Audit Interim Report

This is an interim Smart Contract Audit Report that is executed for proper communication between Saif Sghaier and its clients. This is not to be considered a final report.

Project Name - *PANTHERXchange*
Project Platform - *ETH/BASE*
Project Language - *Solidity*

Project Contract Link -
0xeA2Fa80c7E2AD9265374dDF5ACcbEA134715BEe6
0x15711e03a509e320f046F4b5A8eC066ECBBaD1Ae

Project CodeBase - *<Enter CodeBase link here>*
Project Commit - *<Enter Commit Hash for codebase here>*

File Details

<Enter Name of File and Give It A File ID>
*<File ID **Naming Convention** - File ID will contain 3 letters. The first two letters will be initials from the project name, the third letter will be the initial of the file name. If two or more files contain same initial, then a fourth letter might be added to distinguish between them>*
*<File ID **Example** -*
Project name - Lightning Works
File names - LW0-Contract.sol, LW0-Minter.sol, LW0-Simple.sol
File IDs - LWC, LWM, LWS
Issues IDs- LWC01, LWC02, LWC03 etc.>

File ID	File Name
PNTHR	PNTHR-PantherXchangeCrypto.sol

Audit Details

Report Submission Date - 25/09/2024
Result - Passed

Findings Details

Severity	Number Of Issues	Percentage
Critical	0	0%
High	1	16.6%
Medium	0	0%
Low	3	50%
Informational	2	33.3%

Finding Summary

<Keep all the issues from one file together while filling out this table. Once all issues from one file are done, then move on to next.>
*<Issues **Status** Details -*
***Reported** - When Issue is first reported.*
***Acknowledged** - If client has seen the issues but not taken any action*
***Resolved** - If client has seen the issue and fixed it>*

Issue ID	Type	Line	Severity	Status
PNTHR-01	isSwapping is constant	970	High Severity	Resolved
PNTHR-02	Centralization Risk	-	Low Severity	Resolved
PNTHR-03	Missing Zero address checks	1095 - 1098 - 1101 - 1113 - 1114	Low Severity	Resolved
PNTHR-04	Missing events for critical parameters	1062 - 1067 - 1074	Low Severity	Resolved
PNTHR-05	Unused Local Variables	1153 - 1203 - 1211	Informational	Resolved
PNTHR-06	Unused Private/Internal Functions	1121-1188 / 1190-1220	Informational	Resolved

<Separate Issue Page - Copy this page for every issue and enter it's specific details. Do not write two issues in the same page>

