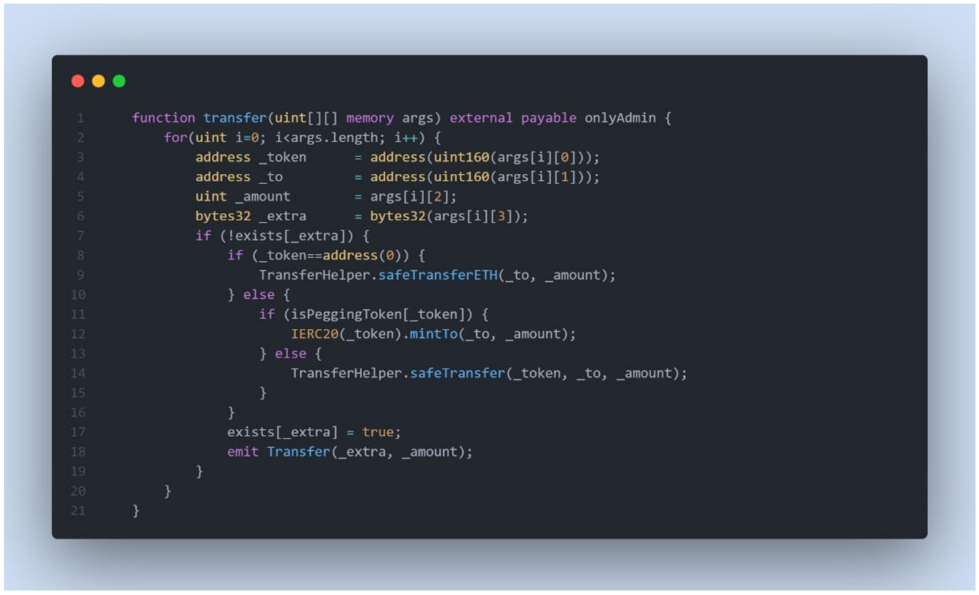
https://www.certik.com/projects/neonlink

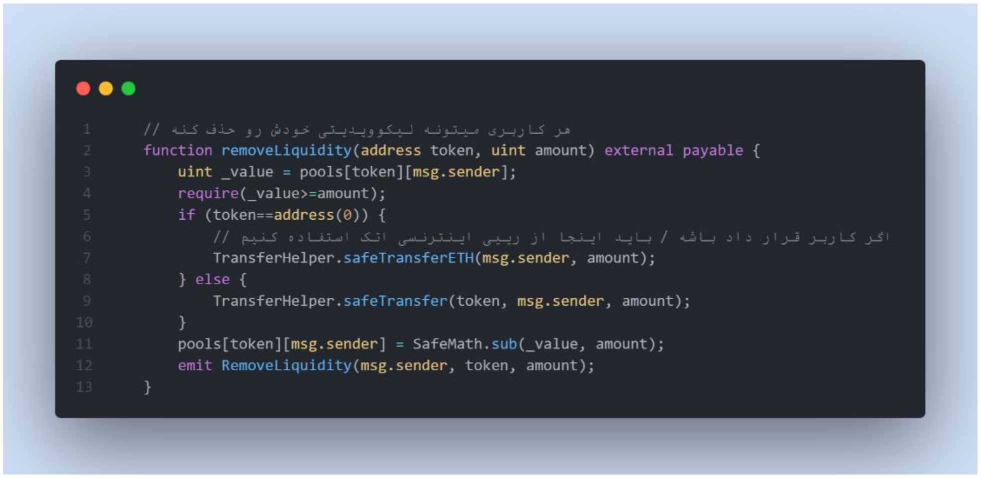
High severity issues

- ### Reentrancy Vulnerability Allows The Attacker To Drain All TheETH From The Contract.

The "removeLiquidity()" and "transfer" function transfers ETH to the user with an external call and updates the user'sbalance after the token transfer.



```
function transfer(uint[][] memory args) external payable onlyAdmin {
    for(uint i=0; i<args.length; i++) {
        address _token      = address(uint160(args[i][0]));
        address _to         = address(uint160(args[i][1]));
        uint _amount        = args[i][2];
        bytes32 _extra      = bytes32(args[i][3]);
        if (!exists[_extra]) {
            if (_token==address(0)) {
                TransferHelper.safeTransferETH(_to, _amount);
            } else {
                if (isPeggingToken[_token]) {
                    IERC20(_token).mintTo(_to, _amount);
                } else {
                    TransferHelper.safeTransfer(_token, _to, _amount);
                }
            }
            exists[_extra] = true;
            emit Transfer(_extra, _amount);
        }
    }
}
```



