SMART CONTRACT

Security Audit Report

Customer:	iBG Finance
Website:	https://ibg.finance
Platform:	Ethereum
Language:	Solidity
Date:	August 29th, 2021

Table of contents

Introduction

AntiHACK.me was contracted by the iBG Finance team to perform the Security audit of the iBG Farming and Staking Reserve 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 August 29th, 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

iBG is a Decentralized Finance (Defi) Wealth management platform designed to bring simplicity to users interested in entering the cryptocurrency and the Defi market.

Audit scope

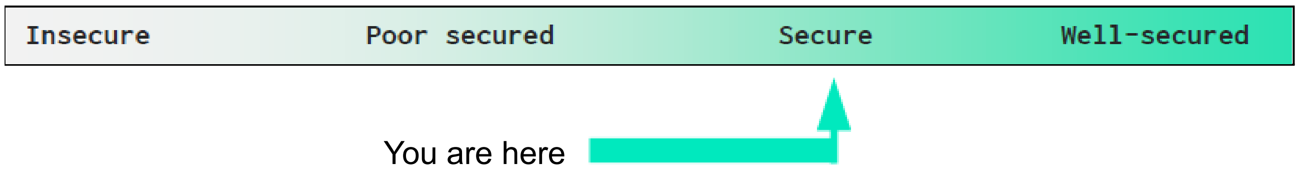
Name	**Code Review and Security Analysis Report for iBG Protocol Smart Contracts**
Platform	**Ethereum / Solidity**
File 1	IBGETHMasterChef.sol
File 1 Smart Contract Code	https://etherscan.io/address/0x4f1983038A682201eF5CeAcff656b2b3d9AF16D4#code
File 2	IBGStakingReserve.sol
File 2 Smart Contract Code	https://etherscan.io/address/0x5c0d83caf296cee593b0809c7a9a77e076ba1e62#code
Audit Date	August 29th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
File 1: IBGETHMasterChef.sol ● Maximum Rewards: 9,352,007 ● Maximum Fee: 5% ● IBG per Block: 0.7 ● Bonus Multiplier: 1 ● IBG tokens are transferred from the Staking Reserve contract while depositing, withdrawing and harvesting.	**YES, This is valid.**
File 2: IBGStakingReserve.sol ● Owner can set operator, who can withdraw IBG Tokens ● IBG Tokens must be deposited into this contract manually	**YES, This is valid.** **Owner wallet private key must be handled securely.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.



We used various tools like MythX, Slither 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 the AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 1 low and some very low-level issues.

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	Moderated
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Random number generation/use vulnerability	Passed
	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	Moderated
	Other code specification issues	Moderated
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

Code Quality

These audit scope have 2 smart contracts. These smart contracts also contain Libraries, Smart contracts inherits and Interfaces. These are compact and well written contracts.

The libraries in the iBG contracts 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 iBG contracts.

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

Some code parts are not well commented on smart contracts.

Documentation

We were given an iBG masterChef and staking reserve smart contracts code in the form of an Etherscan web link. The details of that code are mentioned above in the table.

As mentioned above, some code parts are **not well** commented. So, it is difficult 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. And their core code blocks are written well.

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

AS-IS overview

IBGETHMasterChef.sol

(1) Interface

(a) IBEP20

(2) Inherited contracts

(a) Ownable

(b) IBGToken

(3) Usages

(a) using SafeMath for uint256;

(b) using SafeBEP20 for IBEP20;

(4) Struct

(a) UserInfo

(b) PoolInfo

(5) Events

(a) event Deposit(address indexed user, uint256 indexed pid, uint256 amount);

(b) event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);

(c) event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	read	Passed	No Issue
2	poolLength	external	Passed	No Issue
3	add	write	Function input parameters lack of check	Refer audit finding section
4	set	write	Function input parameters lack of check	Refer audit finding section
5	getMultiplier	write	Passed	No Issue

6	pendingIBG	external	Function input parameters lack of check	Refer audit finding section
7	massUpdatePools	write	Infinite loops possibility at multiple places	Refer audit finding section
8	updatePool	write	Passed	No Issue
9	deposit	write	Function input parameters lack of check	Refer audit finding section
10	withdraw	write	Function input parameters lack of check	Refer audit finding section
11	emergencyWithdraw	write	Passed	No Issue
12	safeIBGTransfer	internal	Passed	No Issue
13	dev	write	Function input parameters lack of check	Refer audit finding section
14	setFeeAddress	write	Function input parameters lack of check	Refer audit finding section
15	updateEmissionRate	write	Function input parameters lack of check	Refer audit finding section
16	owner	read	Passed	No Issue
17	onlyOwner	modifier	Passed	No Issue
18	renounceOwnership	write	access only Owner	No Issue
19	transferOwnership	write	access only Owner	No Issue
20	_transferOwnership	internal	Passed	No Issue

IBGStakingReserve.sol

(1) Interface

(a) IERC20

(2) Inherited contracts

(a) Ownable

(3) Usages

(b) using SafeERC20 for IERC20;

(c) using SafeMath for uint256;

(4) Functions

Sl.	Functions	Type	Observation	Conclusion
1	constructor	write	Passed	No Issue
2	addOrRemoveOperators	write	Owner function	No Issue
3	getIBGBalance	read	Passed	No Issue
4	withdrawIBG	write	Privileged function	No Issue

Severity Definitions

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

Audit Findings

IBGETHMasterChef.sol

Critical

No Critical severity vulnerabilities were found.

High

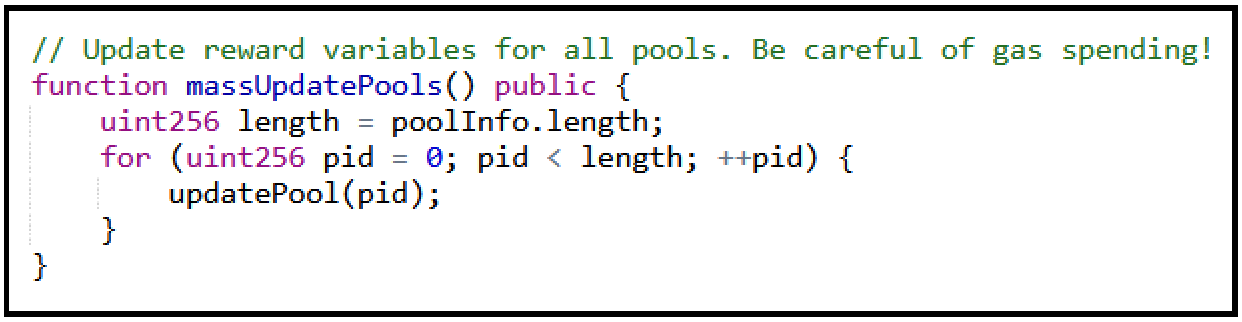
No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Infinite loops possibility at multiple places:



```
// Update reward variables for all pools. Be careful of gas spending!
function massUpdatePools() public {
    uint256 length = poolInfo.length;
    for (uint256 pid = 0; pid < length; ++pid) {
        updatePool(pid);
    }
}
```

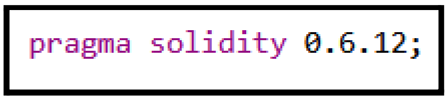
There is a function massUpdatePools(), in the smart contracts, where the poolInfo.length variable is used directly in the loop. It is recommended to put some kind of limits.

Resolution: In practical scenarios, there would not be very many pools, so it does not create an issue. But it is best practice to put some limits on the number of pools.

Status: acknowledged

Very Low / Discussion / Best practices:

(1) Use the latest solidity version:



```
pragma solidity 0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.7 which is the latest version.

Status: acknowledged

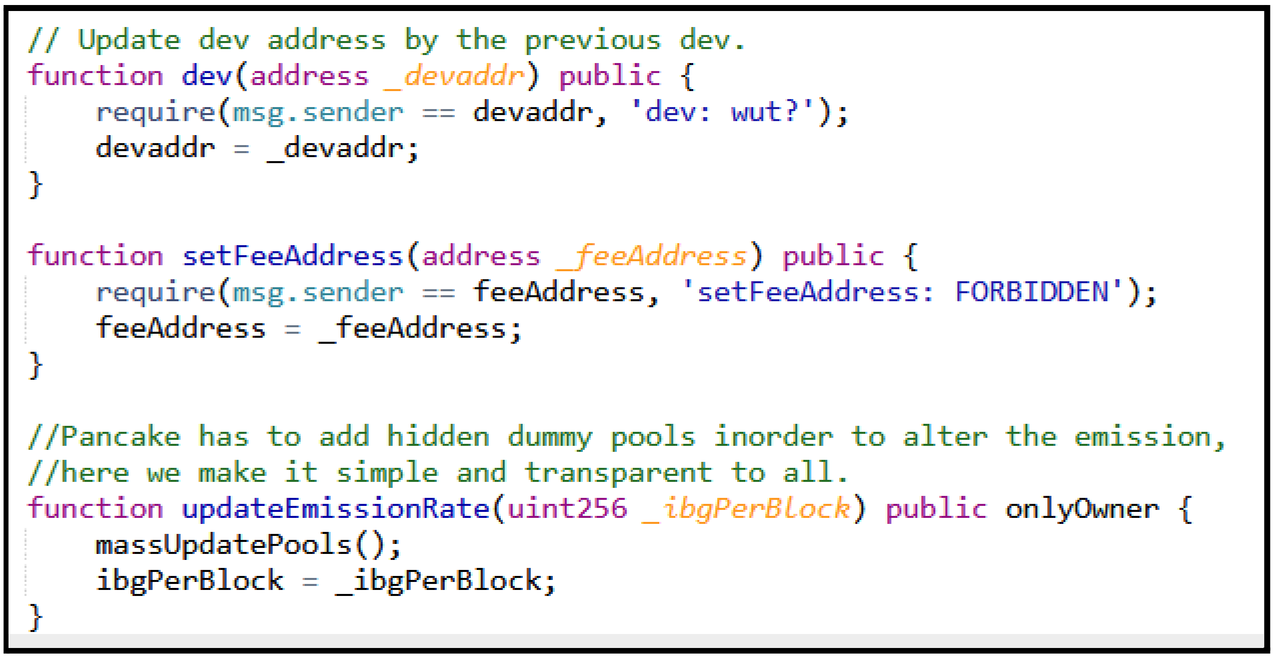
(2) Make variables constant:

These variable values MAX_REWARDS, MAX_FEE It will be unchanged. So, please make it constant. It will save some gas.

Resolution: Declare those variables as constant. Just put a constant keyword. And define constants in the constructor.

Status: acknowledged

(3) Function input parameters lack of check:



```
// Update dev address by the previous dev.
function dev(address _devaddr) public {
    require(msg.sender == devaddr, 'dev: wut?');
    devaddr = _devaddr;
}

function setFeeAddress(address _feeAddress) public {
    require(msg.sender == feeAddress, 'setFeeAddress: FORBIDDEN');
    feeAddress = _feeAddress;
}

//Pancake has to add hidden dummy pools inorder to alter the emission,
//here we make it simple and transparent to all.
function updateEmissionRate(uint256 _ibgPerBlock) public onlyOwner {
    massUpdatePools();
    ibgPerBlock = _ibgPerBlock;
}
```

Variable validation is not performed in some functions.

Resolution: There should be some validations to check the variable is not empty or greater than 0:

- dev() - _devaddr - variable is not empty and > 0
- setFeeAddress() - _feeAddress - variable is not empty and > 0
- updateEmissionRate() - _ibgPerBlock - variable is not empty and > 0
- safeIBGTransfer() - _amount – variable is not empty and > 0
- safeIBGTransfer() - _to – variable is not checked address(0)
- withdraw() - _pid – variable is not empty and > 0
- withdraw() - _amount – variable is not empty and > 0
- deposit() - _pid – variable is not empty and > 0
- pendingIBG() - _user – variable is not checked address(0)
- set() - _allocPoint - variable is not empty and > 0
- set() - _depositFeeBP - variable is not empty and > 0
- add() - _lpToken - variable is not checked address(0)
- add() - _depositFeeBP - variable is not empty and > 0

Status: acknowledged

IBGStakingReserve.sol

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Discussion / Best practices:

(1) Use the latest solidity version:

