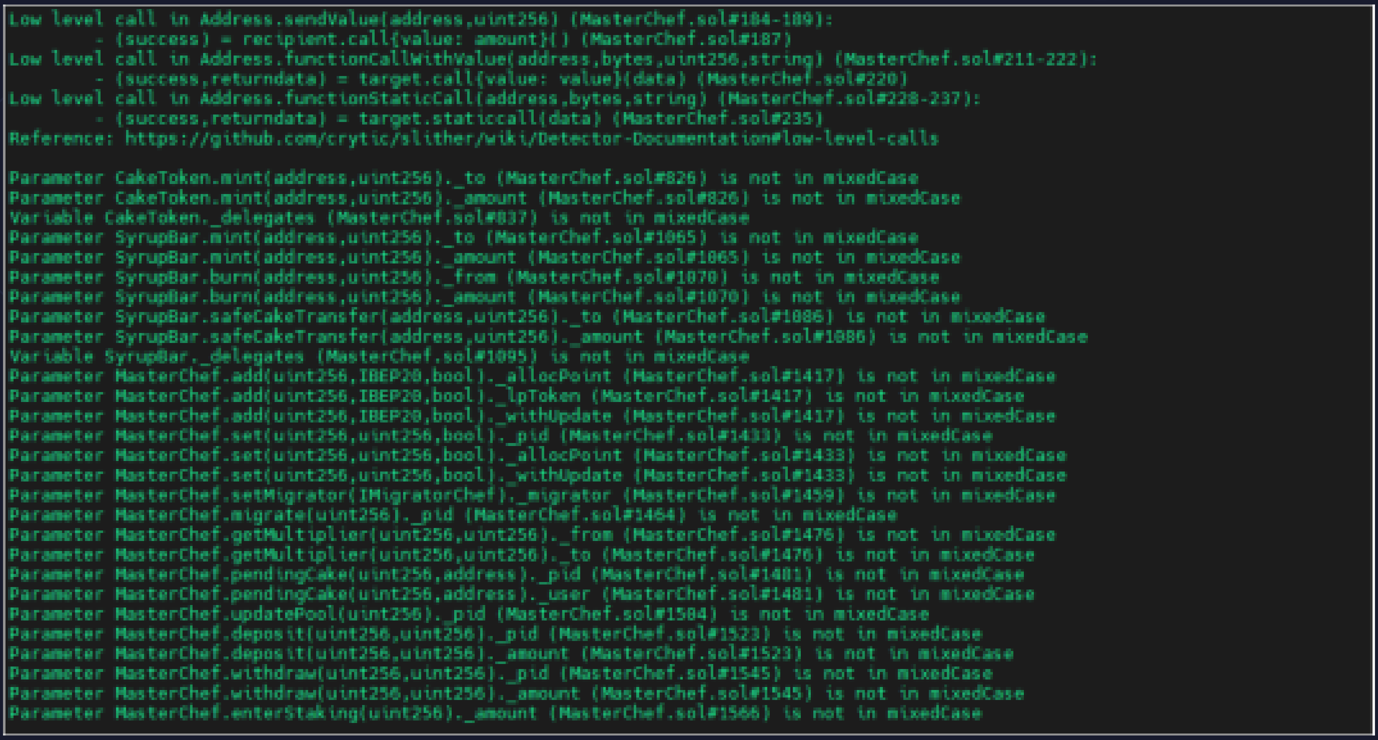
```
MasterChef.updatePool(uint256) (MasterChef.sol#1504-1520) has external calls inside a loop: lpSupply = pool.lpToken.balanceOf(ad
dress(this)) (MasterChef.sol#1509)
MasterChef.updatePool(uint256) (MasterChef.sol#1504-1520) has external calls inside a loop: cake.mint(devaddr,cakeReward.div(10)
) (MasterChef.sol#1516)
MasterChef.updatePool(uint256) (MasterChef.sol#1504-1520) has external calls inside a loop: cake.mint(address(syrup),cakeReward)
 (MasterChef.sol#1517)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in MasterChef.deposit(uint256,uint256) (MasterChef.sol#1523-1542):
        External calls:
        - updatePool(_pid) (MasterChef.sol#1529)
                - cake.mint(devaddr,cakeReward.div(10)) (MasterChef.sol#1516)
                - cake.mint(address(syrup),cakeReward) (MasterChef.sol#1517)
        - safeCakeTransfer(msg.sender,pending) (MasterChef.sol#1533)
                - syrup.safeCakeTransfer(_to,_amount) (MasterChef.sol#1618)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterChef.sol#1537)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (MasterChef.sol#1541)
Reentrancy in MasterChef.emergencyWithdraw(uint256) (MasterChef.sol#1607-1614):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (MasterChef.sol#1610)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,user.amount) (MasterChef.sol#1611)
Reentrancy in MasterChef.enterStaking(uint256) (MasterChef.sol#1566-1584):
        External calls:
        - updatePool(0) (MasterChef.sol#1569)
                - cake.mint(devaddr,cakeReward.div(10)) (MasterChef.sol#1516)
                - cake.mint(address(syrup),cakeReward) (MasterChef.sol#1517)
        - safeCakeTransfer(msg.sender,pending) (MasterChef.sol#1573)
                - syrup.safeCakeTransfer(_to,_amount) (MasterChef.sol#1618)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterChef.sol#1577)
        - syrup.mint(msg.sender,_amount) (MasterChef.sol#1582)
        Event emitted after the call(s):
        - Deposit(msg.sender,0,_amount) (MasterChef.sol#1583)
```



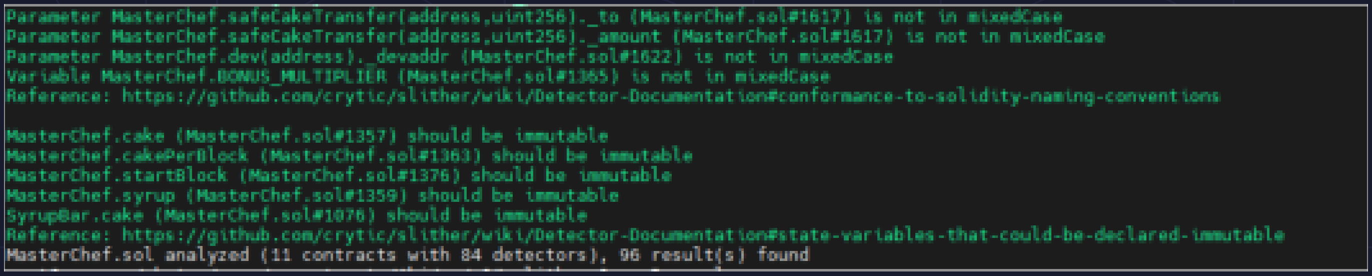
```
Reentrancy in MasterChef.withdraw(uint256,uint256) (MasterChef.sol#1545-1563):
        External calls:
        - updatePool(_pid) (MasterChef.sol#1552)
                - cake.mint(devaddr,cakeReward.div(10)) (MasterChef.sol#1516)
                - cake.mint(address(syrup),cakeReward) (MasterChef.sol#1517)
        - safeCakeTransfer(msg.sender,pending) (MasterChef.sol#1555)
                - syrup.safeCakeTransfer(_to,_amount) (MasterChef.sol#1618)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterChef.sol#1559)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (MasterChef.sol#1562)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```



```
CakeToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (MasterChef.sol#895-936) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,CAKE::delegateBySig: signature expired) (MasterChef.sol#934)
SyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (MasterChef.sol#1153-1194) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,CAKE::delegateBySig: signature expired) (MasterChef.sol#1192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.verifyCallResult(bool,bytes,string) (MasterChef.sol#242-260) uses assembly
        - INLINE ASM (MasterChef.sol#252-255)
CakeToken.getChainId() (MasterChef.sol#1054-1058) uses assembly
        - INLINE ASM (MasterChef.sol#1056)
SyrupBar.getChainId() (MasterChef.sol#1312-1316) uses assembly
        - INLINE ASM (MasterChef.sol#1314)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

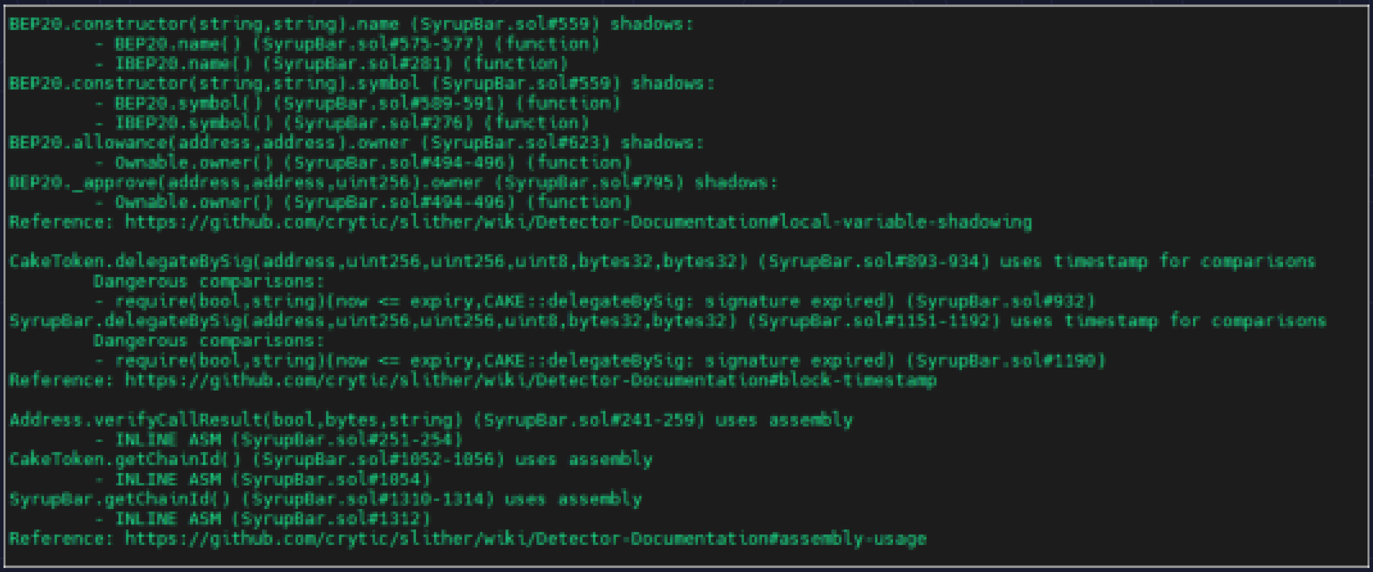


```
Low level call in Address.sendValue(address,uint256) (MasterChef.sol#184-189):