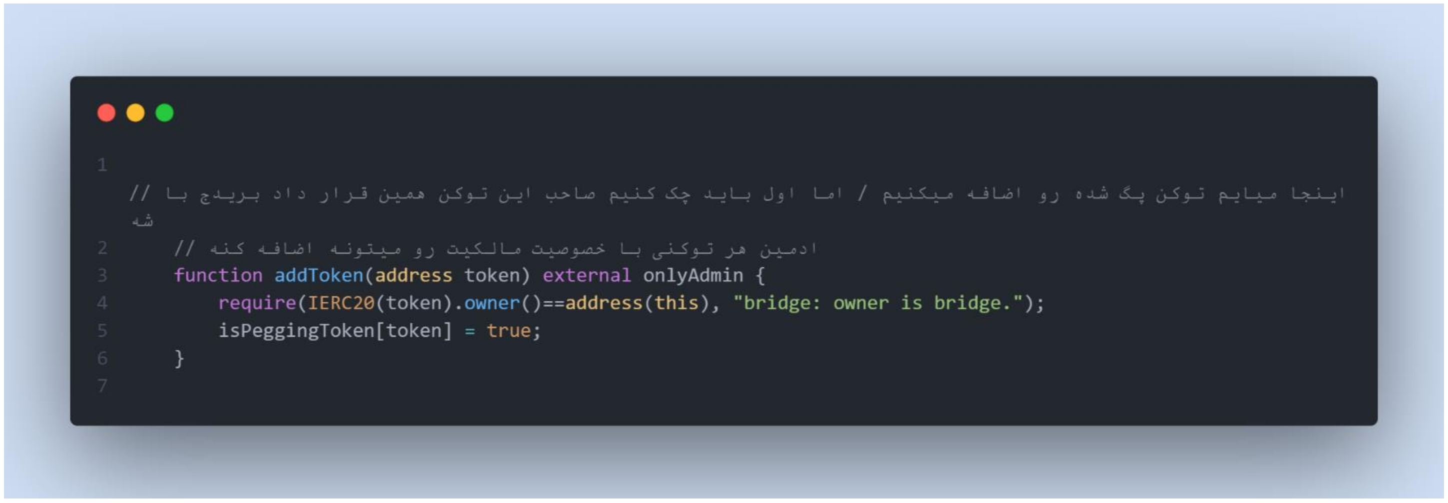
```
// هر کاربری میتونه لیکوویدیتی خودش رو حذف کنه
function removeLiquidity(address token, uint amount) external payable {
    uint _value = pools[token][msg.sender];
    require(_value>=amount);
    if (token==address(0)) {
        // اگر کاربر قرار داد باشه / باید اینجا از ریبی اینترنسی اتک استفاده کنیم
        TransferHelper.safeTransferETH(msg.sender, amount);
    } else {
        TransferHelper.safeTransfer(token, msg.sender, amount);
    }
    pools[token][msg.sender] = SafeMath.sub(_value, amount);
    emit RemoveLiquidity(msg.sender, token, amount);
}
```

Remediation

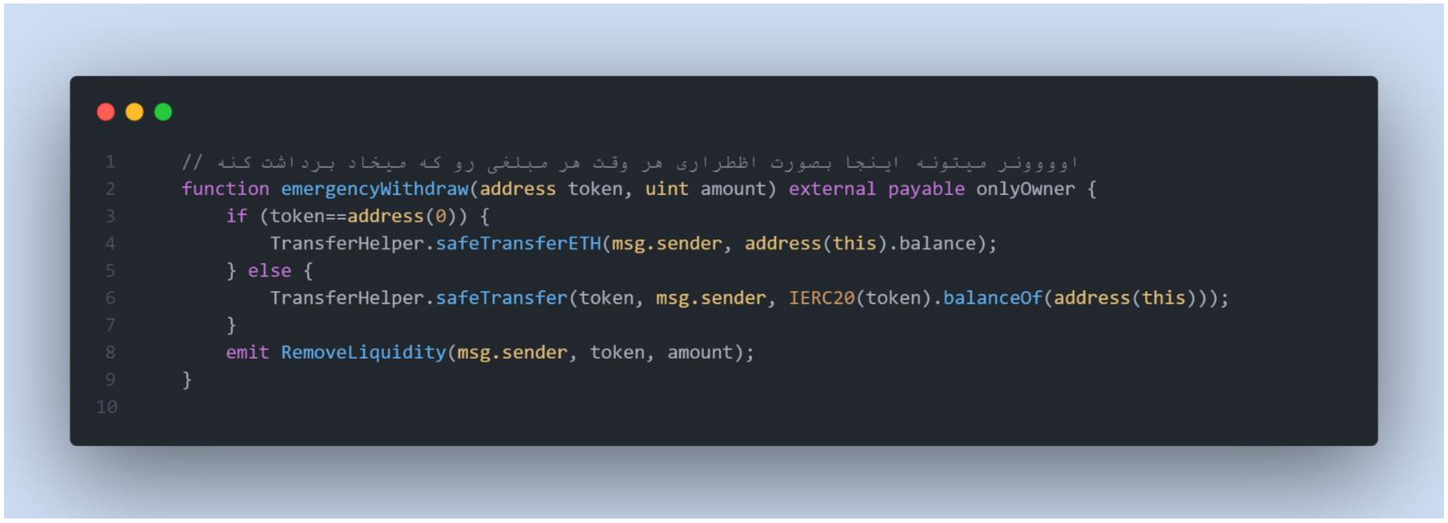
We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknowncontracts or applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for theaforementioned functions to prevent reentrancy attack.

- ## Centralization Risks

In the contract Bridge the role _owner has authority over the function emergencyWithdraw and addToken.



```
// اینجا میایم توکن پگ شده رو اضافه میکنیم / اما اول باید چک کنیم صاحب این توکن همین قرار داد بریدج با
شه
    // ادمین هر توکنی با خصوصیت مالکیت رو میتونه اضافه کنه
    function addToken(address token) external onlyAdmin {
        require(IERC20(token).owner()==address(this), "bridge: owner is bridge.");
        isPeggingToken[token] = true;
    }
```



```
    // اوووونر میتونه اینجا بصورت اظطراری هر وقت هر مبلغی رو که میخاد برداشت کنه
    function emergencyWithdraw(address token, uint amount) external payable onlyOwner {
        if (token==address(0)) {
            TransferHelper.safeTransferETH(msg.sender, address(this).balance);
        } else {
            TransferHelper.safeTransfer(token, msg.sender, IERC20(token).balanceOf(address(this)));
        }
        emit RemoveLiquidity(msg.sender, token, amount);
    }
```

Remediation

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., multisignature wallet.

Low severity issues

- ### Missing Emit Events

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Remediation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.