```
pragma solidity 0.6.12;
```

Using the latest solidity will prevent any compiler-level bugs.

Resolution: Please use 0.8.7 which is the latest version.

Status: acknowledged

(2) Unused libraries

```
using SafeERC20 for IERC20;
```

The SafeERC20 and SafeMath libraries are not used in the code. Although this does not create any security vulnerability, it is better to remove that if not needed, or better use that in the code.

Resolution: Either remove it, or use it in the code.

Status: acknowledged

Centralization

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

- add: The IBGETHMasterChef owner can add a new pool.
- set: The IBGETHMasterChef owner can update the given pool's IBG allocation point and deposit fee.
- updateEmissionRate: The IBGETHMasterChef owner can update the emission rate.
- addOrRemoveOperators: The IBGStakingReserve owner can add an operator wallet who can withdraw IBG tokens.
- transferOwnership: Both contract owners can transfer ownership to any other wallet.
- renounceOwnership: Both contract owners can give up the ownership.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues, but they are not critical. 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 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".**

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

AntiHACK.me Disclaimer

AntiHACK.me 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).

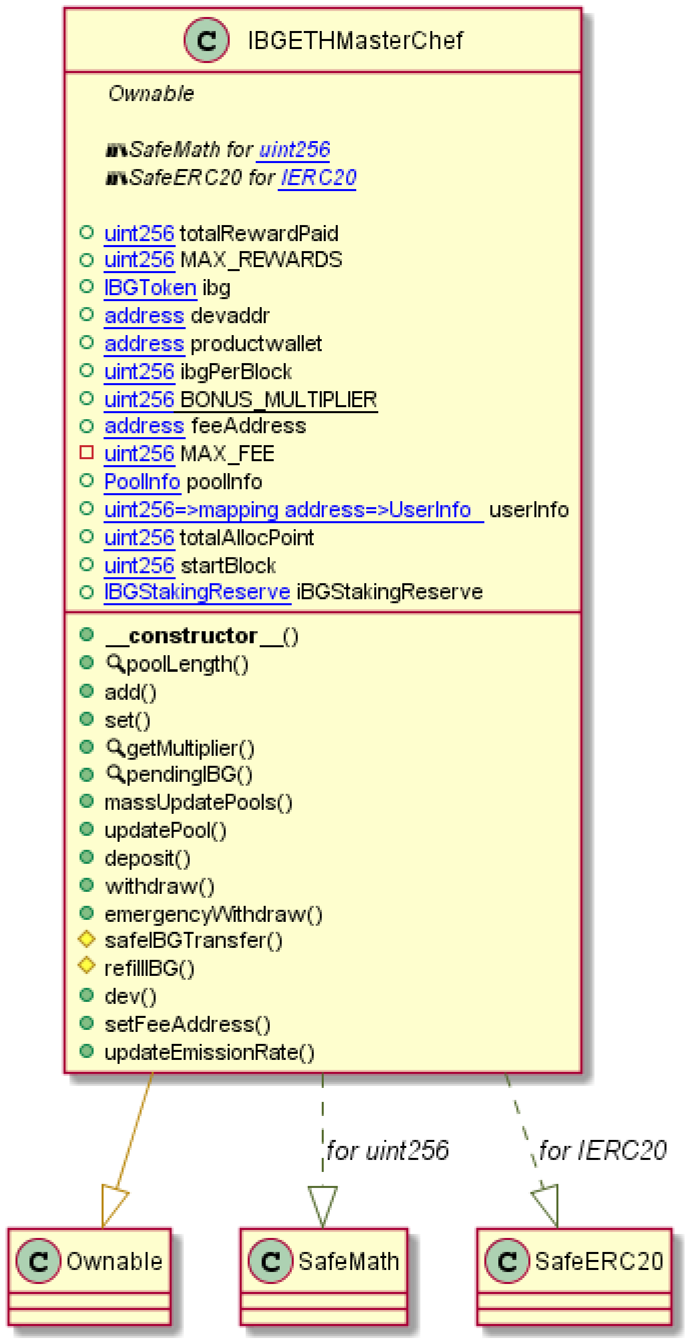
Because the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our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

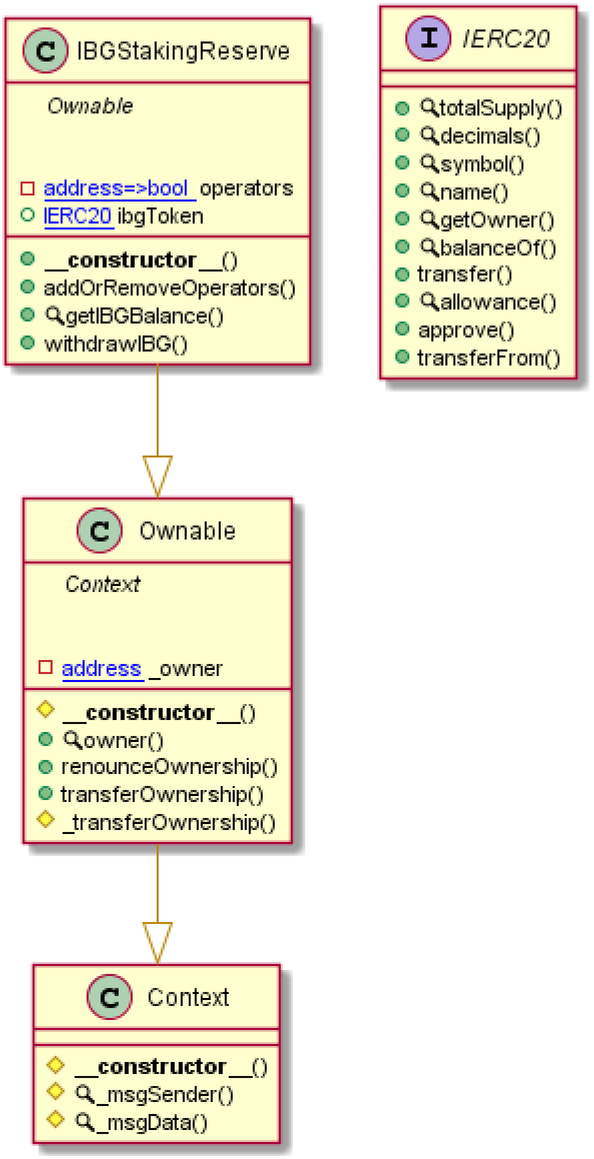
Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - IBGETHMasterChef Diagram

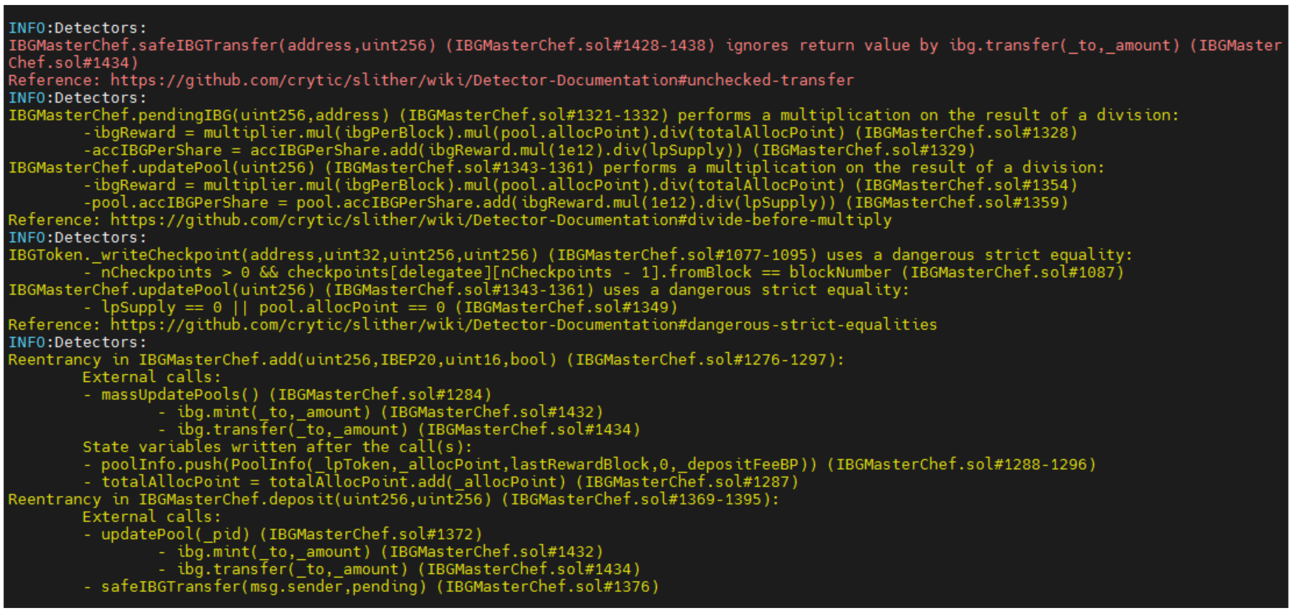


IBGStakingReserve Diagram

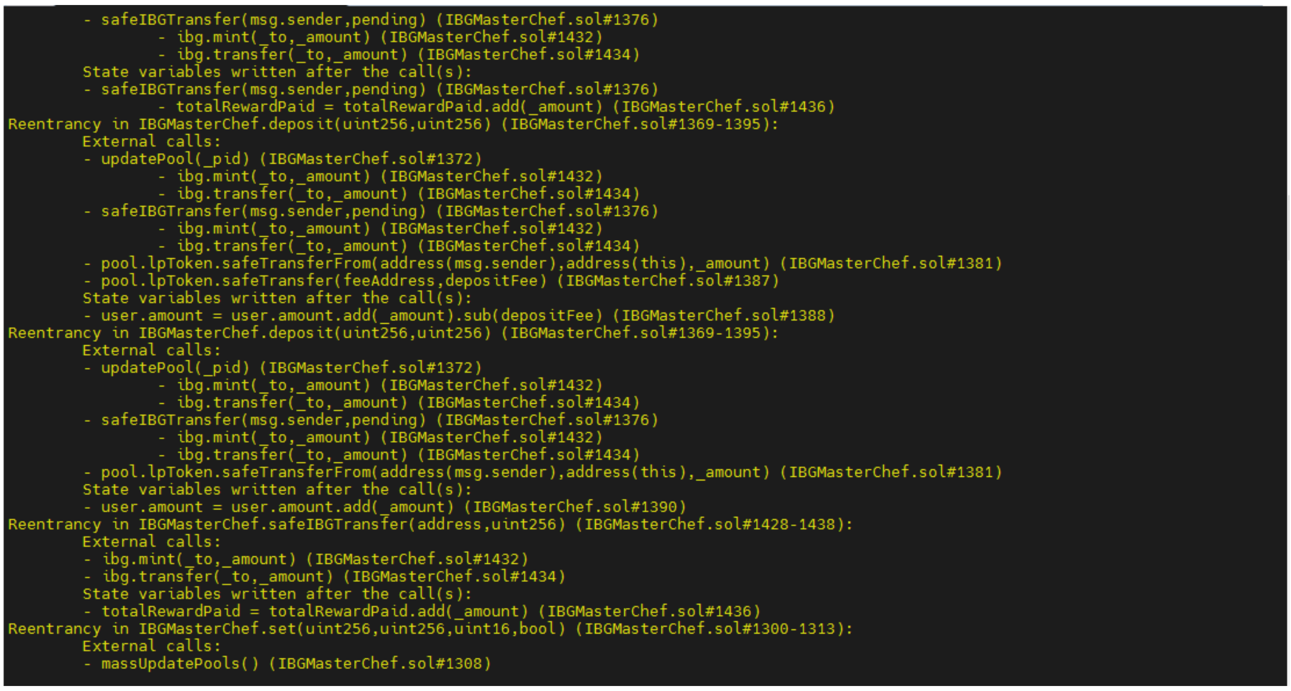


Slither Results Log

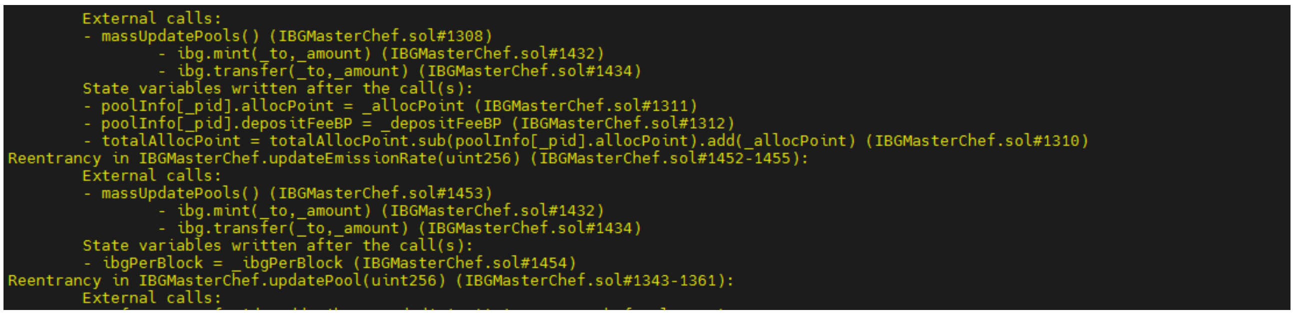
Slither log >> IBGETHMasterChef.sol



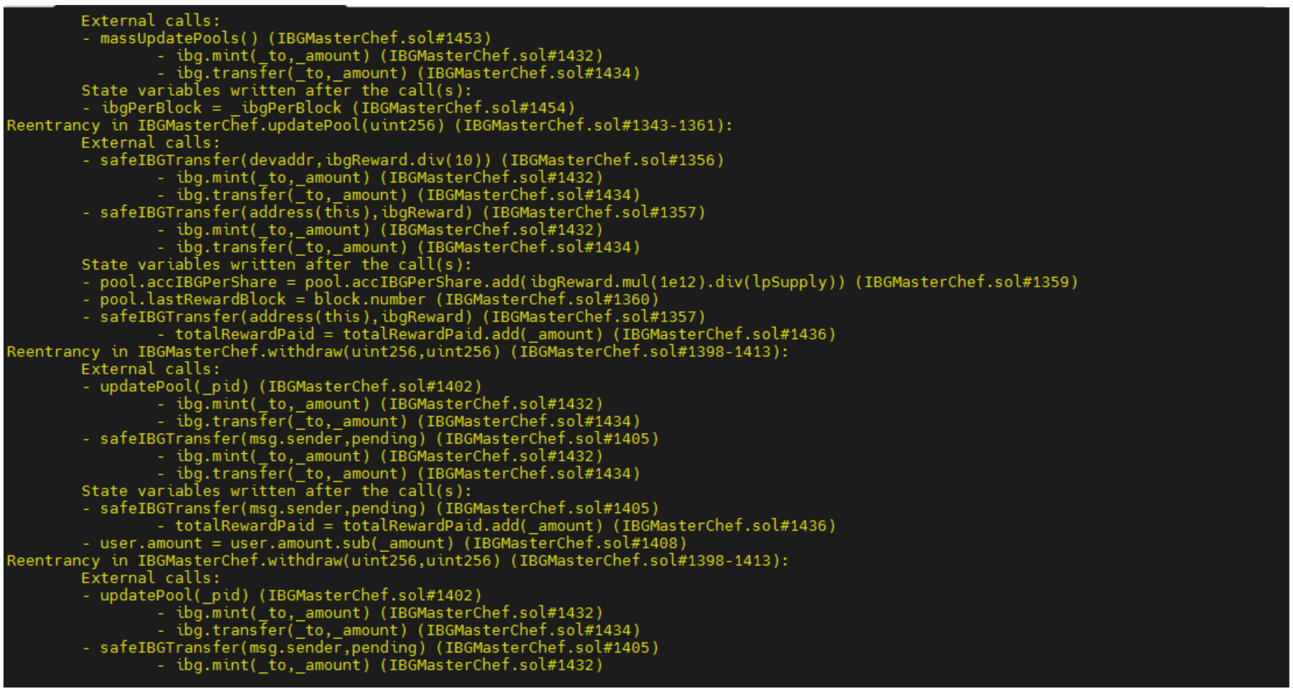
```
INFO:Detectors:
IBGMasterChef.safeIBGTransfer(address,uint256) (IBGMasterChef.sol#1428-1438) ignores return value by ibg.transfer(_to,_amount) (IBGMaster
Chef.sol#1434)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
IBGMasterChef.pendingIBG(uint256,address) (IBGMasterChef.sol#1321-1332) performs a multiplication on the result of a division:
        -ibgReward = multiplier.mul(ibgPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (IBGMasterChef.sol#1328)
        -accIBGPerShare = accIBGPerShare.add(ibgReward.mul(1e12).div(lpSupply)) (IBGMasterChef.sol#1329)
IBGMasterChef.updatePool(uint256) (IBGMasterChef.sol#1343-1361) performs a multiplication on the result of a division:
        -ibgReward = multiplier.mul(ibgPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (IBGMasterChef.sol#1354)
        -pool.accIBGPerShare = pool.accIBGPerShare.add(ibgReward.mul(1e12).div(lpSupply)) (IBGMasterChef.sol#1359)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
IBGToken._writeCheckpoint(address,uint32,uint256,uint256) (IBGMasterChef.sol#1077-1095) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (IBGMasterChef.sol#1087)
IBGMasterChef.updatePool(uint256) (IBGMasterChef.sol#1343-1361) uses a dangerous strict equality:
        - lpSupply == 0 || pool.allocPoint == 0 (IBGMasterChef.sol#1349)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in IBGMasterChef.add(uint256,IBEP20,uint16,bool) (IBGMasterChef.sol#1276-1297):
        External calls:
        - massUpdatePools() (IBGMasterChef.sol#1284)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,_depositFeeBP)) (IBGMasterChef.sol#1288-1296)
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (IBGMasterChef.sol#1287)
Reentrancy in IBGMasterChef.deposit(uint256,uint256) (IBGMasterChef.sol#1369-1395):
        External calls:
        - updatePool(_pid) (IBGMasterChef.sol#1372)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1376)
```



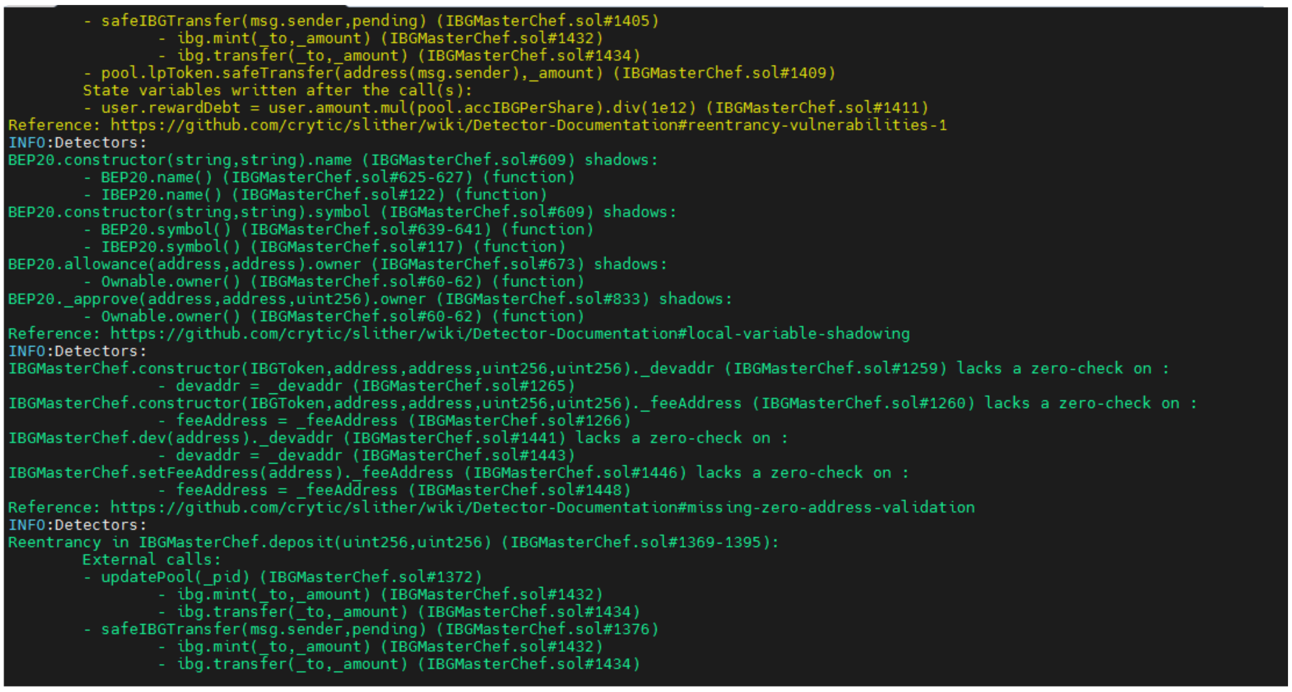
```
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1376)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        State variables written after the call(s):
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1376)
                - totalRewardPaid = totalRewardPaid.add(_amount) (IBGMasterChef.sol#1436)
Reentrancy in IBGMasterChef.deposit(uint256,uint256) (IBGMasterChef.sol#1369-1395):
        External calls:
        - updatePool(_pid) (IBGMasterChef.sol#1372)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1376)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (IBGMasterChef.sol#1381)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (IBGMasterChef.sol#1387)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount).sub(depositFee) (IBGMasterChef.sol#1388)
Reentrancy in IBGMasterChef.deposit(uint256,uint256) (IBGMasterChef.sol#1369-1395):
        External calls:
        - updatePool(_pid) (IBGMasterChef.sol#1372)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1376)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (IBGMasterChef.sol#1381)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (IBGMasterChef.sol#1390)
Reentrancy in IBGMasterChef.safeIBGTransfer(address,uint256) (IBGMasterChef.sol#1428-1438):
        External calls:
        - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
        - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        State variables written after the call(s):
        - totalRewardPaid = totalRewardPaid.add(_amount) (IBGMasterChef.sol#1436)
Reentrancy in IBGMasterChef.set(uint256,uint256,uint16,bool) (IBGMasterChef.sol#1300-1313):
        External calls:
        - massUpdatePools() (IBGMasterChef.sol#1308)
```



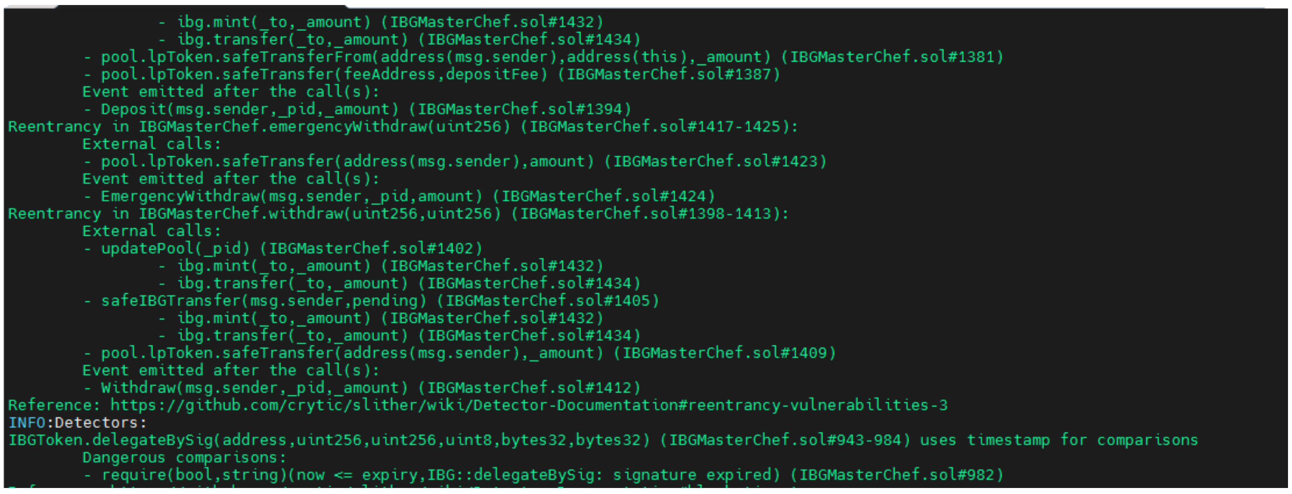
```
        External calls:
        - massUpdatePools() (IBGMasterChef.sol#1308)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        State variables written after the call(s):