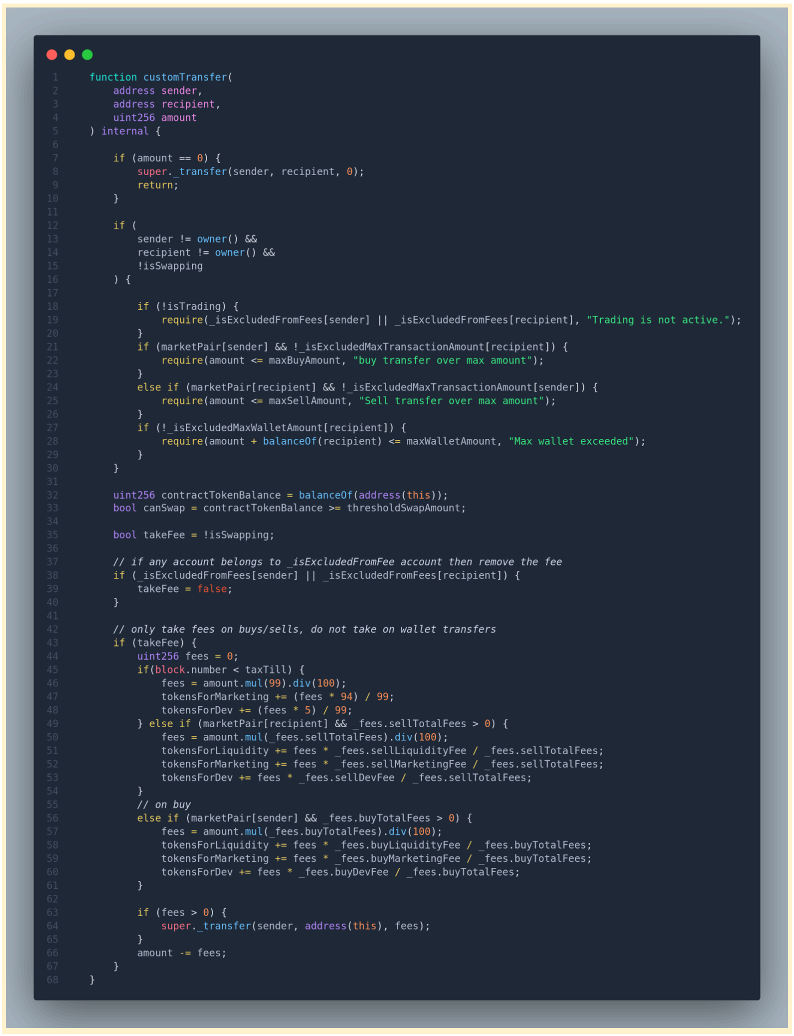
Issue ID - *PNTHER-01*

Type - isSwapping is constant
Severity - High Severity
File - PantherXchangeCrypto.sol
Line - 970
Status - Resolved

Description - isSwapping is declared as a global variable without explicit value thus it takes a default value of false. There is no way to change the value of isSwapping but it is used as a check in customTransfer(address sender, address recipient, uint256 amount) making the check always true. This issue can cause unexpected behavior.

Remediation - Add a function that changes the boolean value of isSwapping.

SnapShot -



```
function customTransfer(
    address sender,
    address recipient,
    uint256 amount
) internal {

    if (amount == 0) {
        super._transfer(sender, recipient, 0);
        return;
    }

    if (
        sender != owner() &&
        recipient != owner() &&
        !isSwapping
    ) {

        if (!isTrading) {
            require(_isExcludedFromFees[sender] || _isExcludedFromFees[recipient], "Trading is not active.");
        }
        if (marketPair[sender] && !_isExcludedMaxTransactionAmount[recipient]) {
            require(amount <= maxBuyAmount, "buy transfer over max amount");
        }
        else if (marketPair[recipient] && !_isExcludedMaxTransactionAmount[sender]) {
            require(amount <= maxSellAmount, "Sell transfer over max amount");
        }
        if (!_isExcludedMaxWalletAmount[recipient]) {
            require(amount + balanceOf(recipient) <= maxWalletAmount, "Max wallet exceeded");
        }
    }

    uint256 contractTokenBalance = balanceOf(address(this));
    bool canSwap = contractTokenBalance >= thresholdSwapAmount;

    bool takeFee = !isSwapping;

    // if any account belongs to _isExcludedFromFee account then remove the fee
    if (_isExcludedFromFees[sender] || _isExcludedFromFees[recipient]) {
        takeFee = false;
    }

    // only take fees on buys/sells, do not take on wallet transfers
    if (takeFee) {
        uint256 fees = 0;
        if(block.number < taxTill) {
            fees = amount.mul(99).div(100);
            tokensForMarketing += (fees * 94) / 99;
            tokensForDev += (fees * 5) / 99;
        } else if (marketPair[recipient] && _fees.sellTotalFees > 0) {
            fees = amount.mul(_fees.sellTotalFees).div(100);
            tokensForLiquidity += fees * _fees.sellLiquidityFee / _fees.sellTotalFees;
            tokensForMarketing += fees * _fees.sellMarketingFee / _fees.sellTotalFees;
            tokensForDev += fees * _fees.sellDevFee / _fees.sellTotalFees;
        }
        // on buy
        else if (marketPair[sender] && _fees.buyTotalFees > 0) {
            fees = amount.mul(_fees.buyTotalFees).div(100);
            tokensForLiquidity += fees * _fees.buyLiquidityFee / _fees.buyTotalFees;
            tokensForMarketing += fees * _fees.buyMarketingFee / _fees.buyTotalFees;
            tokensForDev += fees * _fees.buyDevFee / _fees.buyTotalFees;
        }

        if (fees > 0) {
            super._transfer(sender, address(this), fees);
        }
        amount -= fees;
    }
}
```

Issue ID - *PNTHER-02*

Type - Centralization risk
Severity - Low Severity
File - PantherXchangeCrypto.sol
Line - -
Status - Resolved

Description - The Owner of the contract holds all the privileges. It is considered a bad practice and can lead to loss of funds or losing control of the protocol if the owner address is compromised.

