SPYWOLF
Security Audit Report

Audit prepared for

Rhino

Completed on

May 28, 2025

OVERVIEW

This goal of this report is to review the main aspects of the project to help investors make an informative decision during their research process.

You will find a a summarized review of the following key points:

- ✓ Contract's source code
- ✓ Owners' wallets
- ✓ Tokenomics
- ✓ Team transparency and goals
- ✓ Website's age, code, security and UX
- ✓ Whitepaper and roadmap
- ✓ Social media & online presence

The results of this audit are purely based on the team's evaluation and does not guarantee nor reflect the projects outcome and goal

- SPYWOLF Team -

TABLE OF CONTENTS

RHINOFI

PROJECT DESCRIPTION:

RhinoFi is a fee-based, hyper-deflationary reflection token protocol designed to reward long-term holders and liquidity providers while creating a sustainable ecosystem of value and utility. By leveraging an innovative dynamic sell tax structure, multi-asset reflections, and strategic liquidity incentives, RhinoFi ensures that users (Rhinos) benefit simply by holding RHINO tokens in their wallets.

Release Date: TBA

Launchpad: TBA

Category: DeFi/Dividend

CONTRACT INFO

Token Name	Symbol
RHINO	RHINO
Contract Address	
Not deployed	
Network	Language
Not deployed	Solidity
Deployment Date	Contract Type
Not deployed	Dividend Token
Total Supply	Decimals
369,000,000	18

TAXES

Buy Tax
6%

Sell Tax
12%

*Taxes can be changed in future

Our Contract Review Process

The contract review process pays special attention to the following:

- ✓ Testing the smart contracts against both common and uncommon vulnerabilities
- ✓ Assessing the codebase to ensure compliance with current best practices and industry standards.
- ✓ Ensuring contract logic meets the specifications and intentions of the client.
- ✓ Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- ✓ Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

SMART CONTRACT STATS

Calls Count	Not available
External calls	Not available
Internal calls	Not available
Transactions count	Not available
Last transaction time	Not available
Deployment Date	Not available
Create TX	Not available
Owner	Not available
Deployer	Not available

TOKEN TRANSFERS STATS

Transfer Count	Not available
Total Amount	Not available
Median Transfer Amount	Not available
Average Transfer Amount	Not available
First transfer date	Not available
Last transfer date	Not available
Days token transferred	Not available

VULNERABILITY ANALYSIS

ID	Title	
SWC-100	Function Default Visibility	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Reentrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegatecall to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed

VULNERABILITY ANALYSIS

ID	Title	
SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed

MANUAL CODE REVIEW

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time.

We categorize these vulnerabilities by 4 different threat levels.

THREAT LEVELS

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance, functionality and should be fixed before moving to a live environment.

Low Risk

Issues on this level are minor details and warning that can remain unfixed.