        - poolInfo[_pid].allocPoint = _allocPoint (IBGMasterChef.sol#1311)
        - poolInfo[_pid].depositFeeBP = _depositFeeBP (IBGMasterChef.sol#1312)
        - totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (IBGMasterChef.sol#1310)
Reentrancy in IBGMasterChef.updateEmissionRate(uint256) (IBGMasterChef.sol#1452-1455):
        External calls:
        - massUpdatePools() (IBGMasterChef.sol#1453)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        State variables written after the call(s):
        - ibgPerBlock = _ibgPerBlock (IBGMasterChef.sol#1454)
Reentrancy in IBGMasterChef.updatePool(uint256) (IBGMasterChef.sol#1343-1361):
        External calls:
```



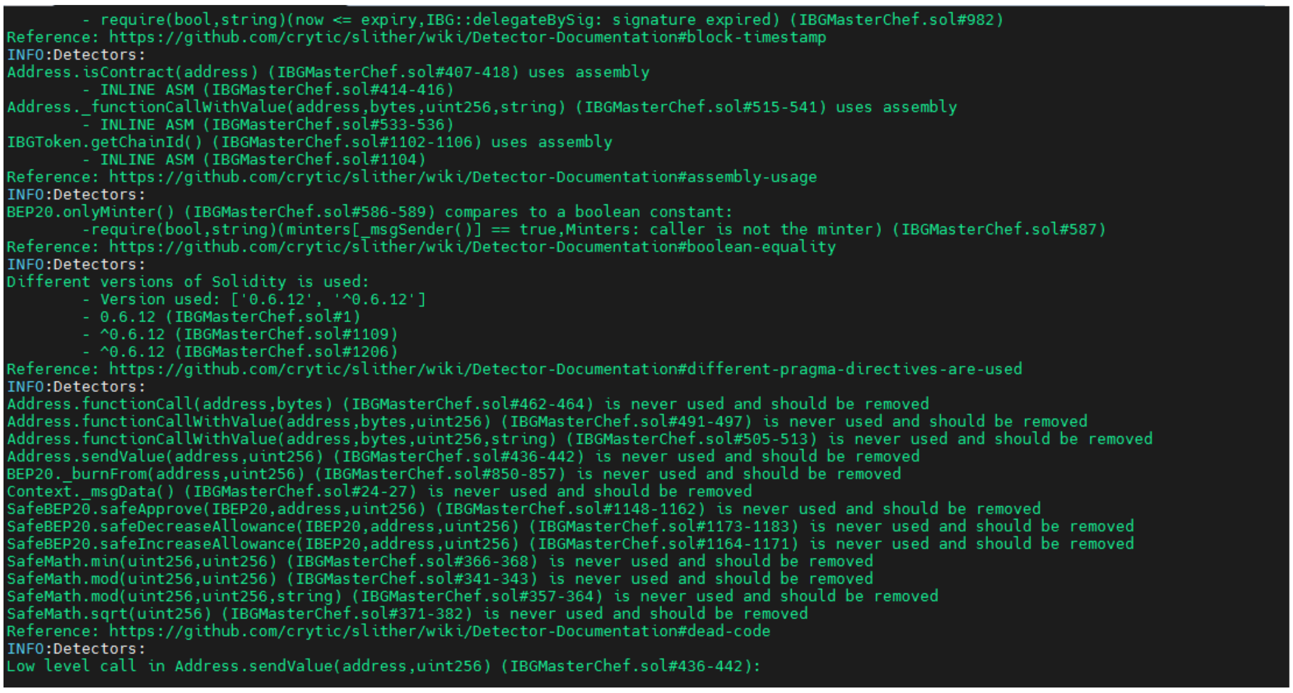
```
        External calls:
        - massUpdatePools() (IBGMasterChef.sol#1453)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        State variables written after the call(s):
        - ibgPerBlock = _ibgPerBlock (IBGMasterChef.sol#1454)
Reentrancy in IBGMasterChef.updatePool(uint256) (IBGMasterChef.sol#1343-1361):
        External calls:
        - safeIBGTransfer(devaddr,ibgReward.div(10)) (IBGMasterChef.sol#1356)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - safeIBGTransfer(address(this),ibgReward) (IBGMasterChef.sol#1357)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        State variables written after the call(s):
        - pool.accIBGPerShare = pool.accIBGPerShare.add(ibgReward.mul(1e12).div(lpSupply)) (IBGMasterChef.sol#1359)
        - pool.lastRewardBlock = block.number (IBGMasterChef.sol#1360)
        - safeIBGTransfer(address(this),ibgReward) (IBGMasterChef.sol#1357)
                - totalRewardPaid = totalRewardPaid.add(_amount) (IBGMasterChef.sol#1436)
Reentrancy in IBGMasterChef.withdraw(uint256,uint256) (IBGMasterChef.sol#1398-1413):
        External calls:
        - updatePool(_pid) (IBGMasterChef.sol#1402)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1405)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        State variables written after the call(s):
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1405)
                - totalRewardPaid = totalRewardPaid.add(_amount) (IBGMasterChef.sol#1436)
        - user.amount = user.amount.sub(_amount) (IBGMasterChef.sol#1408)
Reentrancy in IBGMasterChef.withdraw(uint256,uint256) (IBGMasterChef.sol#1398-1413):
        External calls:
        - updatePool(_pid) (IBGMasterChef.sol#1402)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1405)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
```



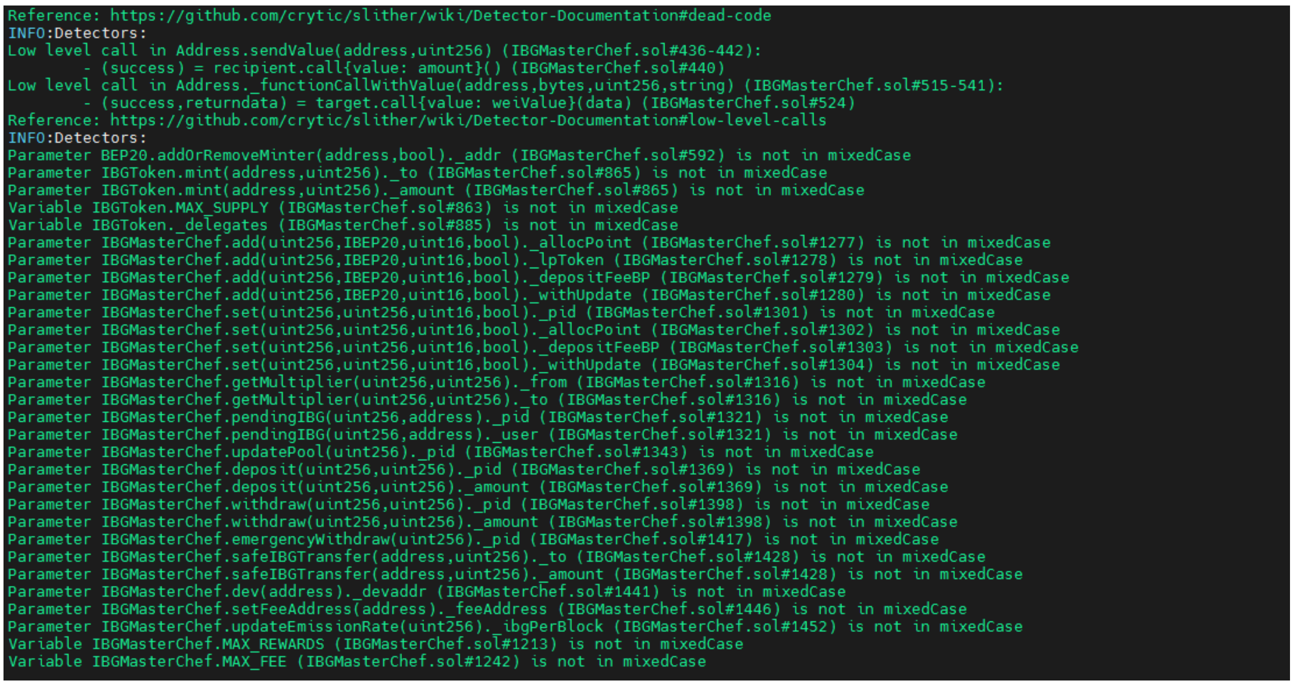
```
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1405)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (IBGMasterChef.sol#1409)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accIBGPerShare).div(1e12) (IBGMasterChef.sol#1411)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
BEP20.constructor(string,string).name (IBGMasterChef.sol#609) shadows:
        - BEP20.name() (IBGMasterChef.sol#625-627) (function)
        - IBEP20.name() (IBGMasterChef.sol#122) (function)
BEP20.constructor(string,string).symbol (IBGMasterChef.sol#609) shadows:
        - BEP20.symbol() (IBGMasterChef.sol#639-641) (function)
        - IBEP20.symbol() (IBGMasterChef.sol#117) (function)
BEP20.allowance(address,address).owner (IBGMasterChef.sol#673) shadows:
        - Ownable.owner() (IBGMasterChef.sol#60-62) (function)
BEP20._approve(address,address,uint256).owner (IBGMasterChef.sol#833) shadows:
        - Ownable.owner() (IBGMasterChef.sol#60-62) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
IBGMasterChef.constructor(IBGToken,address,address,uint256,uint256)._devaddr (IBGMasterChef.sol#1259) lacks a zero-check on :
                - devaddr = _devaddr (IBGMasterChef.sol#1265)
IBGMasterChef.constructor(IBGToken,address,address,uint256,uint256)._feeAddress (IBGMasterChef.sol#1260) lacks a zero-check on :
                - feeAddress = _feeAddress (IBGMasterChef.sol#1266)
IBGMasterChef.dev(address)._devaddr (IBGMasterChef.sol#1441) lacks a zero-check on :
                - devaddr = _devaddr (IBGMasterChef.sol#1443)
IBGMasterChef.setFeeAddress(address)._feeAddress (IBGMasterChef.sol#1446) lacks a zero-check on :
                - feeAddress = _feeAddress (IBGMasterChef.sol#1448)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in IBGMasterChef.deposit(uint256,uint256) (IBGMasterChef.sol#1369-1395):
        External calls:
        - updatePool(_pid) (IBGMasterChef.sol#1372)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1376)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
```



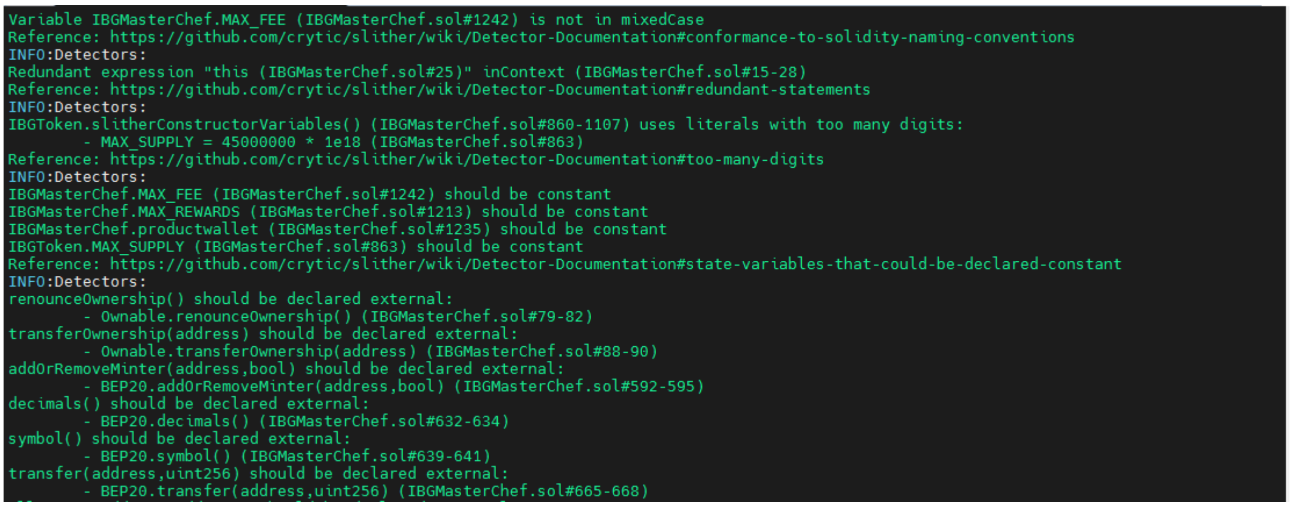
```
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (IBGMasterChef.sol#1381)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (IBGMasterChef.sol#1387)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (IBGMasterChef.sol#1394)
Reentrancy in IBGMasterChef.emergencyWithdraw(uint256) (IBGMasterChef.sol#1417-1425):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),amount) (IBGMasterChef.sol#1423)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,amount) (IBGMasterChef.sol#1424)
Reentrancy in IBGMasterChef.withdraw(uint256,uint256) (IBGMasterChef.sol#1398-1413):
        External calls:
        - updatePool(_pid) (IBGMasterChef.sol#1402)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - safeIBGTransfer(msg.sender,pending) (IBGMasterChef.sol#1405)
                - ibg.mint(_to,_amount) (IBGMasterChef.sol#1432)
                - ibg.transfer(_to,_amount) (IBGMasterChef.sol#1434)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (IBGMasterChef.sol#1409)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (IBGMasterChef.sol#1412)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
IBGToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (IBGMasterChef.sol#943-984) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,IBG::delegateBySig: signature expired) (IBGMasterChef.sol#982)
```



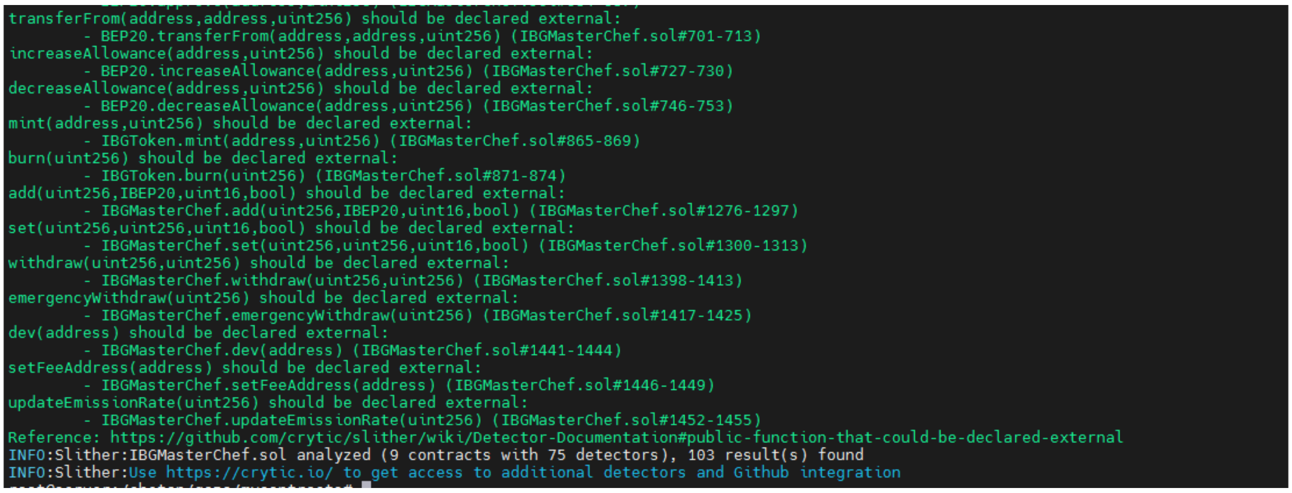
```
        - require(bool,string)(now <= expiry,IBG::delegateBySig: signature expired) (IBGMasterChef.sol#982)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (IBGMasterChef.sol#407-418) uses assembly
        - INLINE ASM (IBGMasterChef.sol#414-416)
Address._functionCallWithValue(address,bytes,uint256,string) (IBGMasterChef.sol#515-541) uses assembly
        - INLINE ASM (IBGMasterChef.sol#533-536)
IBGToken.getChainId() (IBGMasterChef.sol#1102-1106) uses assembly
        - INLINE ASM (IBGMasterChef.sol#1104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
BEP20.onlyMinter() (IBGMasterChef.sol#586-589) compares to a boolean constant:
        -require(bool,string)(minters[_msgSender()] == true,Minters: caller is not the minter) (IBGMasterChef.sol#587)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.6.12', '^0.6.12']
        - 0.6.12 (IBGMasterChef.sol#1)
        - ^0.6.12 (IBGMasterChef.sol#1109)
        - ^0.6.12 (IBGMasterChef.sol#1206)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (IBGMasterChef.sol#462-464) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (IBGMasterChef.sol#491-497) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (IBGMasterChef.sol#505-513) is never used and should be removed
Address.sendValue(address,uint256) (IBGMasterChef.sol#436-442) is never used and should be removed
BEP20._burnFrom(address,uint256) (IBGMasterChef.sol#850-857) is never used and should be removed
Context._msgData() (IBGMasterChef.sol#24-27) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (IBGMasterChef.sol#1148-1162) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (IBGMasterChef.sol#1173-1183) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (IBGMasterChef.sol#1164-1171) is never used and should be removed
SafeMath.min(uint256,uint256) (IBGMasterChef.sol#366-368) is never used and should be removed
SafeMath.mod(uint256,uint256) (IBGMasterChef.sol#341-343) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (IBGMasterChef.sol#357-364) is never used and should be removed
SafeMath.sqrt(uint256) (IBGMasterChef.sol#371-382) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (IBGMasterChef.sol#436-442):
```



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (IBGMasterChef.sol#436-442):
        - (success) = recipient.call{value: amount}() (IBGMasterChef.sol#440)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (IBGMasterChef.sol#515-541):
        - (success,returndata) = target.call{value: weiValue}(data) (IBGMasterChef.sol#524)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter BEP20.addOrRemoveMinter(address,bool)._addr (IBGMasterChef.sol#592) is not in mixedCase
Parameter IBGToken.mint(address,uint256)._to (IBGMasterChef.sol#865) is not in mixedCase
Parameter IBGToken.mint(address,uint256)._amount (IBGMasterChef.sol#865) is not in mixedCase
Variable IBGToken.MAX_SUPPLY (IBGMasterChef.sol#863) is not in mixedCase
Variable IBGToken._delegates (IBGMasterChef.sol#885) is not in mixedCase
Parameter IBGMasterChef.add(uint256,IBEP20,uint16,bool)._allocPoint (IBGMasterChef.sol#1277) is not in mixedCase
Parameter IBGMasterChef.add(uint256,IBEP20,uint16,bool)._lpToken (IBGMasterChef.sol#1278) is not in mixedCase
Parameter IBGMasterChef.add(uint256,IBEP20,uint16,bool)._depositFeeBP (IBGMasterChef.sol#1279) is not in mixedCase
Parameter IBGMasterChef.add(uint256,IBEP20,uint16,bool)._withUpdate (IBGMasterChef.sol#1280) is not in mixedCase
Parameter IBGMasterChef.set(uint256,uint256,uint16,bool)._pid (IBGMasterChef.sol#1301) is not in mixedCase
Parameter IBGMasterChef.set(uint256,uint256,uint16,bool)._allocPoint (IBGMasterChef.sol#1302) is not in mixedCase
Parameter IBGMasterChef.set(uint256,uint256,uint16,bool)._depositFeeBP (IBGMasterChef.sol#1303) is not in mixedCase
Parameter IBGMasterChef.set(uint256,uint256,uint16,bool)._withUpdate (IBGMasterChef.sol#1304) is not in mixedCase
Parameter IBGMasterChef.getMultiplier(uint256,uint256)._from (IBGMasterChef.sol#1316) is not in mixedCase
Parameter IBGMasterChef.getMultiplier(uint256,uint256)._to (IBGMasterChef.sol#1316) is not in mixedCase
Parameter IBGMasterChef.pendingIBG(uint256,address)._pid (IBGMasterChef.sol#1321) is not in mixedCase
Parameter IBGMasterChef.pendingIBG(uint256,address)._user (IBGMasterChef.sol#1321) is not in mixedCase
Parameter IBGMasterChef.updatePool(uint256)._pid (IBGMasterChef.sol#1343) is not in mixedCase
Parameter IBGMasterChef.deposit(uint256,uint256)._pid (IBGMasterChef.sol#1369) is not in mixedCase
Parameter IBGMasterChef.deposit(uint256,uint256)._amount (IBGMasterChef.sol#1369) is not in mixedCase
Parameter IBGMasterChef.withdraw(uint256,uint256)._pid (IBGMasterChef.sol#1398) is not in mixedCase
Parameter IBGMasterChef.withdraw(uint256,uint256)._amount (IBGMasterChef.sol#1398) is not in mixedCase
Parameter IBGMasterChef.emergencyWithdraw(uint256)._pid (IBGMasterChef.sol#1417) is not in mixedCase
Parameter IBGMasterChef.safeIBGTransfer(address,uint256)._to (IBGMasterChef.sol#1428) is not in mixedCase
Parameter IBGMasterChef.safeIBGTransfer(address,uint256)._amount (IBGMasterChef.sol#1428) is not in mixedCase
Parameter IBGMasterChef.dev(address)._devaddr (IBGMasterChef.sol#1441) is not in mixedCase
Parameter IBGMasterChef.setFeeAddress(address)._feeAddress (IBGMasterChef.sol#1446) is not in mixedCase
Parameter IBGMasterChef.updateEmissionRate(uint256)._ibgPerBlock (IBGMasterChef.sol#1452) is not in mixedCase
Variable IBGMasterChef.MAX_REWARDS (IBGMasterChef.sol#1213) is not in mixedCase
Variable IBGMasterChef.MAX_FEE (IBGMasterChef.sol#1242) is not in mixedCase
```

