Audit Report for **PepuLock**
Date: 19 August 2025
Audit result: **High-Risk Major Flag**

Name: PepuLock

Symbol: -

Decimals: -

Network: -

Type: ETH L2 network

Owner: -

Deployer: -

Token Supply:-

Checksum: **a**fa6dc77b1f71cf281fa48605b74e722

Token Overview:

Fees: 0-100%

Fee Privilege: Owner

Ownership: Owned

Minting: No

Max Tx: No

Static Analysis

A static analysis of the code was performed using Slither. No issues were found.

N/A

Ownership Privileges:

- The owner can update the fees by more than 100%.
- The owner can update the fee wallet address.
- The validlock address can unlock and edit the lock.
- The validlock address can edit the lock description.

Findings:

Critical: 0
High: 1
Medium: 1
Low: 1
Informational & Optimizations: 1

Centralization – The owner can set high fees

Severity: High
Function: updateFee
Status: Open
Overview:
The updateFee function allows the contract owner to set arbitrarily high fees without limits or time delays. This could potentially trap user funds by making the contract economically unusable through excessive fees, posing a significant centralization risk.

```
function updateFee(uint256 newFee) external onlyOwner {
        fee = newFee;
        emit FeeUpdated(newFee);
    }
```

Suggestions:
Implement a maximum fee cap of 25% by adding a constant and require statement: require(newFee<= MAX_FEE_BPS, "Fee exceeds 25%”);

Centralization – Insufficient Validation for Fee Wallet Address(Potential Honeypot).

Severity: Medium

Status: Open

Overview:
While the function checks for a zero address, it doesn't validate if the new wallet address is a contract that may not properly handle ETH transfers. If the fee wallet is set to a contract address that cannot receive ETH, it could block all future token locking operations.

```
function updateFeeWallet(address newWallet) external onlyOwner {
        if (newWallet == address(0)) revert InvalidFeeWallet();
        feeWallet = newWallet;
        emit FeeWalletUpdated(newWallet);
    }
```

Suggestions:
Add validation for contract addresses and ensure they can receive ETH.

Centralization – Lack of Two-Step Transfer Pattern in Critical Address Updates

Severity: Low

Status: Open

Overview:
The contract implements critical address changes (like fee wallet updates) in a single step without verification. This common smart contract vulnerability risks permanent loss of funds or protocol functionality if addresses are incorrectly set through human error, typos, or compromised admin keys.

Suggestions:
Implement a two-step transfer pattern requiring explicit acceptance from new addresses before finalizing any critical address changes.

General Impact:
1- Funds could be permanently lost
2- Protocol functionality could be broken
3- No recovery mechanism available
4- Affects all future fee collections

5- Compromised admin keys could cause immediate damage
This is a common security pattern issue that affects many protocols and should be addressed using well-established two-step ownership/role transfer patterns from trusted libraries like OpenZeppelin.