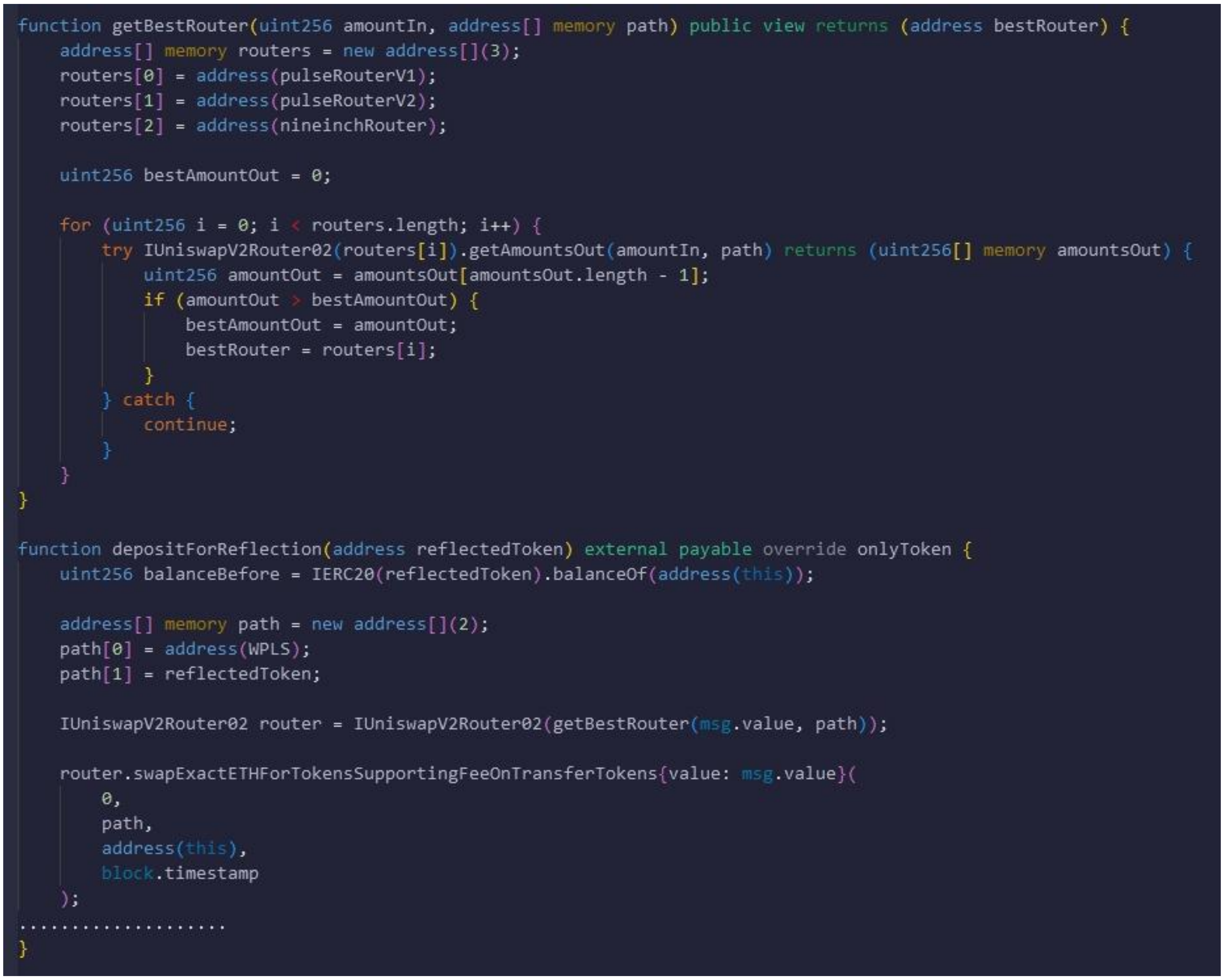
Informational

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.

FOUND THREATS

⚠ Low Risk

If getBestRouter() unexpectedly returns address(0), subsequent invocations of depositForReflection() will result in a failure.

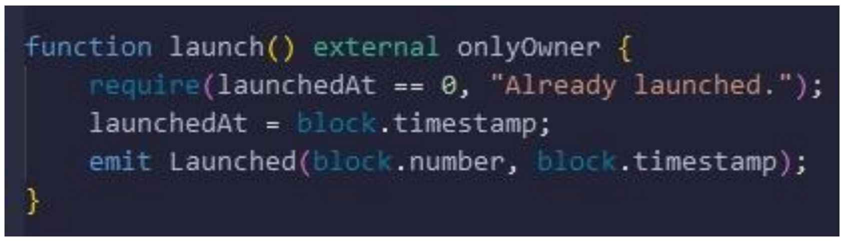


```
function getBestRouter(uint256 amountIn, address[] memory path) public view returns (address bestRouter) {
    address[] memory routers = new address[](3);
    routers[0] = address(pulseRouterV1);
    routers[1] = address(pulseRouterV2);
    routers[2] = address(nineinchRouter);

    uint256 bestAmountOut = 0;

    for (uint256 i = 0; i < routers.length; i++) {
        try IUniswapV2Router02(routers[i]).getAmountsOut(amountIn, path) returns (uint256[] memory amountsOut) {
            uint256 amountOut = amountsOut[amountsOut.length - 1];
            if (amountOut > bestAmountOut) {
                bestAmountOut = amountOut;
                bestRouter = routers[i];
            }
        } catch {
            continue;
        }
    }
}

function depositForReflection(address reflectedToken) external payable override onlyToken {
    uint256 balanceBefore = IERC20(reflectedToken).balanceOf(address(this));

    address[] memory path = new address[](2);
    path[0] = address(WPLS);
    path[1] = reflectedToken;

    IUniswapV2Router02 router = IUniswapV2Router02(getBestRouter(msg.value, path));

    router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: msg.value}(
        0,
        path,
        address(this),
        block.timestamp
    );
....................
}
```

