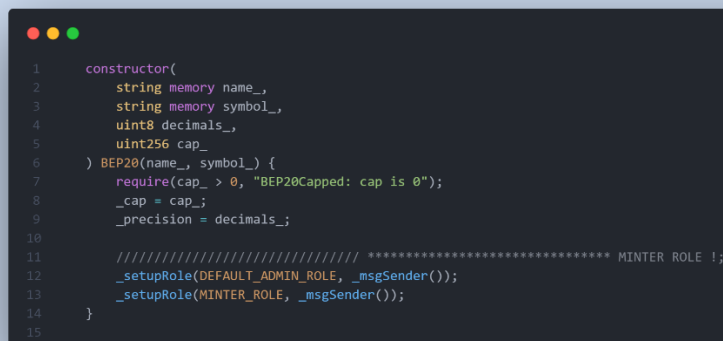


PeggedBFHT.sol

Medium severity issues

- **MINTER_ROLE / Centralization**

In the constructor function, the MINTER_ROLE / DEFAULT_ADMIN_ROLE is given to the deployer. This could be a centralization risk, only MINTER_ROLE can mint new tokens in function mint at line 982 and distribute those tokens without obtaining the consensus of the community.



```
1  constructor(  
2      string memory name_,  
3      string memory symbol_,  
4      uint8 decimals_,  
5      uint256 cap_  
6  ) BEP20(name_, symbol_) {  
7      require(cap_ > 0, "BEP20Capped: cap is 0");  
8      _cap = cap_;  
9      _precision = decimals_;  
10  
11     ////////////////////////////////// MINTER ROLE !;  
12     _setupRole(DEFAULT_ADMIN_ROLE, _msgSender());  
13     _setupRole(MINTER_ROLE, _msgSender());  
14 }  
15
```

✓ Recommendation

The risk describes the current project design and potentially makes iterations to improve the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets

Low severity issues

In function burn at line 997, we are using MINTER_ROLE in the onlyRole modifier. In this function, only MINTER_ROLE can burn his tokens! And we are using msg.sender in the body of the function.

✓ Recommendation

We can remove onlyRole modifier because we are using msg.sender in the body of the function and anyone that call this function ONLY can burn his tokens.

• Unlocked Pragma Used

Contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

```
// bad
pragma solidity ^0.4.4;

// good
pragma solidity 0.4.4;
```

✓ Recommendation

Should lock pragmas to a specific compiler version.

• Missing Emit Events

There should always be events emitted in the sensitive functions.

- function mint(address account, uint256 amount) external onlyRole(MINTER_ROLE);
- function burn(uint256 amount) external onlyRole(MINTER_ROLE);

✓ Recommendation

It is recommended emitting events for sensitive functions.