        - (success) = recipient.call{value: amount}() (MasterChef.sol#187)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MasterChef.sol#211-222):
        - (success,returndata) = target.call{value: value}(data) (MasterChef.sol#220)
Low level call in Address.functionStaticCall(address,bytes,string) (MasterChef.sol#228-237):
        - (success,returndata) = target.staticcall(data) (MasterChef.sol#235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter CakeToken.mint(address,uint256)._to (MasterChef.sol#826) is not in mixedCase
Parameter CakeToken.mint(address,uint256)._amount (MasterChef.sol#826) is not in mixedCase
Variable CakeToken._delegates (MasterChef.sol#837) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._to (MasterChef.sol#1065) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._amount (MasterChef.sol#1065) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._from (MasterChef.sol#1070) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._amount (MasterChef.sol#1070) is not in mixedCase
Parameter SyrupBar.safeCakeTransfer(address,uint256)._to (MasterChef.sol#1086) is not in mixedCase
Parameter SyrupBar.safeCakeTransfer(address,uint256)._amount (MasterChef.sol#1086) is not in mixedCase
Variable SyrupBar._delegates (MasterChef.sol#1095) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,bool)._allocPoint (MasterChef.sol#1417) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,bool)._lpToken (MasterChef.sol#1417) is not in mixedCase
Parameter MasterChef.add(uint256,IBEP20,bool)._withUpdate (MasterChef.sol#1417) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._pid (MasterChef.sol#1433) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._allocPoint (MasterChef.sol#1433) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,bool)._withUpdate (MasterChef.sol#1433) is not in mixedCase
Parameter MasterChef.setMigrator(IMigratorChef)._migrator (MasterChef.sol#1459) is not in mixedCase
Parameter MasterChef.migrate(uint256)._pid (MasterChef.sol#1464) is not in mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._from (MasterChef.sol#1476) is not in mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._to (MasterChef.sol#1476) is not in mixedCase
Parameter MasterChef.pendingCake(uint256,address)._pid (MasterChef.sol#1481) is not in mixedCase
Parameter MasterChef.pendingCake(uint256,address)._user (MasterChef.sol#1481) is not in mixedCase
Parameter MasterChef.updatePool(uint256)._pid (MasterChef.sol#1504) is not in mixedCase
Parameter MasterChef.deposit(uint256,uint256)._pid (MasterChef.sol#1523) is not in mixedCase
Parameter MasterChef.deposit(uint256,uint256)._amount (MasterChef.sol#1523) is not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._pid (MasterChef.sol#1545) is not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._amount (MasterChef.sol#1545) is not in mixedCase
Parameter MasterChef.enterStaking(uint256)._amount (MasterChef.sol#1566) is not in mixedCase
```

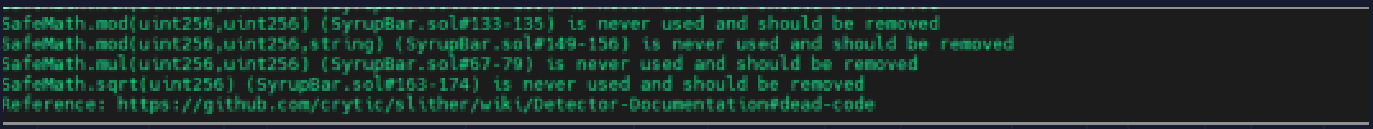


```
Parameter MasterChef.safeCakeTransfer(address,uint256)._to (MasterChef.sol#1617) is not in mixedCase
Parameter MasterChef.safeCakeTransfer(address,uint256)._amount (MasterChef.sol#1617) is not in mixedCase
Parameter MasterChef.dev(address)._devaddr (MasterChef.sol#1622) is not in mixedCase
Variable MasterChef.BONUS_MULTIPLIER (MasterChef.sol#1365) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

MasterChef.cake (MasterChef.sol#1357) should be immutable
MasterChef.cakePerBlock (MasterChef.sol#1363) should be immutable
MasterChef.startBlock (MasterChef.sol#1376) should be immutable
MasterChef.syrup (MasterChef.sol#1359) should be immutable
SyrupBar.cake (MasterChef.sol#1076) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
MasterChef.sol analyzed (11 contracts with 84 detectors), 96 result(s) found
```

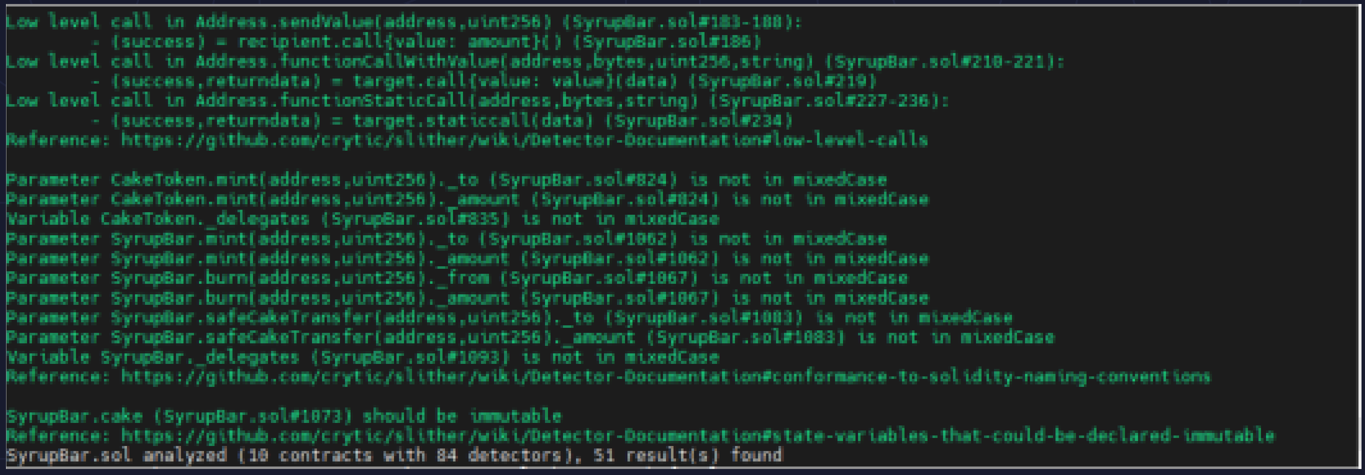