FOUND THREATS

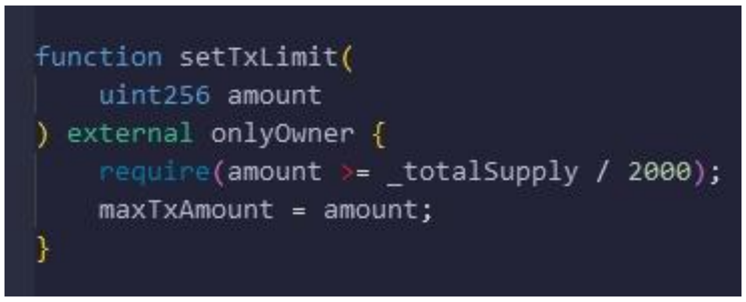
Informational

- The contract owner has the ability to enable trading; however, once activated, trading cannot be disabled.



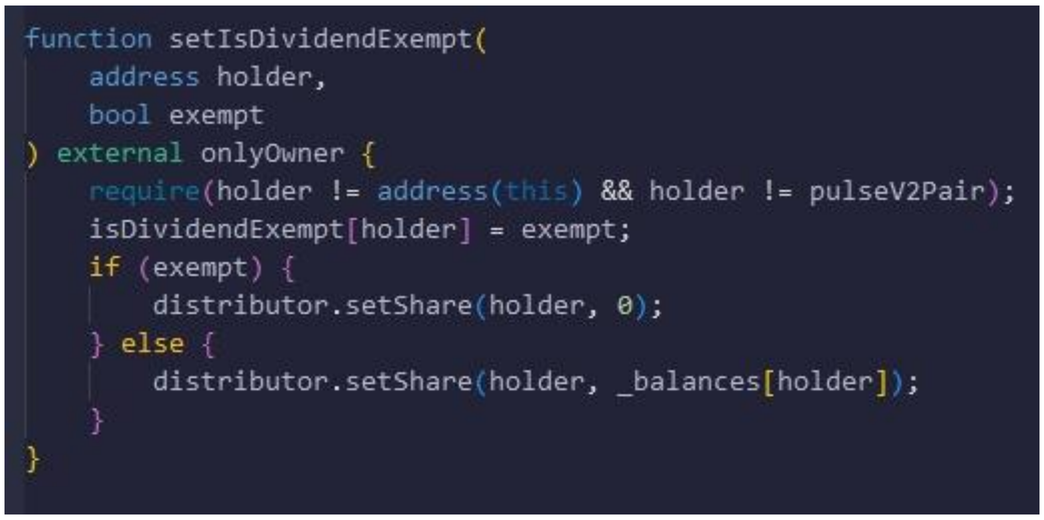
```
function launch() external onlyOwner {
    require(launchedAt == 0, "Already launched.");
    launchedAt = block.timestamp;
    emit Launched(block.number, block.timestamp);
}
```

- The contract owner can set max transaction limit but cannot lower it than 0.05% of total supply.



```
function setTxLimit(
    uint256 amount
) external onlyOwner {
    require(amount >= _totalSupply / 2000);
    maxTxAmount = amount;
}
```

- The contract ownercan exclude addresses from dividends.