Remediation - Consider adding more roles (admins) or using a multisignature.

SnapShot -
No snapshot required.

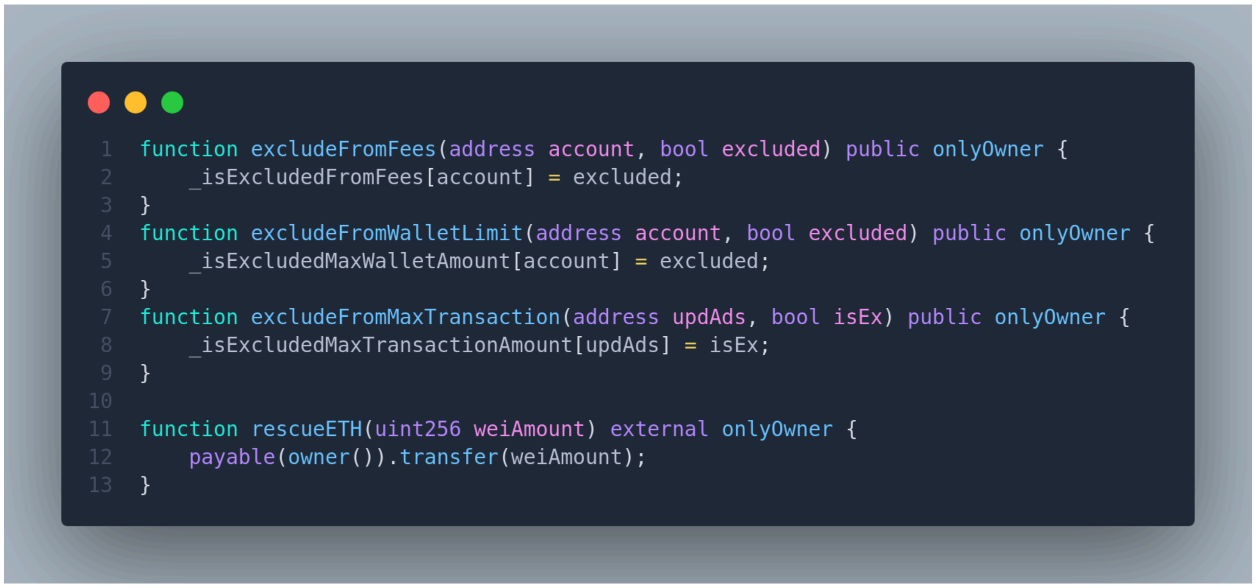
Issue ID - *PNTHER-03*

Type - Missing Zero address checks
Severity - Low Severity
File - PantherXchangeCrypto.sol
Line - 1095 - 1098 - 1101 - 1113 - 1114
Status - Resolved

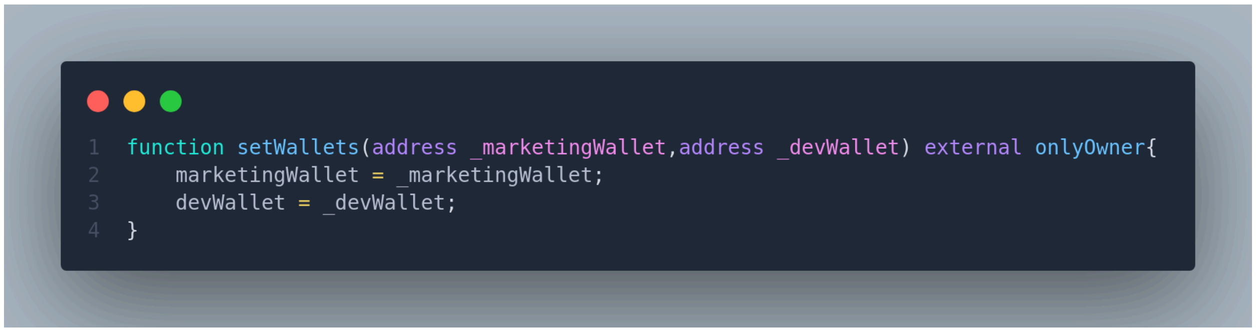
Description - The functions mentioned below do not check if the input parameters are zero addresses. Even Though the owner is the trust entity to call those functions. It is advised to add input checks.