Slither log >> SyrupBar.sol



```
BEP20.constructor(string,string).name (SyrupBar.sol#559) shadows:
        - BEP20.name() (SyrupBar.sol#575-577) (function)
        - IBEP20.name() (SyrupBar.sol#281) (function)
BEP20.constructor(string,string).symbol (SyrupBar.sol#559) shadows:
        - BEP20.symbol() (SyrupBar.sol#589-591) (function)
        - IBEP20.symbol() (SyrupBar.sol#276) (function)
BEP20.allowance(address,address).owner (SyrupBar.sol#623) shadows:
        - Ownable.owner() (SyrupBar.sol#494-496) (function)
BEP20._approve(address,address,uint256).owner (SyrupBar.sol#795) shadows:
        - Ownable.owner() (SyrupBar.sol#494-496) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

CakeToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#893-934) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,CAKE::delegateBySig: signature expired) (SyrupBar.sol#932)
SyrupBar.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (SyrupBar.sol#1151-1192) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,CAKE::delegateBySig: signature expired) (SyrupBar.sol#1190)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.verifyCallResult(bool,bytes,string) (SyrupBar.sol#241-259) uses assembly
        - INLINE ASM (SyrupBar.sol#251-254)
CakeToken.getChainId() (SyrupBar.sol#1052-1056) uses assembly
        - INLINE ASM (SyrupBar.sol#1054)
SyrupBar.getChainId() (SyrupBar.sol#1310-1314) uses assembly
        - INLINE ASM (SyrupBar.sol#1312)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```



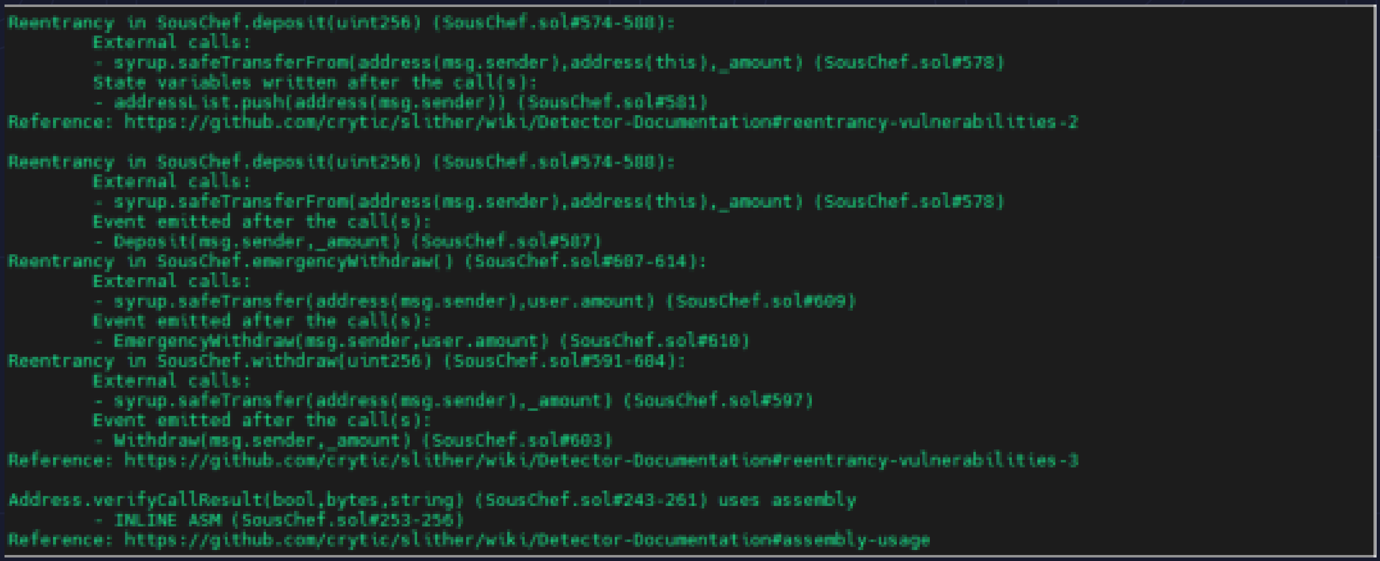
```
SafeMath.mod(uint256,uint256) (SyrupBar.sol#133-135) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SyrupBar.sol#149-156) is never used and should be removed
SafeMath.mul(uint256,uint256) (SyrupBar.sol#67-79) is never used and should be removed
SafeMath.sqrt(uint256) (SyrupBar.sol#163-174) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```



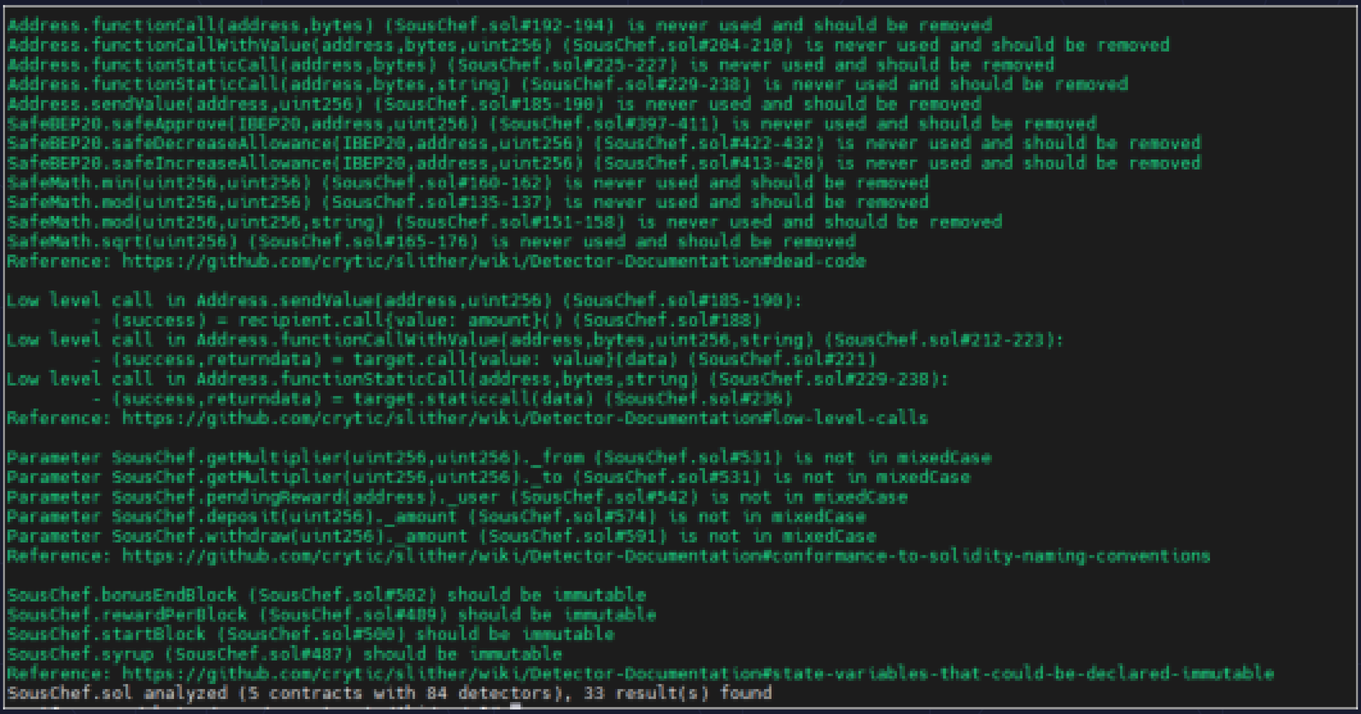
```
Low level call in Address.sendValue(address,uint256) (SyrupBar.sol#183-188):
        - (success) = recipient.call{value: amount}() (SyrupBar.sol#186)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (SyrupBar.sol#210-221):
        - (success,returndata) = target.call{value: value}(data) (SyrupBar.sol#219)
Low level call in Address.functionStaticCall(address,bytes,string) (SyrupBar.sol#227-236):
        - (success,returndata) = target.staticcall(data) (SyrupBar.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter CakeToken.mint(address,uint256)._to (SyrupBar.sol#824) is not in mixedCase
Parameter CakeToken.mint(address,uint256)._amount (SyrupBar.sol#824) is not in mixedCase
Variable CakeToken._delegates (SyrupBar.sol#835) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._to (SyrupBar.sol#1062) is not in mixedCase
Parameter SyrupBar.mint(address,uint256)._amount (SyrupBar.sol#1062) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._from (SyrupBar.sol#1067) is not in mixedCase
Parameter SyrupBar.burn(address,uint256)._amount (SyrupBar.sol#1067) is not in mixedCase
Parameter SyrupBar.safeCakeTransfer(address,uint256)._to (SyrupBar.sol#1083) is not in mixedCase
Parameter SyrupBar.safeCakeTransfer(address,uint256)._amount (SyrupBar.sol#1083) is not in mixedCase
Variable SyrupBar._delegates (SyrupBar.sol#1093) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SyrupBar.cake (SyrupBar.sol#1073) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SyrupBar.sol analyzed (10 contracts with 84 detectors), 51 result(s) found
```

LOVELY INSPECTOR

Slither log >> SousChef.sol



```
Reentrancy in SousChef.deposit(uint256) (SousChef.sol#574-588):
        External calls:
        - syrup.safeTransferFrom(address(msg.sender),address(this),_amount) (SousChef.sol#578)
        State variables written after the call(s):