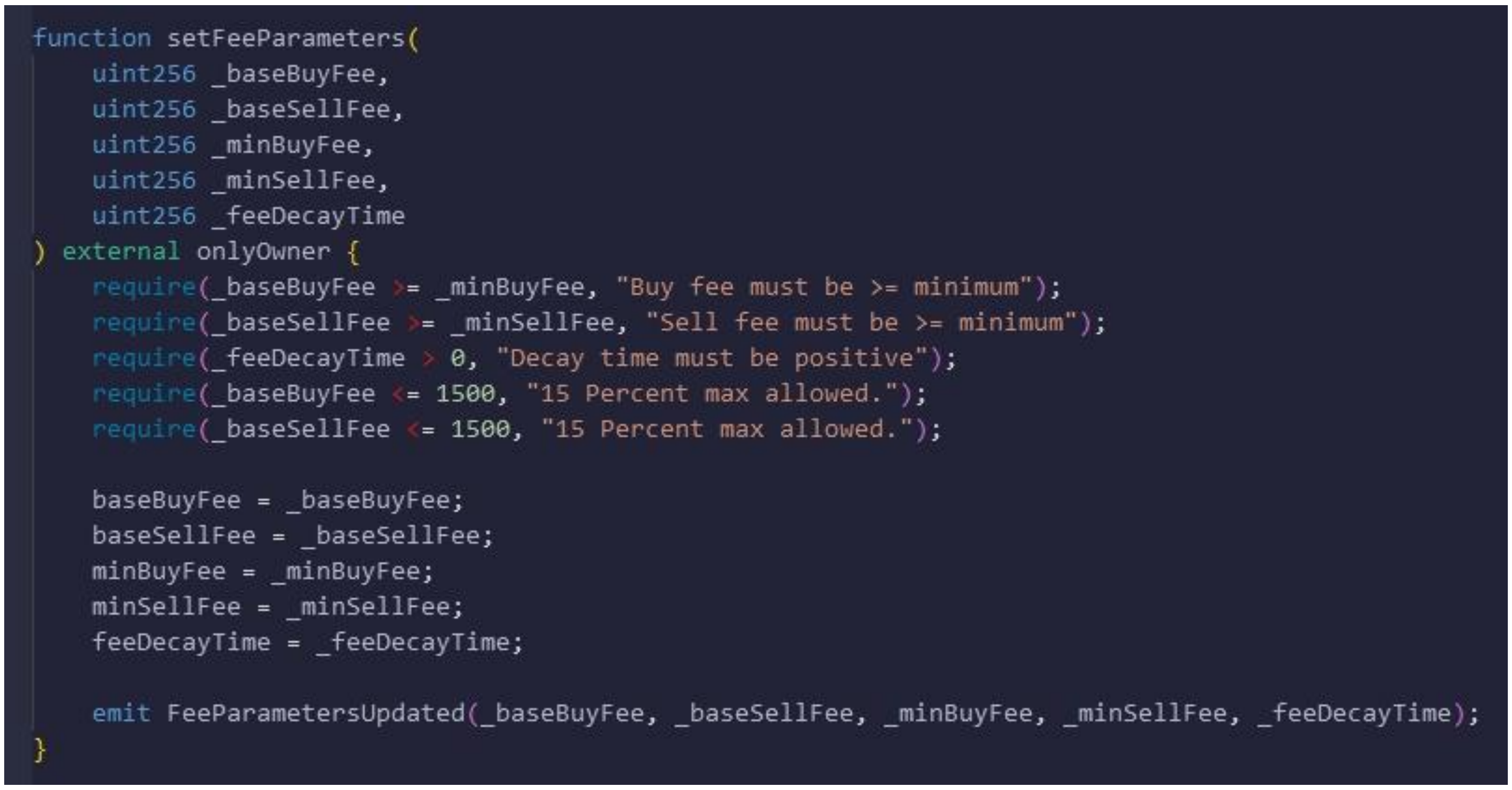
```
function setIsDividendExempt(
    address holder,
    bool exempt
) external onlyOwner {
    require(holder != address(this) && holder != pulseV2Pair);
    isDividendExempt[holder] = exempt;
    if (exempt) {
        distributor.setShare(holder, 0);
    } else {
        distributor.setShare(holder, _balances[holder]);
    }
}
```

FOUND THREATS

Informational

- The contract owner has the authority to adjust buy, sell, and transfer fees, up to a maximum of 15%.

When fees are greater than 0, a percentage of tokens will be deducted from each transaction initiated by users. The deducted amount corresponds to the applicable fee percentage of the total tokens bought, sold, or transferred.