Remediation - Add zero address checks for these functions.

SnapShot -



```
function excludeFromFees(address account, bool excluded) public onlyOwner {
    _isExcludedFromFees[account] = excluded;
}
function excludeFromWalletLimit(address account, bool excluded) public onlyOwner {
    _isExcludedMaxWalletAmount[account] = excluded;
}
function excludeFromMaxTransaction(address updAds, bool isEx) public onlyOwner {
    _isExcludedMaxTransactionAmount[updAds] = isEx;
}

function rescueETH(uint256 weiAmount) external onlyOwner {
    payable(owner()).transfer(weiAmount);
}
```



```
function setWallets(address _marketingWallet,address _devWallet) external onlyOwner{
    marketingWallet = _marketingWallet;
    devWallet = _devWallet;
}
```

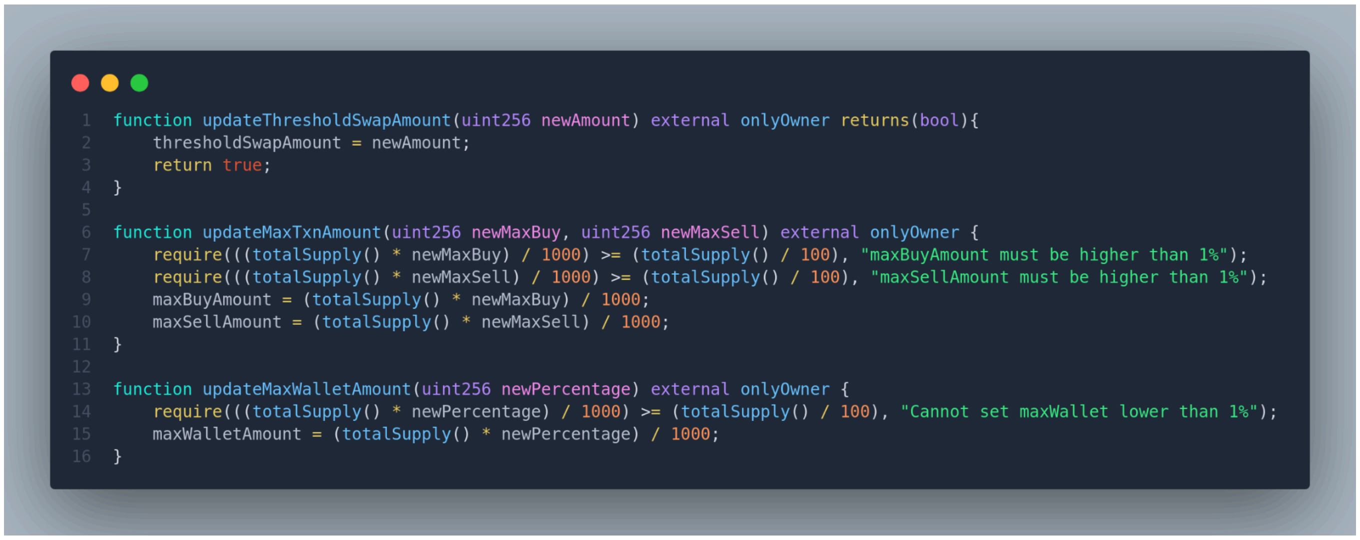
Issue ID - *PNTHER-04*

Type - Missing events for critical parameters
Severity - Low Severity
File - PantherXchangeCrypto.sol
Line - 1062 - 1067 - 1074
Status - Resolved

Description - the functions below do not emit an event so it is hard to keep track of those critical parameters.

Remediation - Emit an event for critical parameter changes.