        - addressList.push(address(msg.sender)) (SousChef.sol#581)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in SousChef.deposit(uint256) (SousChef.sol#574-588):
        External calls:
        - syrup.safeTransferFrom(address(msg.sender),address(this),_amount) (SousChef.sol#578)
        Event emitted after the call(s):
        - Deposit(msg.sender,_amount) (SousChef.sol#587)
Reentrancy in SousChef.emergencyWithdraw() (SousChef.sol#607-614):
        External calls:
        - syrup.safeTransfer(address(msg.sender),user.amount) (SousChef.sol#609)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,user.amount) (SousChef.sol#610)
Reentrancy in SousChef.withdraw(uint256) (SousChef.sol#591-604):
        External calls:
        - syrup.safeTransfer(address(msg.sender),_amount) (SousChef.sol#597)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_amount) (SousChef.sol#603)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.verifyCallResult(bool,bytes,string) (SousChef.sol#243-261) uses assembly
        - INLINE ASM (SousChef.sol#253-256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```



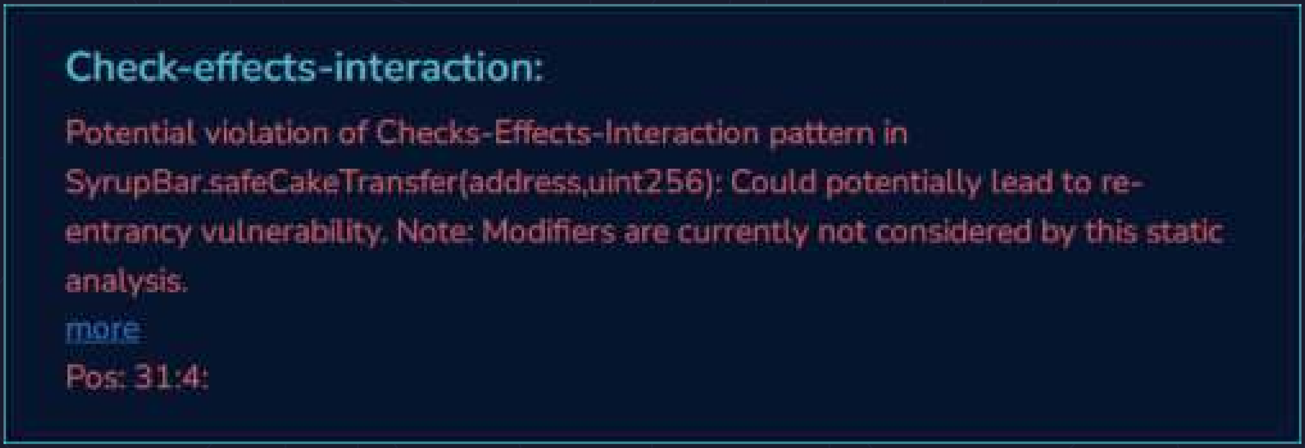
```
Address.functionCall(address,bytes) (SousChef.sol#192-194) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (SousChef.sol#204-210) is never used and should be removed
Address.functionStaticCall(address,bytes) (SousChef.sol#225-227) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (SousChef.sol#229-238) is never used and should be removed
Address.sendValue(address,uint256) (SousChef.sol#185-190) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (SousChef.sol#397-411) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (SousChef.sol#422-432) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (SousChef.sol#413-420) is never used and should be removed
SafeMath.min(uint256,uint256) (SousChef.sol#160-162) is never used and should be removed
SafeMath.mod(uint256,uint256) (SousChef.sol#135-137) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (SousChef.sol#151-158) is never used and should be removed
SafeMath.sqrt(uint256) (SousChef.sol#165-176) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Low level call in Address.sendValue(address,uint256) (SousChef.sol#185-190):
        - (success) = recipient.call{value: amount}() (SousChef.sol#188)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (SousChef.sol#212-223):
        - (success,returndata) = target.call{value: value}(data) (SousChef.sol#221)
Low level call in Address.functionStaticCall(address,bytes,string) (SousChef.sol#229-238):
        - (success,returndata) = target.staticcall(data) (SousChef.sol#236)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter SousChef.getMultiplier(uint256,uint256)._from (SousChef.sol#531) is not in mixedCase
Parameter SousChef.getMultiplier(uint256,uint256)._to (SousChef.sol#531) is not in mixedCase
Parameter SousChef.pendingReward(address)._user (SousChef.sol#542) is not in mixedCase
Parameter SousChef.deposit(uint256)._amount (SousChef.sol#574) is not in mixedCase
Parameter SousChef.withdraw(uint256)._amount (SousChef.sol#591) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

SousChef.bonusEndBlock (SousChef.sol#502) should be immutable
SousChef.rewardPerBlock (SousChef.sol#489) should be immutable
SousChef.startBlock (SousChef.sol#500) should be immutable
SousChef.syrup (SousChef.sol#487) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
SousChef.sol analyzed (5 contracts with 84 detectors), 33 result(s) found
```

LOVELY INSPECTOR

SOLIDITY STATIC ANALYSIS

Static code analysis is used to identify many common coding problems before a program is released. It involves examining the code manually or using tools to automate the process. Static code analysis tools can automatically scan the code without executing it.