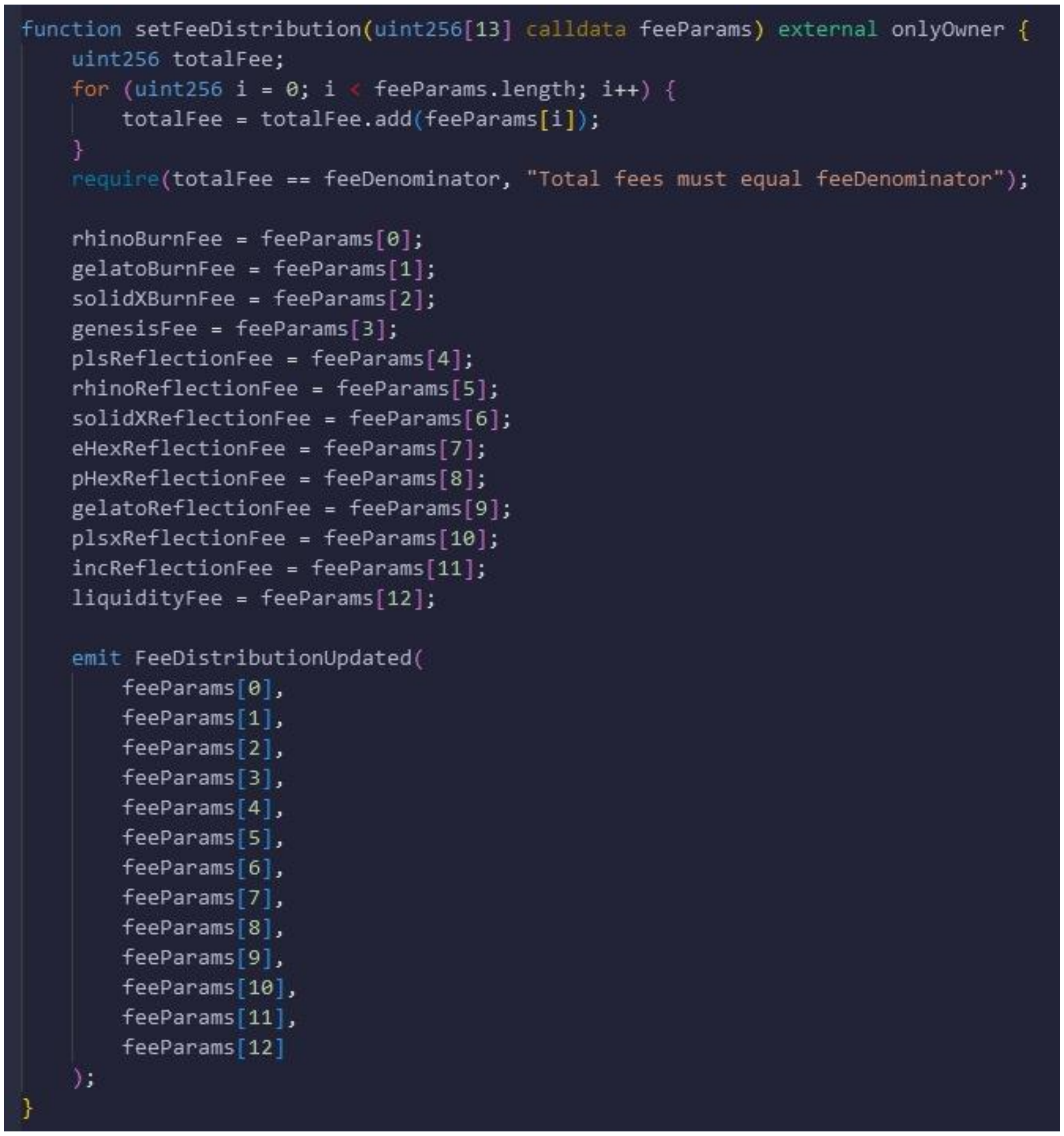


```
function setFeeParameters(
    uint256 _baseBuyFee,
    uint256 _baseSellFee,
    uint256 _minBuyFee,
    uint256 _minSellFee,
    uint256 _feeDecayTime
) external onlyOwner {
    require(_baseBuyFee >= _minBuyFee, "Buy fee must be >= minimum");
    require(_baseSellFee >= _minSellFee, "Sell fee must be >= minimum");
    require(_feeDecayTime > 0, "Decay time must be positive");
    require(_baseBuyFee <= 1500, "15 Percent max allowed.");
    require(_baseSellFee <= 1500, "15 Percent max allowed.");

    baseBuyFee = _baseBuyFee;
    baseSellFee = _baseSellFee;
    minBuyFee = _minBuyFee;
    minSellFee = _minSellFee;
    feeDecayTime = _feeDecayTime;

    emit FeeParametersUpdated(_baseBuyFee, _baseSellFee, _minBuyFee, _minSellFee, _feeDecayTime);
}
```

FOUND THREATS

Informational

- The contract owner has the authority to configure the allocation of fee distributions.