SnapShot -



```
function updateThresholdSwapAmount(uint256 newAmount) external onlyOwner returns(bool){
    thresholdSwapAmount = newAmount;
    return true;
}

function updateMaxTxnAmount(uint256 newMaxBuy, uint256 newMaxSell) external onlyOwner {
    require(((totalSupply() * newMaxBuy) / 1000) >= (totalSupply() / 100), "maxBuyAmount must be higher than 1%");
    require(((totalSupply() * newMaxSell) / 1000) >= (totalSupply() / 100), "maxSellAmount must be higher than 1%");
    maxBuyAmount = (totalSupply() * newMaxBuy) / 1000;
    maxSellAmount = (totalSupply() * newMaxSell) / 1000;
}

function updateMaxWalletAmount(uint256 newPercentage) external onlyOwner {
    require(((totalSupply() * newPercentage) / 1000) >= (totalSupply() / 100), "Cannot set maxWallet lower than 1%");
    maxWalletAmount = (totalSupply() * newPercentage) / 1000;
}
```

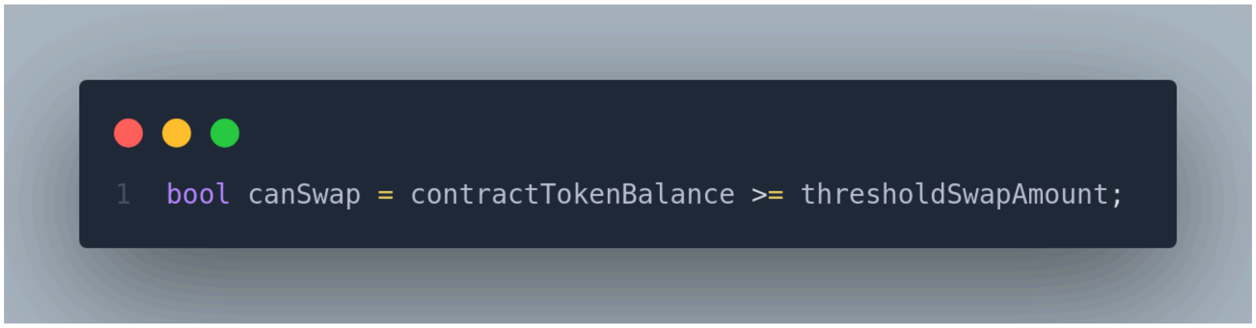
Issue ID - *PNTHER-05*

Type - Unused Local Variables
Severity - Informational
File - PantherXchangeCrypto.sol
Line - 1153 - 1203 - 1211
Status - Resolved

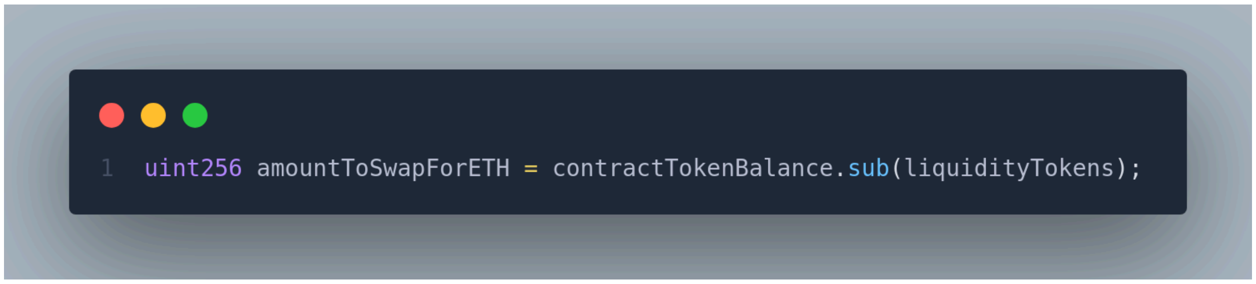
Description - The local variables below are declared but are not used.

Remediation - Remove them or use them inside the functions.

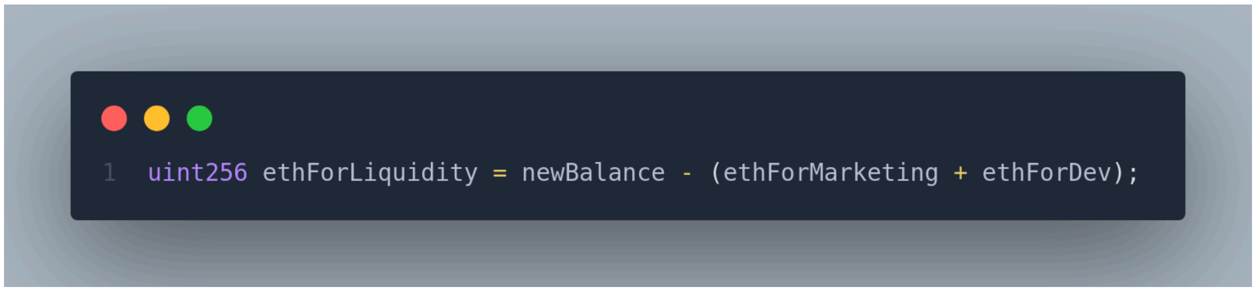
SnapShot -



```
bool canSwap = contractTokenBalance >= thresholdSwapAmount;
```



```
uint256 amountToSwapForETH = contractTokenBalance.sub(liquidityTokens);
```



```
uint256 ethForLiquidity = newBalance - (ethForMarketing + ethForDev);
```