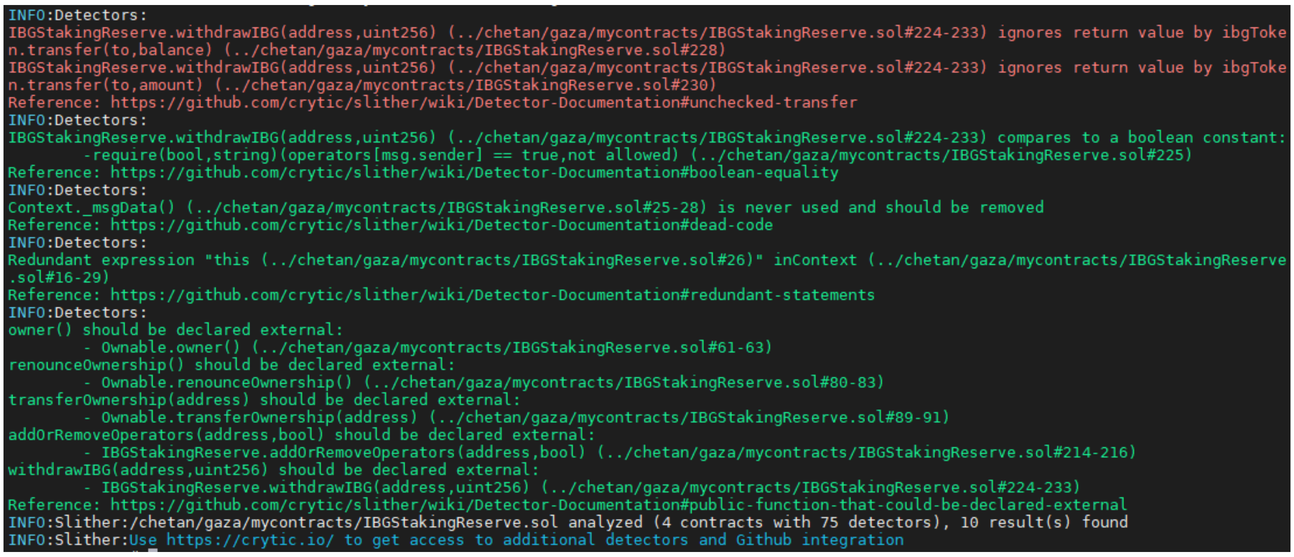


```
Variable IBGMasterChef.MAX_FEE (IBGMasterChef.sol#1242) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (IBGMasterChef.sol#25)" inContext (IBGMasterChef.sol#15-28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
IBGToken.slitherConstructorVariables() (IBGMasterChef.sol#860-1107) uses literals with too many digits:
        - MAX_SUPPLY = 45000000 * 1e18 (IBGMasterChef.sol#863)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
IBGMasterChef.MAX_FEE (IBGMasterChef.sol#1242) should be constant
IBGMasterChef.MAX_REWARDS (IBGMasterChef.sol#1213) should be constant
IBGMasterChef.productwallet (IBGMasterChef.sol#1235) should be constant
IBGToken.MAX_SUPPLY (IBGMasterChef.sol#863) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (IBGMasterChef.sol#79-82)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (IBGMasterChef.sol#88-90)
addOrRemoveMinter(address,bool) should be declared external:
        - BEP20.addOrRemoveMinter(address,bool) (IBGMasterChef.sol#592-595)
decimals() should be declared external:
        - BEP20.decimals() (IBGMasterChef.sol#632-634)
symbol() should be declared external:
        - BEP20.symbol() (IBGMasterChef.sol#639-641)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (IBGMasterChef.sol#665-668)
```



```
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (IBGMasterChef.sol#701-713)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (IBGMasterChef.sol#727-730)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (IBGMasterChef.sol#746-753)
mint(address,uint256) should be declared external:
        - IBGToken.mint(address,uint256) (IBGMasterChef.sol#865-869)
burn(uint256) should be declared external:
        - IBGToken.burn(uint256) (IBGMasterChef.sol#871-874)
add(uint256,IBEP20,uint16,bool) should be declared external:
        - IBGMasterChef.add(uint256,IBEP20,uint16,bool) (IBGMasterChef.sol#1276-1297)
set(uint256,uint256,uint16,bool) should be declared external:
        - IBGMasterChef.set(uint256,uint256,uint16,bool) (IBGMasterChef.sol#1300-1313)
withdraw(uint256,uint256) should be declared external:
        - IBGMasterChef.withdraw(uint256,uint256) (IBGMasterChef.sol#1398-1413)
emergencyWithdraw(uint256) should be declared external:
        - IBGMasterChef.emergencyWithdraw(uint256) (IBGMasterChef.sol#1417-1425)
dev(address) should be declared external:
        - IBGMasterChef.dev(address) (IBGMasterChef.sol#1441-1444)
setFeeAddress(address) should be declared external:
        - IBGMasterChef.setFeeAddress(address) (IBGMasterChef.sol#1446-1449)
updateEmissionRate(uint256) should be declared external:
        - IBGMasterChef.updateEmissionRate(uint256) (IBGMasterChef.sol#1452-1455)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:IBGMasterChef.sol analyzed (9 contracts with 75 detectors), 103 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

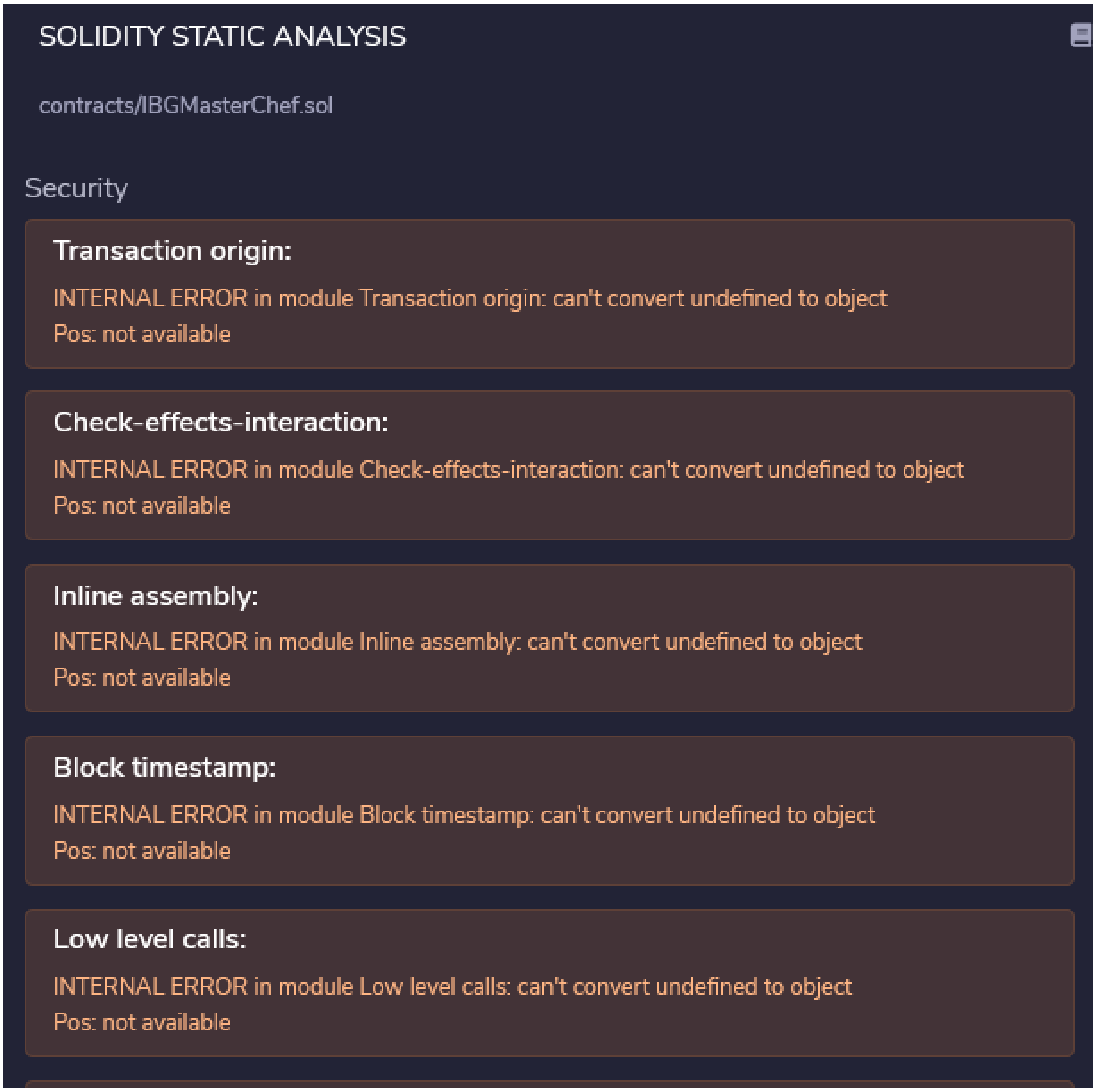
Slither log >> IBGStakingReserve.sol