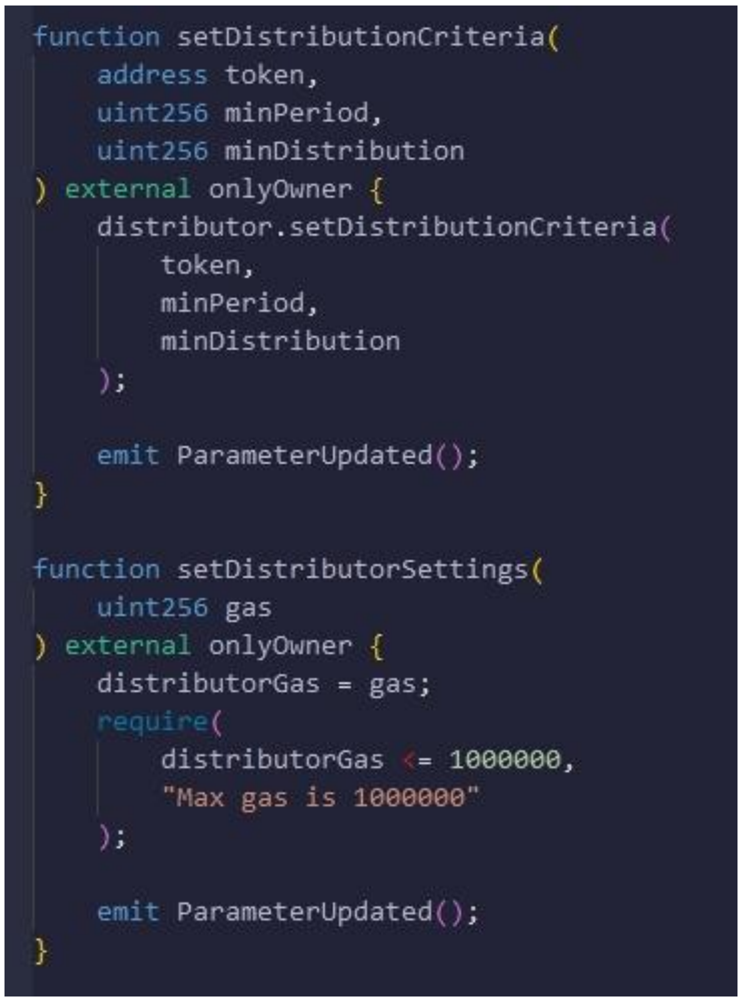


```
function setFeeDistribution(uint256[13] calldata feeParams) external onlyOwner {
    uint256 totalFee;
    for (uint256 i = 0; i < feeParams.length; i++) {
        totalFee = totalFee.add(feeParams[i]);
    }
    require(totalFee == feeDenominator, "Total fees must equal feeDenominator");

    rhinoBurnFee = feeParams[0];
    gelatoBurnFee = feeParams[1];
    solidXBurnFee = feeParams[2];
    genesisFee = feeParams[3];
    plsReflectionFee = feeParams[4];
    rhinoReflectionFee = feeParams[5];
    solidXReflectionFee = feeParams[6];
    eHexReflectionFee = feeParams[7];
    pHexReflectionFee = feeParams[8];
    gelatoReflectionFee = feeParams[9];
    plsxReflectionFee = feeParams[10];
    incReflectionFee = feeParams[11];
    liquidityFee = feeParams[12];

    emit FeeDistributionUpdated(
        feeParams[0],
        feeParams[1],
        feeParams[2],
        feeParams[3],
        feeParams[4],
        feeParams[5],
        feeParams[6],
        feeParams[7],
        feeParams[8],
        feeParams[9],
        feeParams[10],
        feeParams[11],
        feeParams[12]
    );
}
```

FOUND THREATS

Informational

- The contract owner has the authority to modify the reward distribution criteria for the dividend distributor.