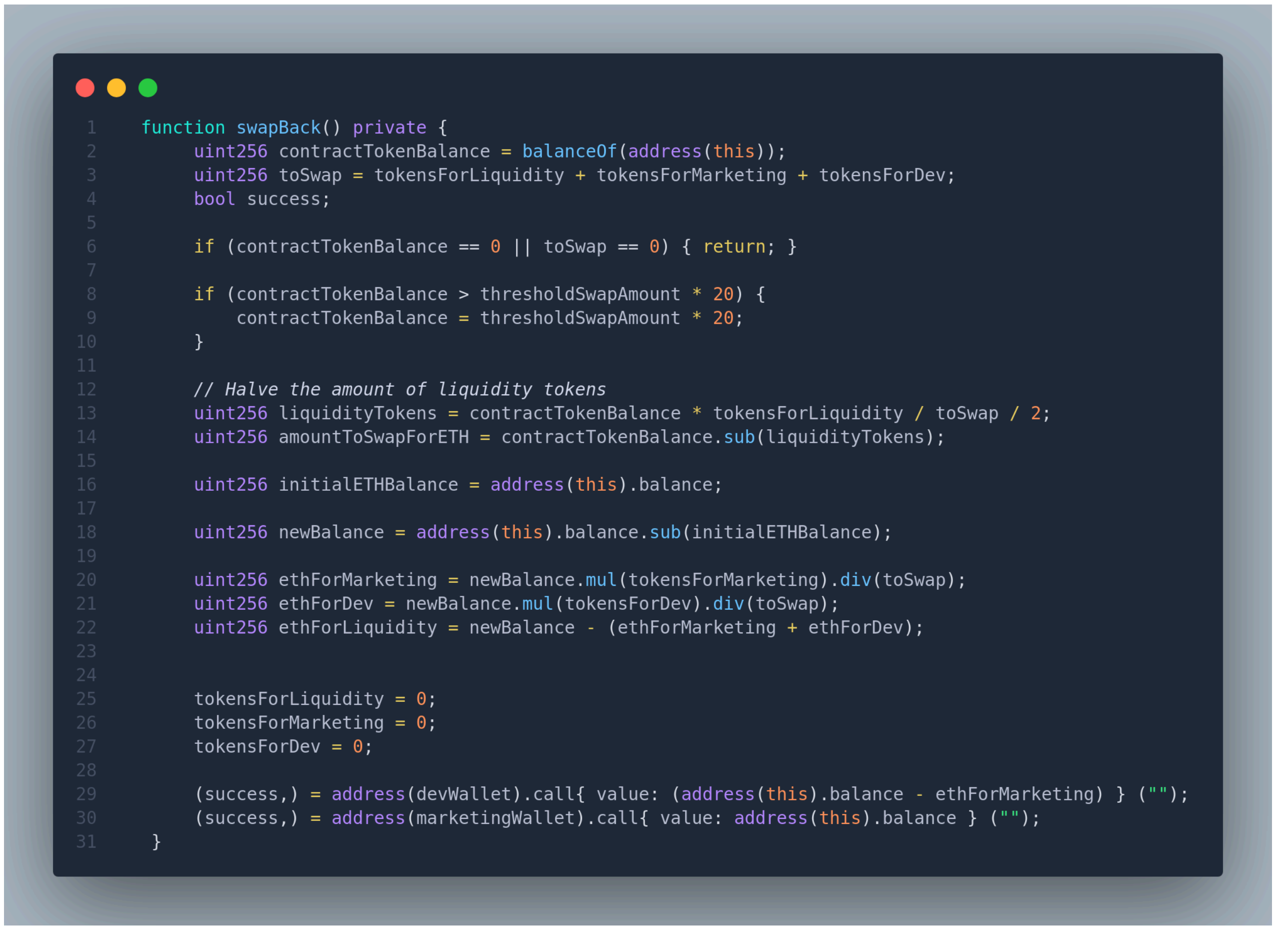
Issue ID - *PNTHER-06*

Type - Unused Private/Internal functions
Severity - Informational
File - PantherXchangeCrypto.sol
Line - 1121-1188 / 1190-1220
Status - Resolved