MasterChef.sol

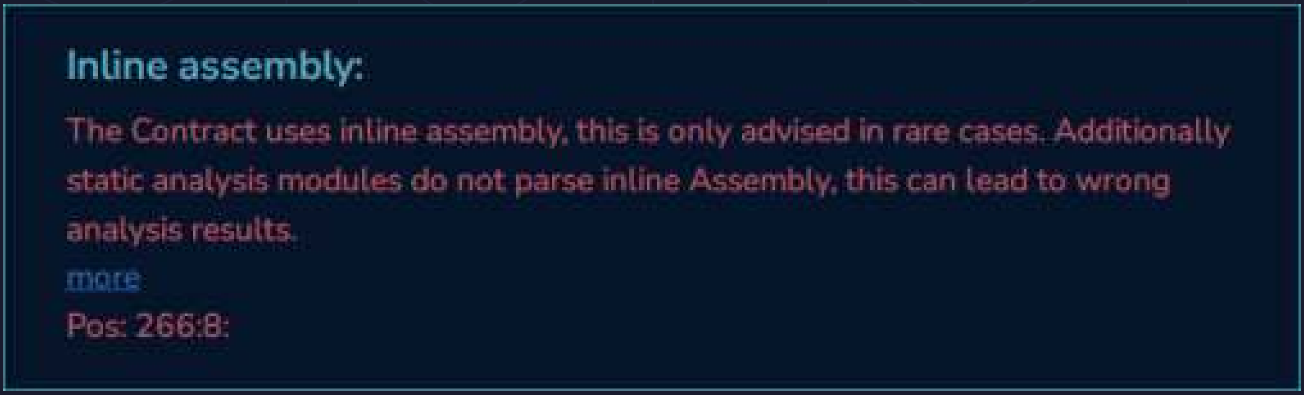


Check-effects-interaction:

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

more

Pos: 31:4:

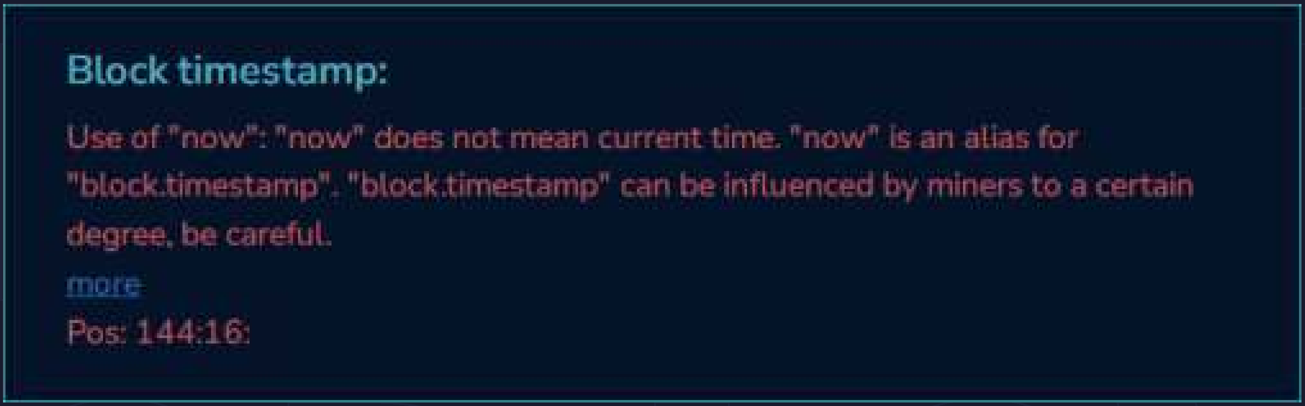


Inline assembly:

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: 266:8:

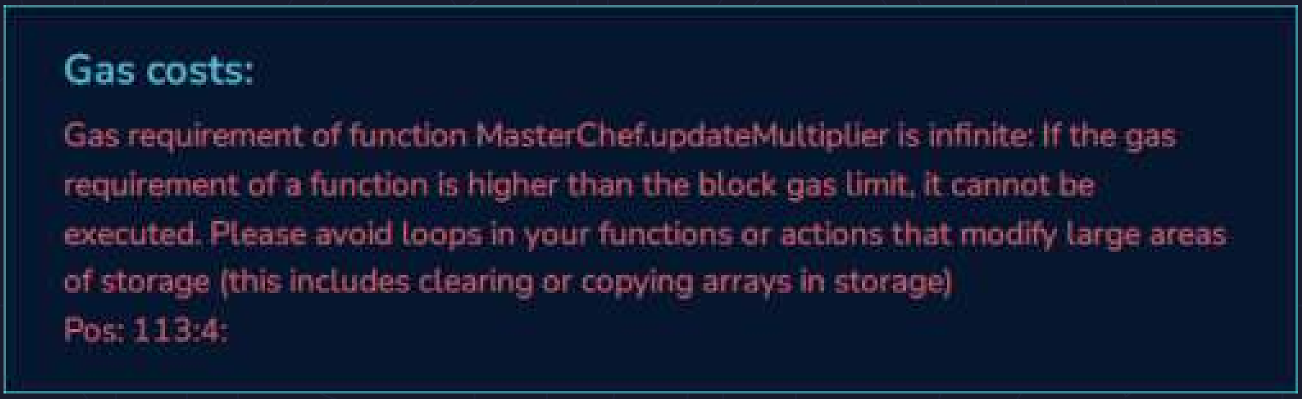


Block timestamp:

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.

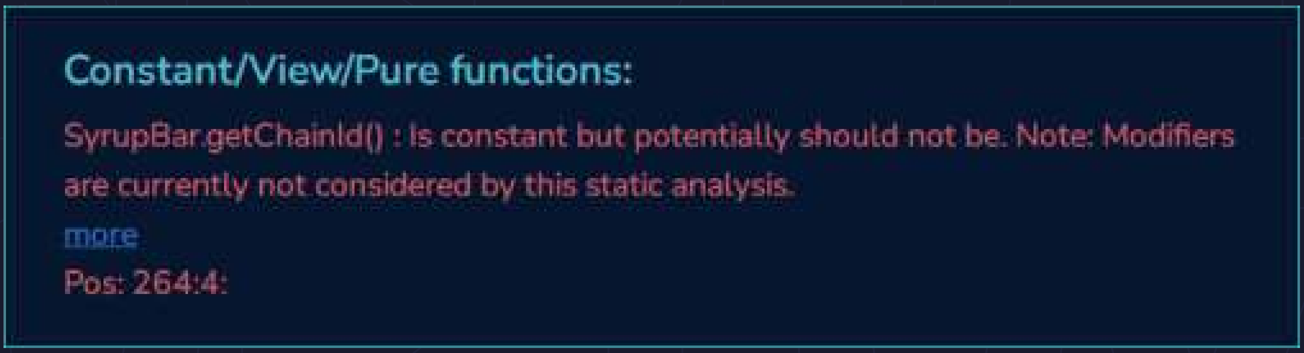
more

Pos: 144:16:



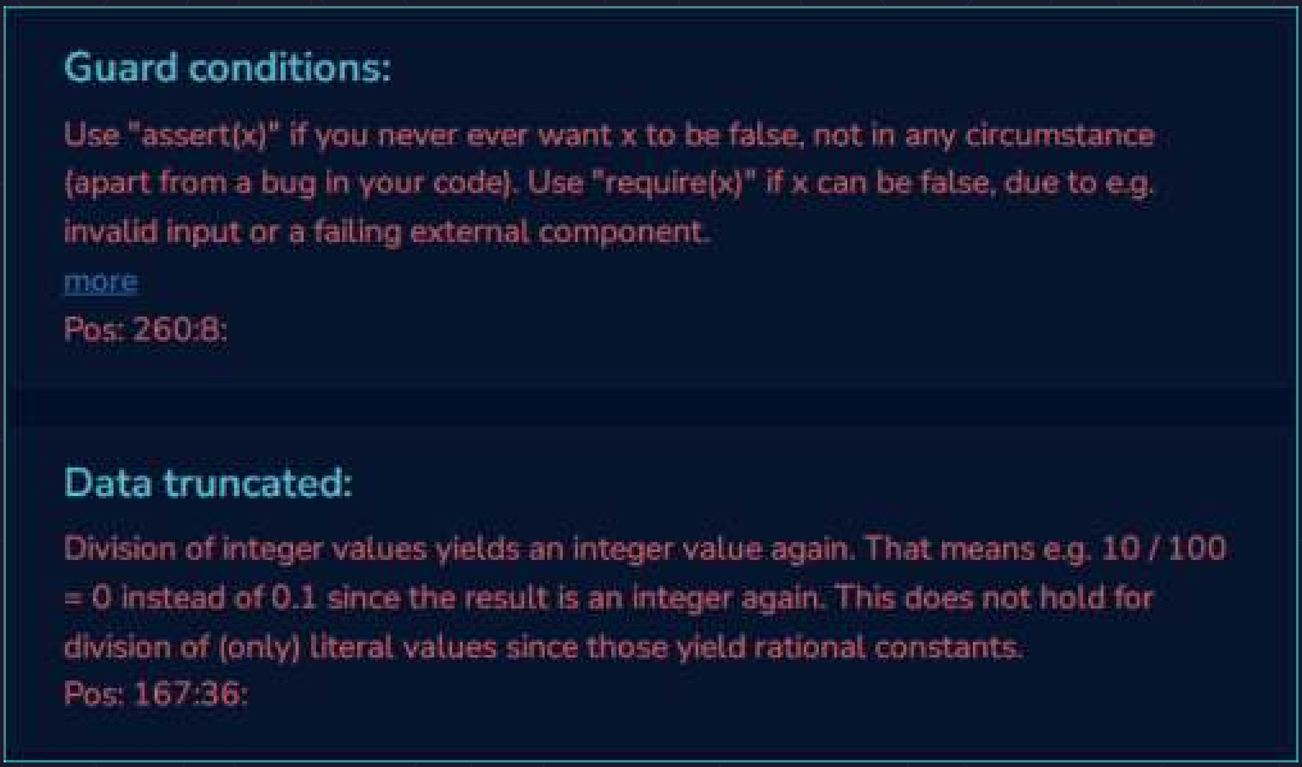
Gas costs:

Gas requirement of function MasterChef.updateMultiplier 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: 113:4:



Constant/View/Pure functions:

SyrupBar.getChainId() : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 264:4:



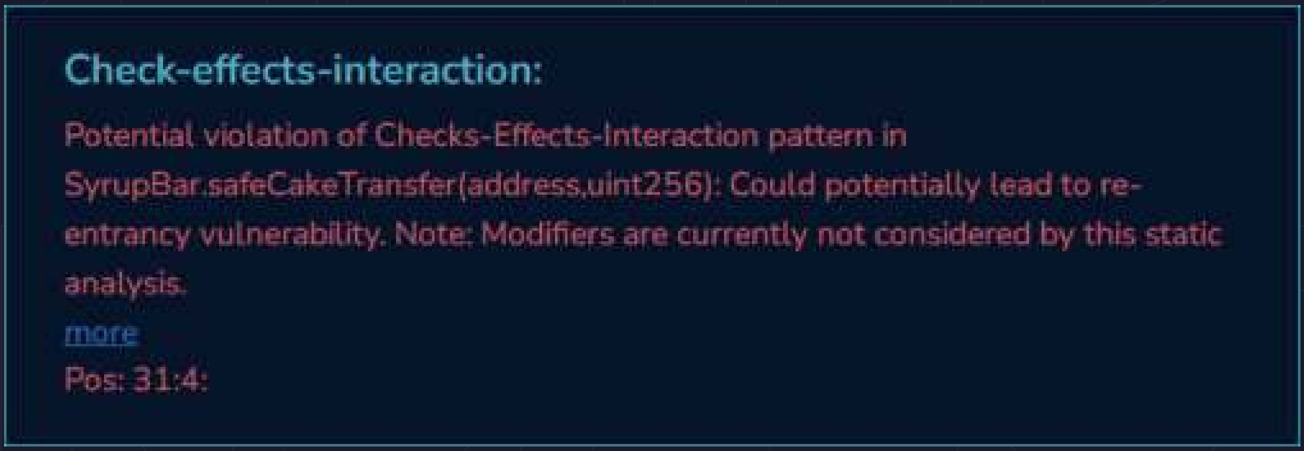
Guard conditions:

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: 260:8:

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: 167:36:

SyrupBar.sol

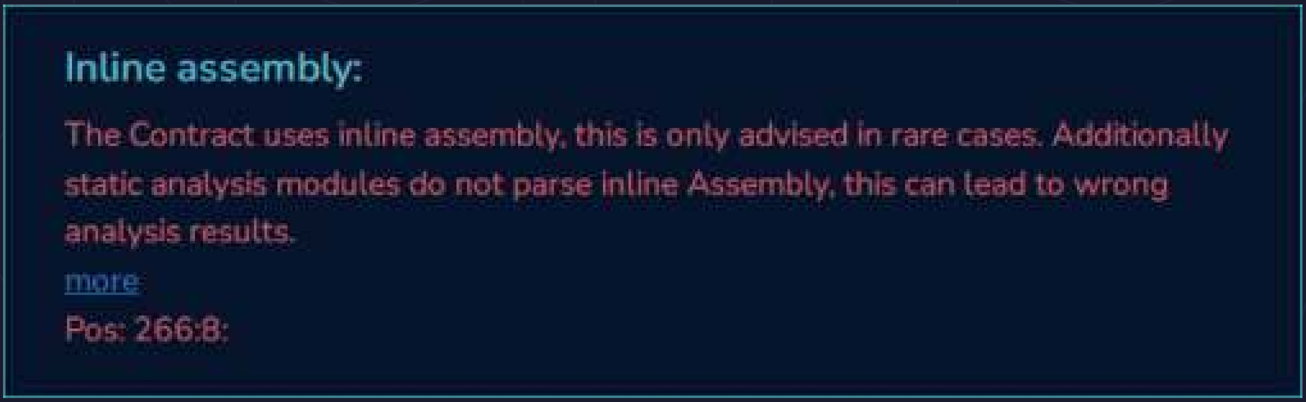


Check-effects-interaction:

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

more

Pos: 31:4:

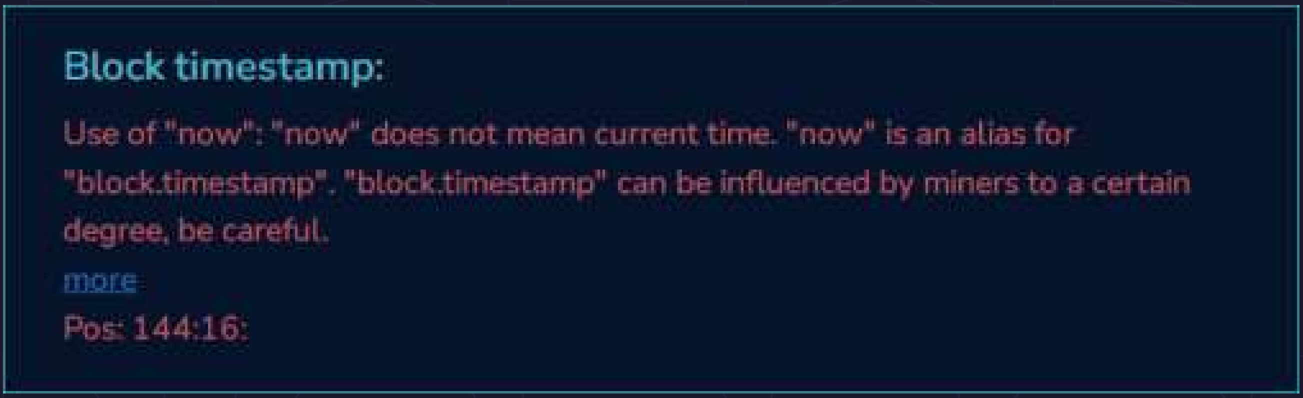


Inline assembly:

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: 266:8:

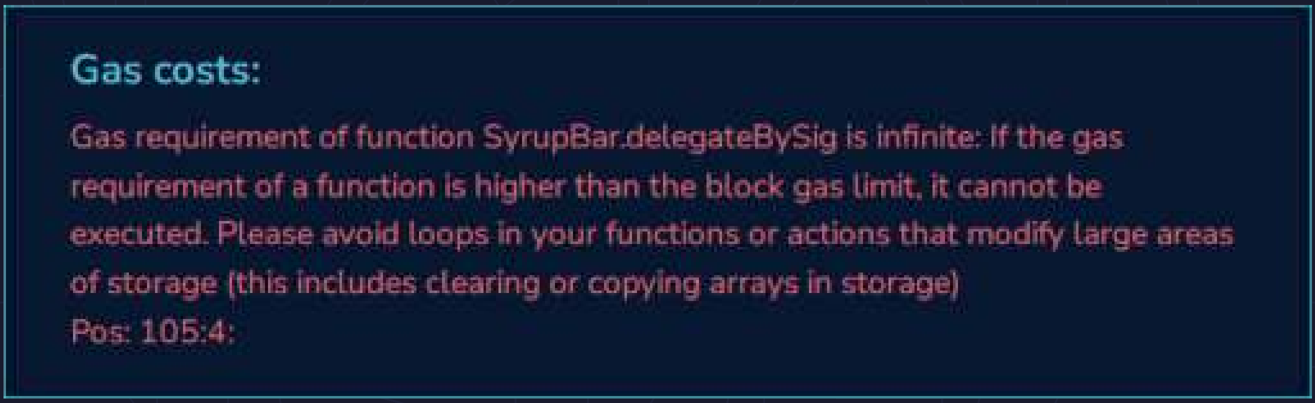


Block timestamp:

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.