```
INFO:Detectors:
IBGStakingReserve.withdrawIBG(address,uint256) (../chetan/gaza/mycontracts/IBGStakingReserve.sol#224-233) ignores return value by ibgToke
n.transfer(to,balance) (../chetan/gaza/mycontracts/IBGStakingReserve.sol#228)
IBGStakingReserve.withdrawIBG(address,uint256) (../chetan/gaza/mycontracts/IBGStakingReserve.sol#224-233) ignores return value by ibgToke
n.transfer(to,amount) (../chetan/gaza/mycontracts/IBGStakingReserve.sol#230)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
IBGStakingReserve.withdrawIBG(address,uint256) (../chetan/gaza/mycontracts/IBGStakingReserve.sol#224-233) compares to a boolean constant:
        -require(bool,string)(operators[msg.sender] == true,not allowed) (../chetan/gaza/mycontracts/IBGStakingReserve.sol#225)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Context._msgData() (../chetan/gaza/mycontracts/IBGStakingReserve.sol#25-28) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Redundant expression "this (../chetan/gaza/mycontracts/IBGStakingReserve.sol#26)" inContext (../chetan/gaza/mycontracts/IBGStakingReserve
.sol#16-29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
owner() should be declared external:
        - Ownable.owner() (../chetan/gaza/mycontracts/IBGStakingReserve.sol#61-63)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (../chetan/gaza/mycontracts/IBGStakingReserve.sol#80-83)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (../chetan/gaza/mycontracts/IBGStakingReserve.sol#89-91)
addOrRemoveOperators(address,bool) should be declared external:
        - IBGStakingReserve.addOrRemoveOperators(address,bool) (../chetan/gaza/mycontracts/IBGStakingReserve.sol#214-216)
withdrawIBG(address,uint256) should be declared external:
        - IBGStakingReserve.withdrawIBG(address,uint256) (../chetan/gaza/mycontracts/IBGStakingReserve.sol#224-233)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:/chetan/gaza/mycontracts/IBGStakingReserve.sol analyzed (4 contracts with 75 detectors), 10 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Solidity static analysis