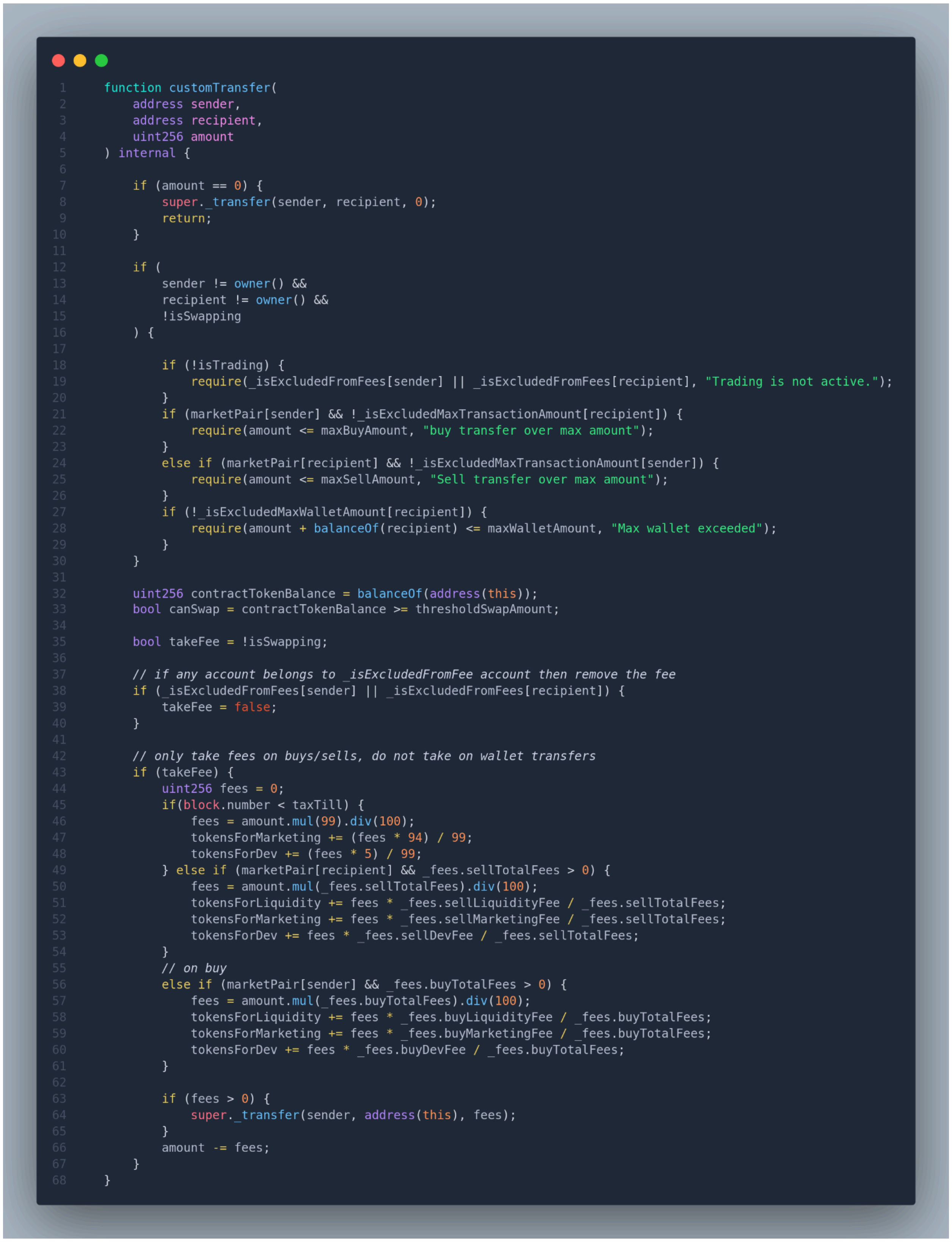
Description - The functions below are private and internal, meaning they can be accessed only within another public/external function but they are not used.