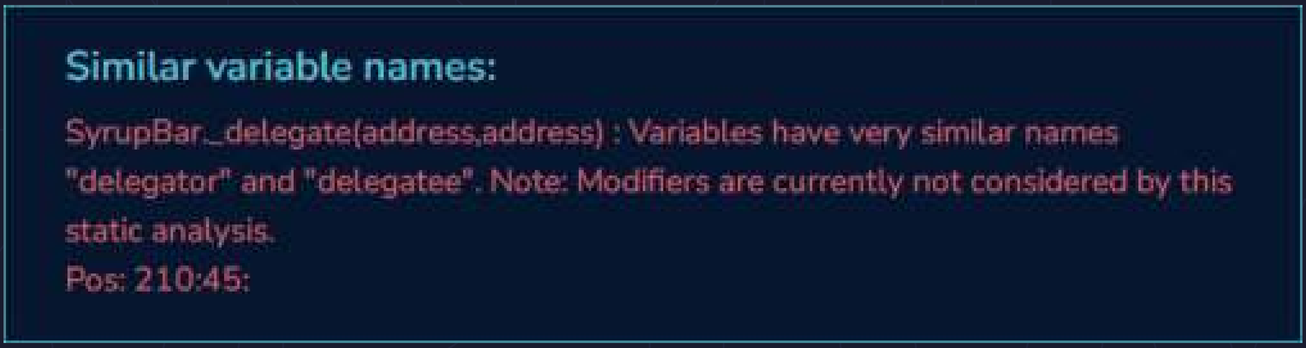
more

Pos: 144:16:



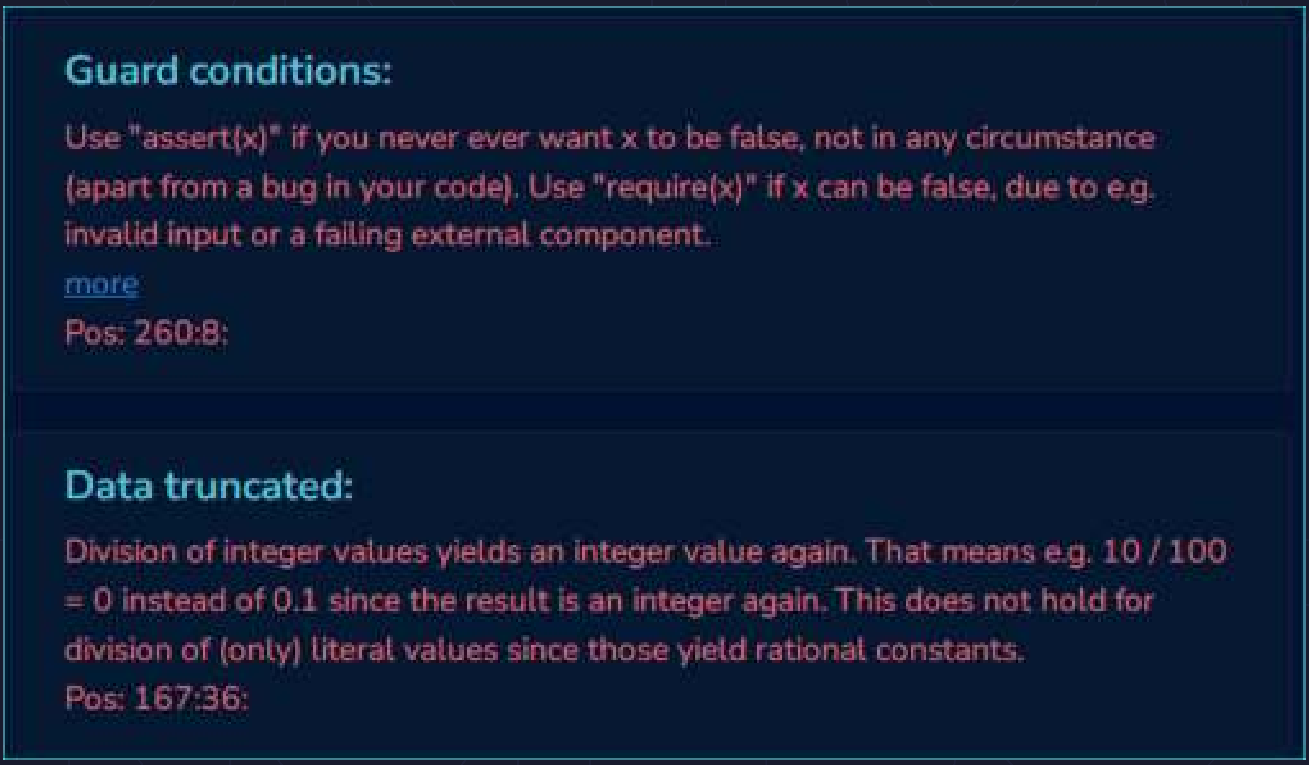
Gas costs:

Gas requirement of function SyrupBar.delegateBySig 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: 105:4:



Similar variable names:

SyrupBar._delegate(address,address) : Variables have very similar names "delegator" and "delegatee". Note: Modifiers are currently not considered by this static analysis.