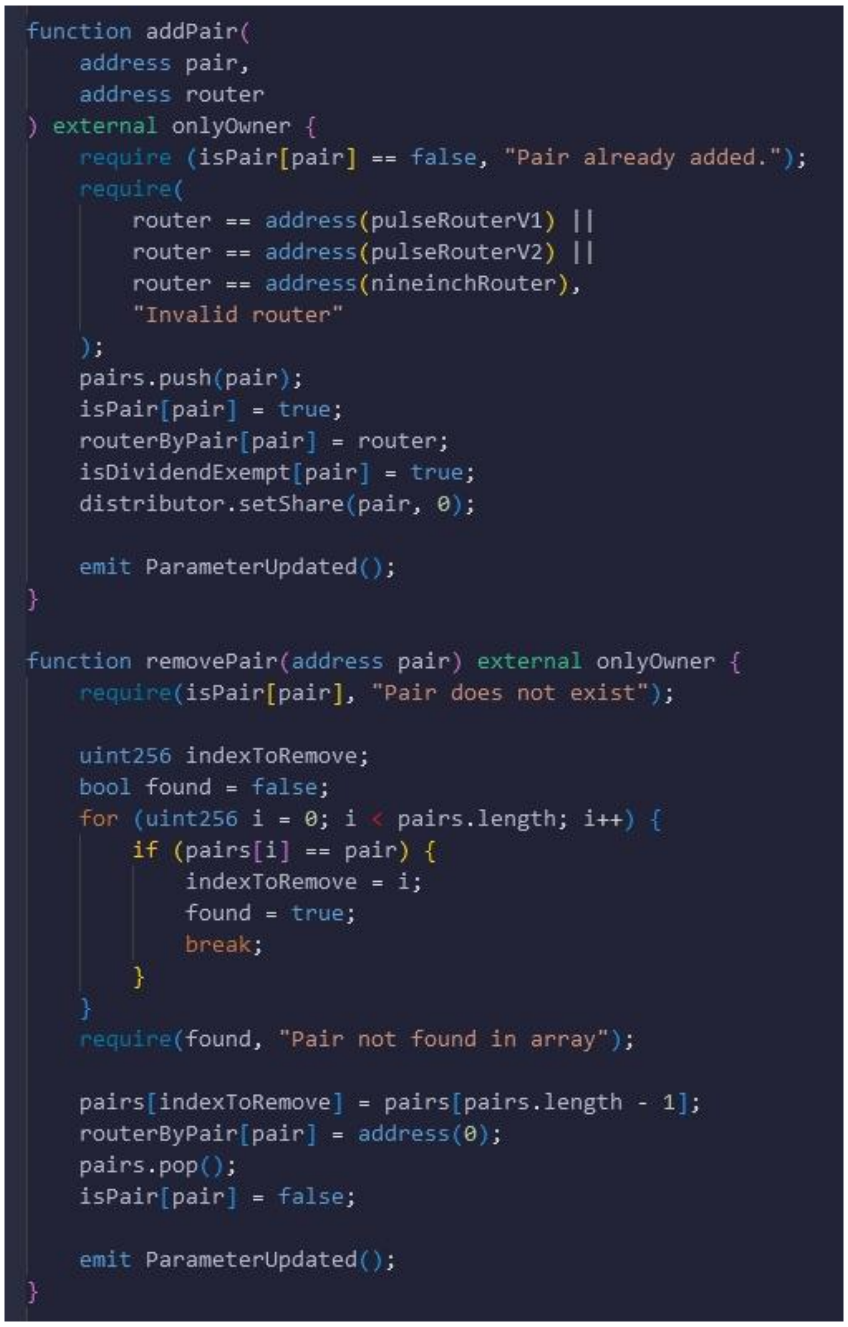


```
function setDistributionCriteria(
    address token,
    uint256 minPeriod,
    uint256 minDistribution
) external onlyOwner {
    distributor.setDistributionCriteria(
        token,
        minPeriod,
        minDistribution
    );

    emit ParameterUpdated();
}

function setDistributorSettings(
    uint256 gas
) external onlyOwner {
    distributorGas = gas;
    require(
        distributorGas <= 1000000,
        "Max gas is 1000000"
    );

    emit ParameterUpdated();
}
```

FOUND THREATS

Informational

- The contract owner has the ability to add or remove addresses from the pairs array, designating them as liquidity pairs.