IBGETHMasterChef.sol



SOLIDITY STATIC ANALYSIS

contracts/IBGMasterChef.sol

Security

Transaction origin:

INTERNAL ERROR in module Transaction origin: can't convert undefined to object
Pos: not available

Check-effects-interaction:

INTERNAL ERROR in module Check-effects-interaction: can't convert undefined to object
Pos: not available

Inline assembly:

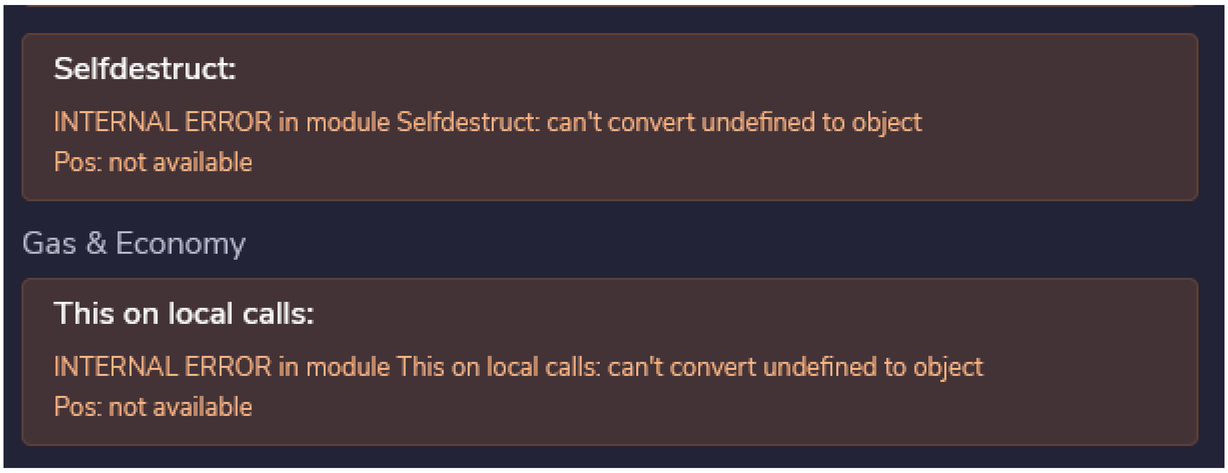
INTERNAL ERROR in module Inline assembly: can't convert undefined to object
Pos: not available

Block timestamp:

INTERNAL ERROR in module Block timestamp: can't convert undefined to object
Pos: not available

Low level calls:

INTERNAL ERROR in module Low level calls: can't convert undefined to object
Pos: not available



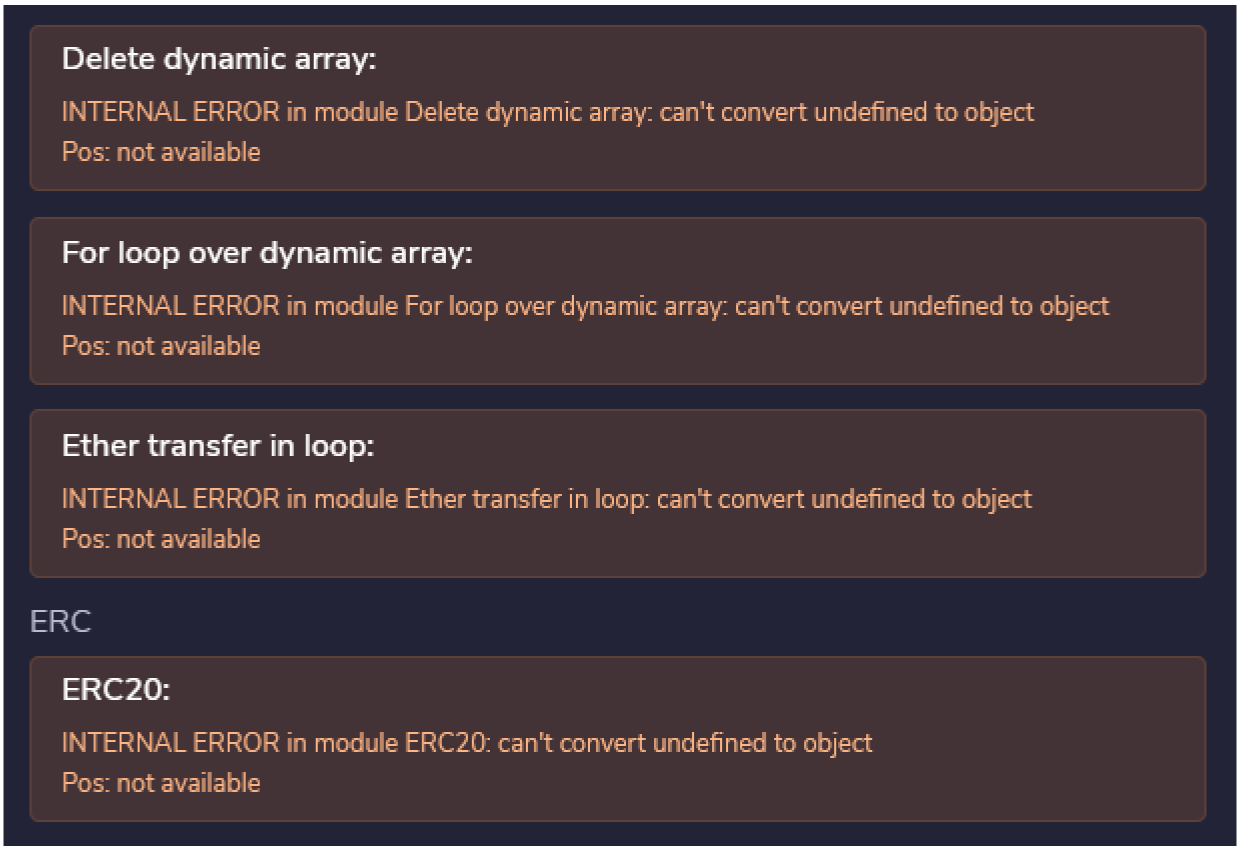
Selfdestruct:

INTERNAL ERROR in module Selfdestruct: can't convert undefined to object
Pos: not available

Gas & Economy

This on local calls:

INTERNAL ERROR in module This on local calls: can't convert undefined to object
Pos: not available



Delete dynamic array:

INTERNAL ERROR in module Delete dynamic array: can't convert undefined to object
Pos: not available

For loop over dynamic array:

INTERNAL ERROR in module For loop over dynamic array: can't convert undefined to object
Pos: not available

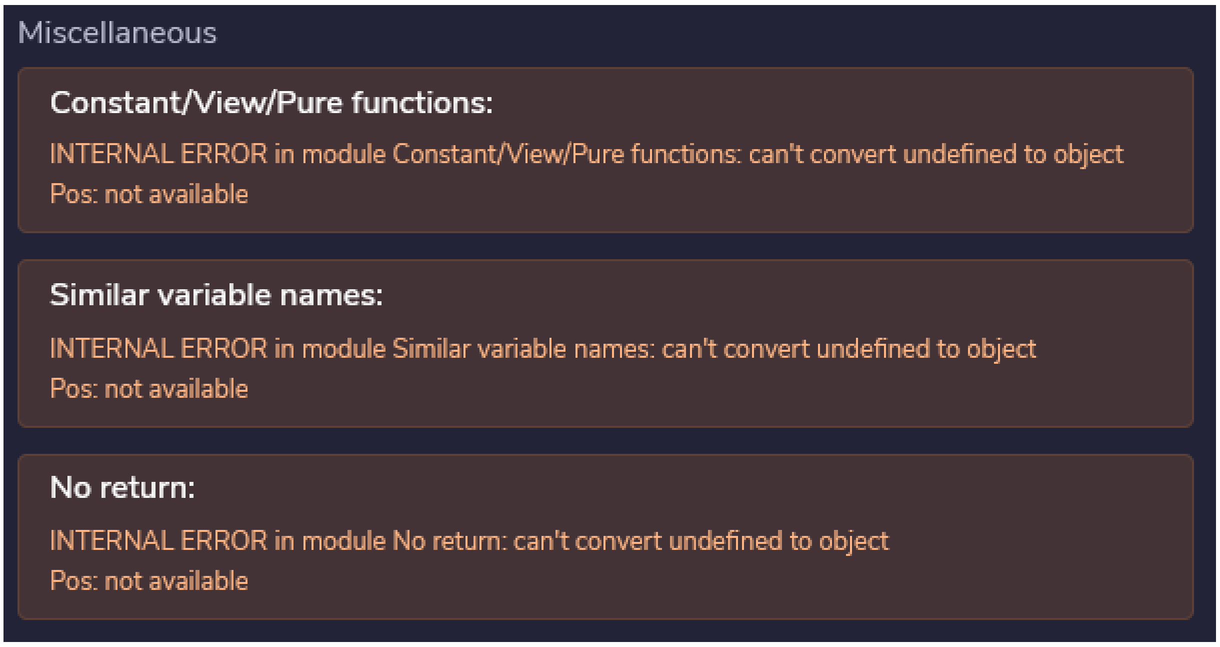
Ether transfer in loop:

INTERNAL ERROR in module Ether transfer in loop: can't convert undefined to object
Pos: not available

ERC

ERC20:

INTERNAL ERROR in module ERC20: can't convert undefined to object
Pos: not available



Miscellaneous

Constant/View/Pure functions:

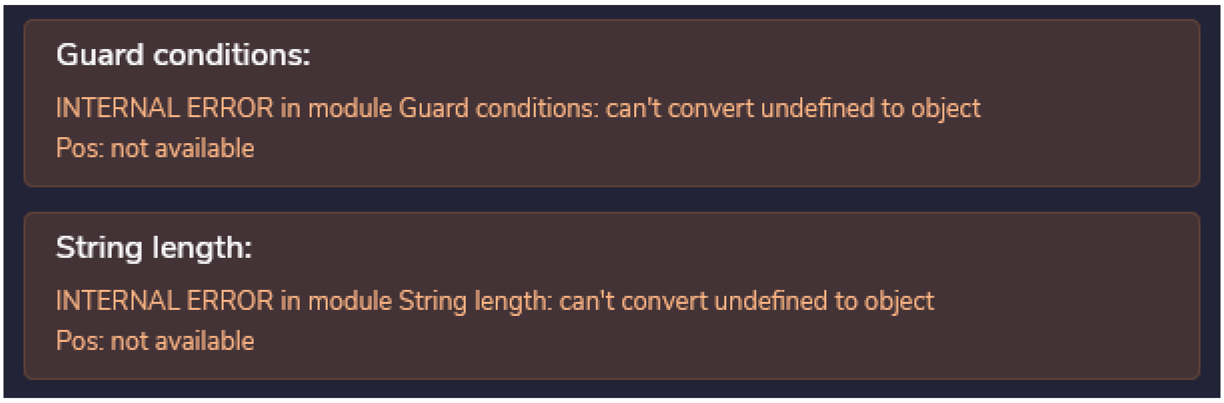
INTERNAL ERROR in module Constant/View/Pure functions: can't convert undefined to object
Pos: not available

Similar variable names:

INTERNAL ERROR in module Similar variable names: can't convert undefined to object
Pos: not available

No return:

INTERNAL ERROR in module No return: can't convert undefined to object
Pos: not available