Remediation - Remove them or use them inside public/external functions.

SnapShot -



```
function swapBack() private {
     uint256 contractTokenBalance = balanceOf(address(this));
     uint256 toSwap = tokensForLiquidity + tokensForMarketing + tokensForDev;
     bool success;

     if (contractTokenBalance == 0 || toSwap == 0) { return; }

     if (contractTokenBalance > thresholdSwapAmount * 20) {
         contractTokenBalance = thresholdSwapAmount * 20;
     }

     // Halve the amount of liquidity tokens
     uint256 liquidityTokens = contractTokenBalance * tokensForLiquidity / toSwap / 2;
     uint256 amountToSwapForETH = contractTokenBalance.sub(liquidityTokens);

     uint256 initialETHBalance = address(this).balance;

     uint256 newBalance = address(this).balance.sub(initialETHBalance);

     uint256 ethForMarketing = newBalance.mul(tokensForMarketing).div(toSwap);
     uint256 ethForDev = newBalance.mul(tokensForDev).div(toSwap);
     uint256 ethForLiquidity = newBalance - (ethForMarketing + ethForDev);


     tokensForLiquidity = 0;
     tokensForMarketing = 0;
     tokensForDev = 0;

     (success,) = address(devWallet).call{ value: (address(this).balance - ethForMarketing) } ("");
     (success,) = address(marketingWallet).call{ value: address(this).balance } ("");
 }
```



```
function customTransfer(
    address sender,
    address recipient,
    uint256 amount
) internal {

    if (amount == 0) {
        super._transfer(sender, recipient, 0);
        return;
    }

    if (
        sender != owner() &&
        recipient != owner() &&
        !isSwapping
    ) {

        if (!isTrading) {
            require(_isExcludedFromFees[sender] || _isExcludedFromFees[recipient], "Trading is not active.");
        }
        if (marketPair[sender] && !_isExcludedMaxTransactionAmount[recipient]) {
            require(amount <= maxBuyAmount, "buy transfer over max amount");
        }
        else if (marketPair[recipient] && !_isExcludedMaxTransactionAmount[sender]) {
            require(amount <= maxSellAmount, "Sell transfer over max amount");
        }
        if (!_isExcludedMaxWalletAmount[recipient]) {
            require(amount + balanceOf(recipient) <= maxWalletAmount, "Max wallet exceeded");
        }
    }

    uint256 contractTokenBalance = balanceOf(address(this));
    bool canSwap = contractTokenBalance >= thresholdSwapAmount;

    bool takeFee = !isSwapping;

    // if any account belongs to _isExcludedFromFee account then remove the fee
    if (_isExcludedFromFees[sender] || _isExcludedFromFees[recipient]) {
        takeFee = false;
    }

    // only take fees on buys/sells, do not take on wallet transfers
    if (takeFee) {
        uint256 fees = 0;
        if(block.number < taxTill) {
            fees = amount.mul(99).div(100);
            tokensForMarketing += (fees * 94) / 99;
            tokensForDev += (fees * 5) / 99;
        } else if (marketPair[recipient] && _fees.sellTotalFees > 0) {
            fees = amount.mul(_fees.sellTotalFees).div(100);
            tokensForLiquidity += fees * _fees.sellLiquidityFee / _fees.sellTotalFees;
            tokensForMarketing += fees * _fees.sellMarketingFee / _fees.sellTotalFees;
            tokensForDev += fees * _fees.sellDevFee / _fees.sellTotalFees;
        }
        // on buy
        else if (marketPair[sender] && _fees.buyTotalFees > 0) {
            fees = amount.mul(_fees.buyTotalFees).div(100);
            tokensForLiquidity += fees * _fees.buyLiquidityFee / _fees.buyTotalFees;
            tokensForMarketing += fees * _fees.buyMarketingFee / _fees.buyTotalFees;
            tokensForDev += fees * _fees.buyDevFee / _fees.buyTotalFees;
        }

        if (fees > 0) {
            super._transfer(sender, address(this), fees);
        }
        amount -= fees;
    }
}
```