Pos: 210:45:



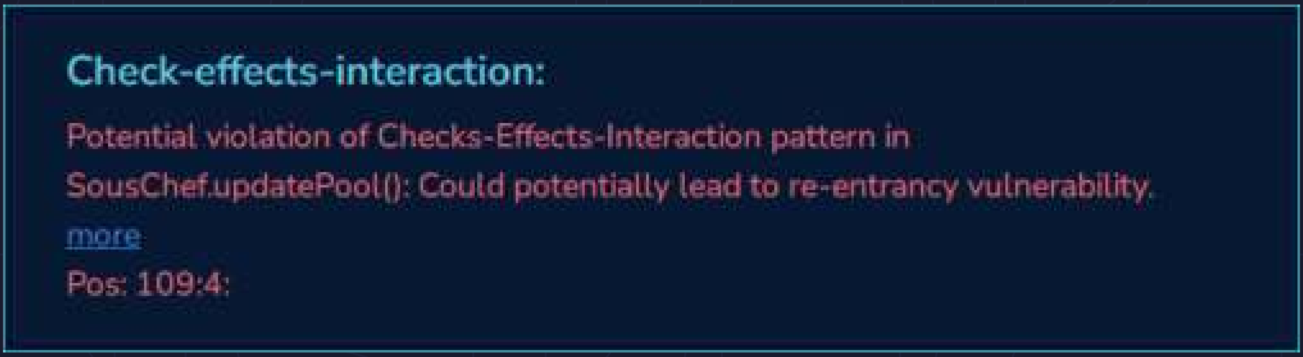
Guard conditions:

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: 260:8:

Data truncated:

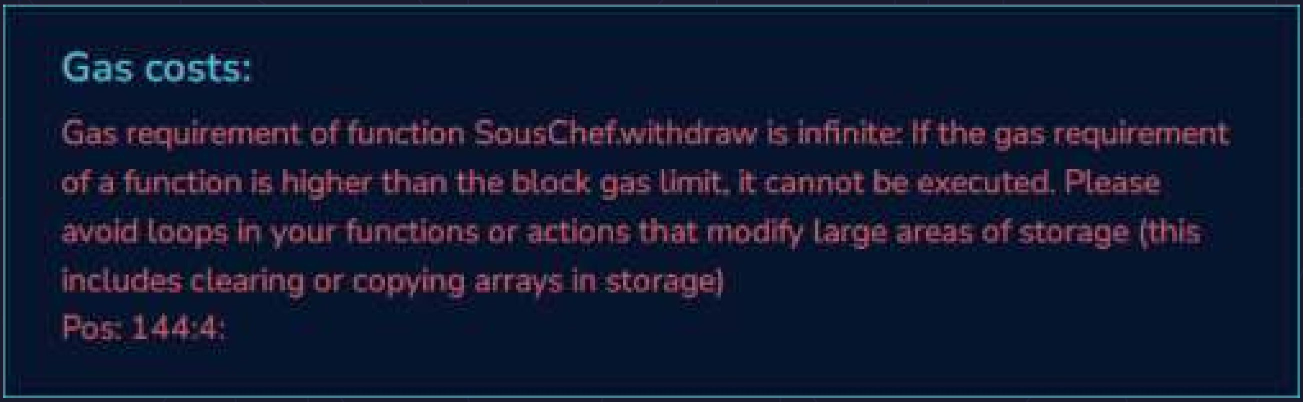
Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 167:36:

SousChef.sol



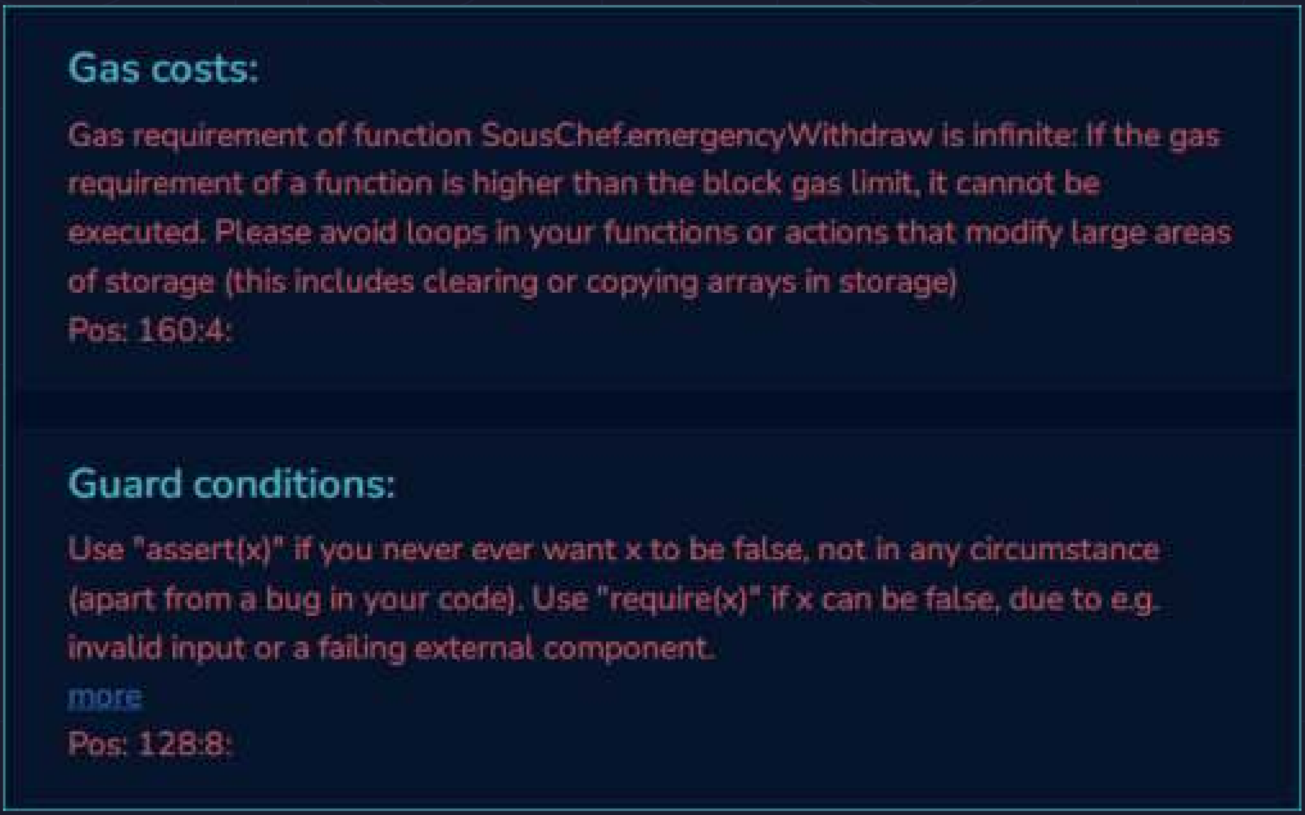
Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SousChef.updatePool(): Could potentially lead to re-entrancy vulnerability.
more
Pos: 109:4:



Gas costs:

Gas requirement of function SousChef.withdraw 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: 144:4:



Gas costs:

Gas requirement of function SousChef.emergencyWithdraw 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: 160:4:

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: 128:8:

LOVELY INSPECTOR

COMPLIANCE ANALYSIS

Linters are the utility tools that analyze the given source code and report programming errors, bugs, and stylistic errors. For the Solidity language, there are some linter tools available that a developer can use to improve the quality of their Solidity contracts.

MasterChef.sol

```
Compiler version 0.6.12 does not satisfy the ^0.5.8 semver requirement
Pos: 1:0
global import of path @pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol is not allowed. Specify names to import
individually or bind all exports of the module into a name (import "path" as Name)
Pos: 1:2
Use double quotes for string literals
Pos: 8:2
global import of path @pancakeswap/pancake-swap-lib/contracts/token/BEP20/IBEP20.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name (import "path" as Name)
Pos: 1:3
Use double quotes for string literals
Pos: 8:3
global import of path @pancakeswap/pancake-swap-lib/contracts/token/BEP20/SafeBEP20.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name (import "path" as Name)
Pos: 1:4
Use double quotes for string literals
Pos: 8:4
global import of path @pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol is not allowed. Specify names to import
individually or bind all exports of the module into a name (import "path" as Name)
Pos: 1:5
Use double quotes for string literals
Pos: 8:5
global import of path ./SyrupBar.sol is not allowed. Specify names to import individually or bind all exports of the module into a
name (import "path" as Name)
Pos: 1:8
Variable name must be in mixedCase
Pos: 5:70
Use double quotes for string literals
Pos: 29:230
Use double quotes for string literals
Pos: 29:252
```

SyrupBar.sol

```
Compiler version 0.6.12 does not satisfy the ^0.5.8 semver requirement
Pos: 1:0
global import of path @pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name (import "path" as Name)
Pos: 1:2
Error message for require is too long
Pos: 9:143
Avoid making time-based decisions in your business logic
Pos: 17:143
Avoid using inline assembly. It is acceptable only in rare cases
Pos: 9:265
```

SousChef.sol

```
Compiler version 0.6.12 does not satisfy the ^0.5.8 semver requirement
Pos: 1:0
global import of path @pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol is not allowed. Specify names to import
individually or bind all exports of the module into a name (import "path" as Name)
Pos: 1:2
Use double quotes for string literals
Pos: 8:2
global import of path @pancakeswap/pancake-swap-lib/contracts/token/BEP20/IBEP20.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name (import "path" as Name)
Pos: 1:3
Use double quotes for string literals
Pos: 8:3
global import of path @pancakeswap/pancake-swap-lib/contracts/token/BEP20/SafeBEP20.sol is not allowed. Specify names to
import individually or bind all exports of the module into a name (import "path" as Name)
Pos: 1:4
Use double quotes for string literals
Pos: 8:4
Use double quotes for string literals
Pos: 31:127
```

LOVELY INSPECTOR

SOFTWARE ANALYSIS RESULT

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

LOVELY INSPECTOR

INSPECTOR LOVELY

INFO

Website: Inspector.lovely.finance

Telegram community: t.me/inspectorlovely

Twitter: twitter.com/InspectorLovely

inspector.lovely.finance