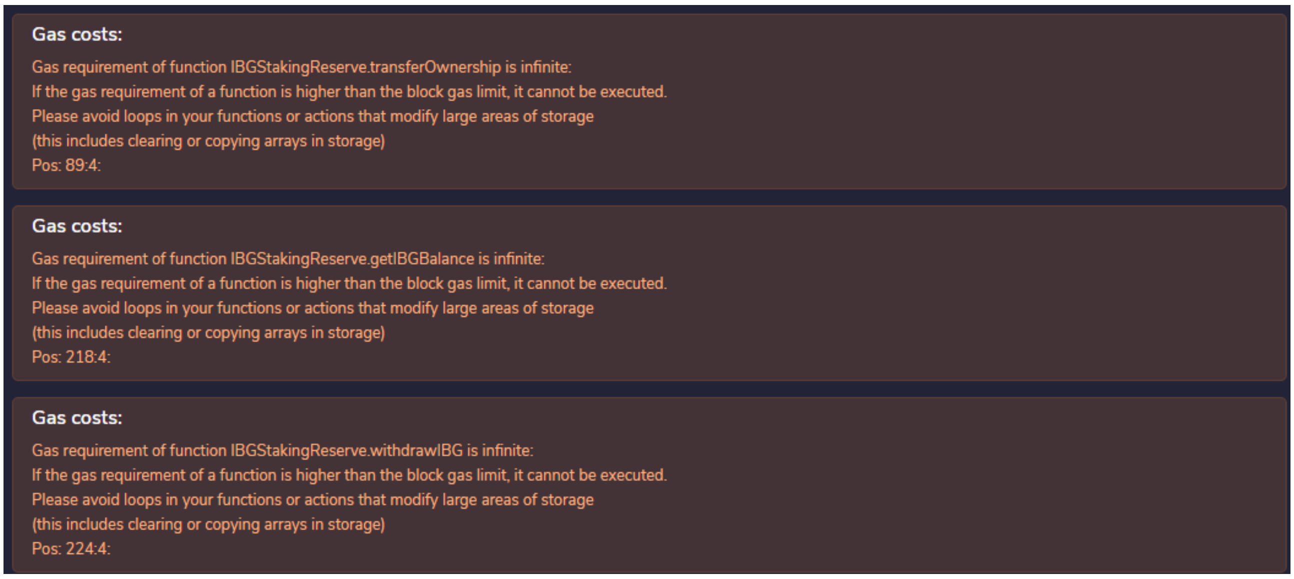
Guard conditions:

INTERNAL ERROR in module Guard conditions: can't convert undefined to object
Pos: not available

String length:

INTERNAL ERROR in module String length: can't convert undefined to object
Pos: not available

IBGStakingReserve.sol



Gas costs:

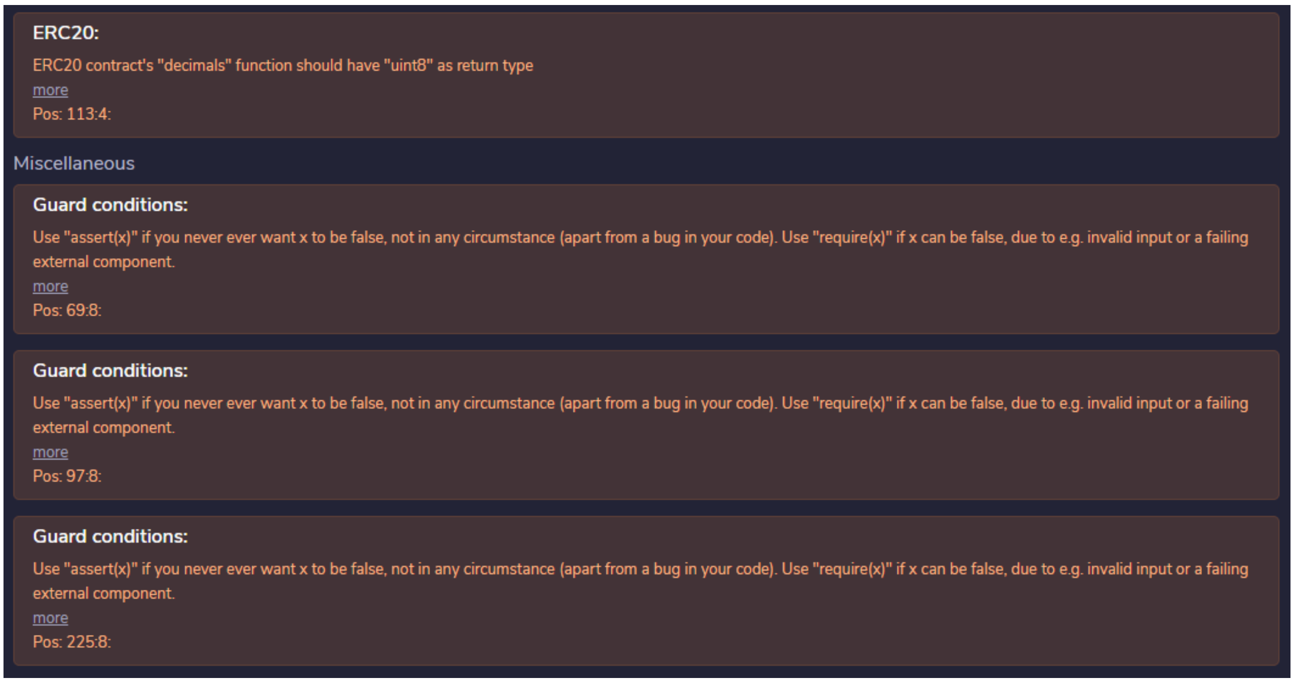
Gas requirement of function IBGStakingReserve.transferOwnership is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 89:4:

Gas costs:

Gas requirement of function IBGStakingReserve.getIBGBalance 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: 218:4:

Gas costs:

Gas requirement of function IBGStakingReserve.withdrawIBG 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: 224:4:



ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 113:4:

Miscellaneous

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: 69:8:

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: 97:8:

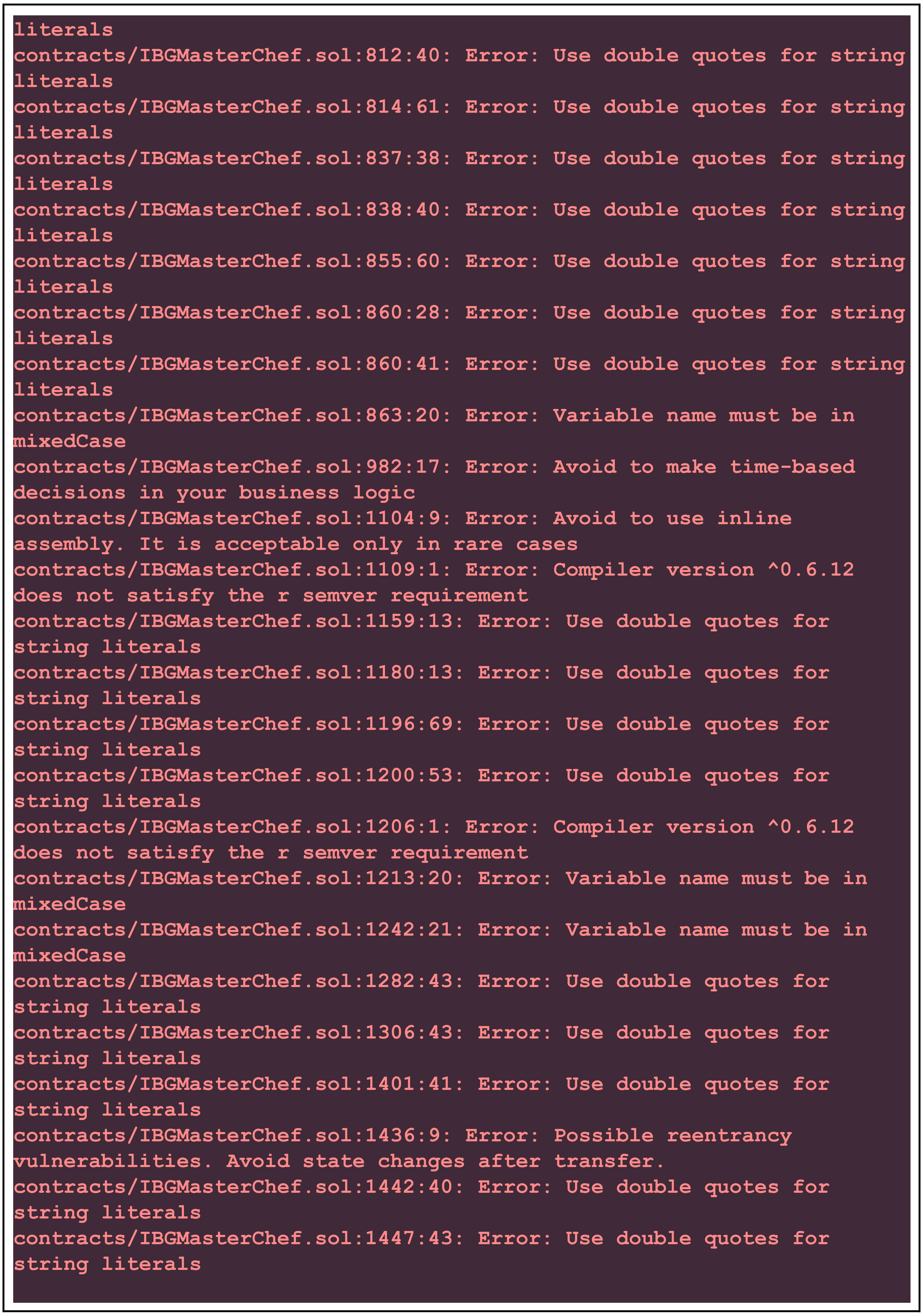
Guard conditions:

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: 225:8:

Solhint Linter

IBGETHMasterChef.sol

```
contracts/IBGMasterChef.sol:1:1: Error: Compiler version 0.6.12 does
not satisfy the r semver requirement
contracts/IBGMasterChef.sol:18:28: Error: Code contains empty blocks
contracts/IBGMasterChef.sol:68:41: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:96:41: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:225:25: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:241:26: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:284:29: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:302:26: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:342:26: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:437:50: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:440:58: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:441:26: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:463:43: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:496:59: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:511:49: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:521:37: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:587:47: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:710:59: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:750:69: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:775:39: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:776:42: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:778:59: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:793:40: Error: Use double quotes for string
```



```
literals
contracts/IBGMasterChef.sol:812:40: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:814:61: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:837:38: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:838:40: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:855:60: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:860:28: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:860:41: Error: Use double quotes for string
literals
contracts/IBGMasterChef.sol:863:20: Error: Variable name must be in
mixedCase
contracts/IBGMasterChef.sol:982:17: Error: Avoid to make time-based
decisions in your business logic
contracts/IBGMasterChef.sol:1104:9: Error: Avoid to use inline
assembly. It is acceptable only in rare cases
contracts/IBGMasterChef.sol:1109:1: Error: Compiler version ^0.6.12
does not satisfy the r semver requirement
contracts/IBGMasterChef.sol:1159:13: Error: Use double quotes for
string literals
contracts/IBGMasterChef.sol:1180:13: Error: Use double quotes for
string literals
contracts/IBGMasterChef.sol:1196:69: Error: Use double quotes for
string literals
contracts/IBGMasterChef.sol:1200:53: Error: Use double quotes for
string literals
contracts/IBGMasterChef.sol:1206:1: Error: Compiler version ^0.6.12
does not satisfy the r semver requirement
contracts/IBGMasterChef.sol:1213:20: Error: Variable name must be in
mixedCase
contracts/IBGMasterChef.sol:1242:21: Error: Variable name must be in
mixedCase
contracts/IBGMasterChef.sol:1282:43: Error: Use double quotes for
string literals
contracts/IBGMasterChef.sol:1306:43: Error: Use double quotes for
string literals
contracts/IBGMasterChef.sol:1401:41: Error: Use double quotes for
string literals
contracts/IBGMasterChef.sol:1436:9: Error: Possible reentrancy
vulnerabilities. Avoid state changes after transfer.
contracts/IBGMasterChef.sol:1442:40: Error: Use double quotes for
string literals
contracts/IBGMasterChef.sol:1447:43: Error: Use double quotes for
string literals
```

IBGStakingReserve.sol

```
contracts/IBGStakingReserve.sol:2:1: Error: Compiler version ^0.6.12
does not satisfy the r semver requirement
contracts/IBGStakingReserve.sol:19:28: Error: Code contains empty
blocks
contracts/IBGStakingReserve.sol:69:41: Error: Use double quotes for
string literals
contracts/IBGStakingReserve.sol:97:41: Error: Use double quotes for
string literals
```