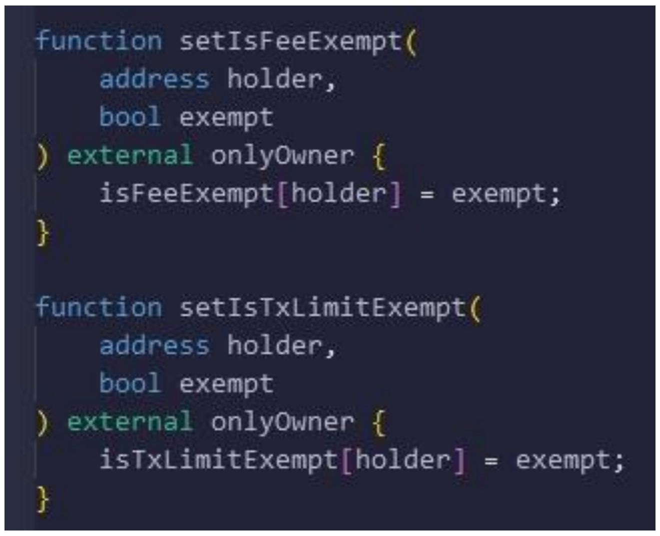


```
function addPair(
    address pair,
    address router
) external onlyOwner {
    require (isPair[pair] == false, "Pair already added.");
    require(
        router == address(pulseRouterV1) ||
        router == address(pulseRouterV2) ||
        router == address(nineinchRouter),
        "Invalid router"
    );
    pairs.push(pair);
    isPair[pair] = true;
    routerByPair[pair] = router;
    isDividendExempt[pair] = true;
    distributor.setShare(pair, 0);

    emit ParameterUpdated();
}

function removePair(address pair) external onlyOwner {
    require(isPair[pair], "Pair does not exist");

    uint256 indexToRemove;
    bool found = false;
    for (uint256 i = 0; i < pairs.length; i++) {
        if (pairs[i] == pair) {
            indexToRemove = i;
            found = true;
            break;
        }
    }
    require(found, "Pair not found in array");

    pairs[indexToRemove] = pairs[pairs.length - 1];
    routerByPair[pair] = address(0);
    pairs.pop();
    isPair[pair] = false;

    emit ParameterUpdated();
}
```

FOUND THREATS

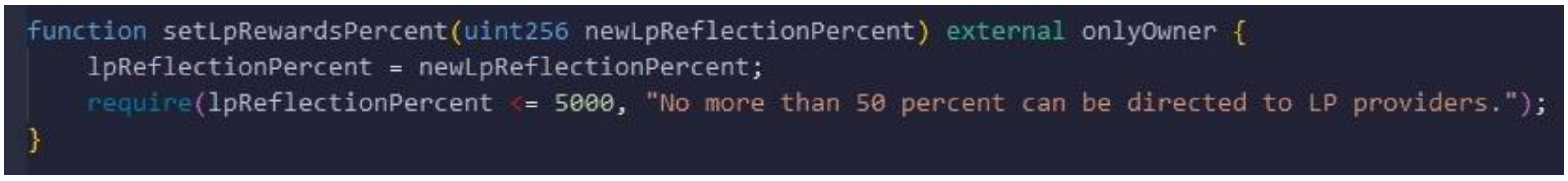
Informational

- The contract owner can exclude addresses from fees and max transaction limit.



```
function setIsFeeExempt(
    address holder,
    bool exempt
) external onlyOwner {
    isFeeExempt[holder] = exempt;
}

function setIsTxLimitExempt(
    address holder,
    bool exempt
) external onlyOwner {
    isTxLimitExempt[holder] = exempt;
}
```

- The contract owner has the ability to configure LP reflection reward percentage.



```
function setLpRewardsPercent(uint256 newLpReflectionPercent) external onlyOwner {
    lpReflectionPercent = newLpReflectionPercent;
    require(lpReflectionPercent <= 5000, "No more than 50 percent can be directed to LP providers.");
}
```

TOKENOMICS

No information available about the initial tokens distribution at the time of the audit.

SPYWOLF

CRYPTO SECURITY

Audits | KYCs | dApps
Contract Development

ABOUT US

We are a growing crypto security agency offering audits, KYCs and consulting services for some of the top names in the crypto industry.

- ✓ OVER 700 SUCCESSFUL CLIENTS
- ✓ MORE THAN 1000 SCAMS EXPOSED
- ✓ MILLIONS SAVED IN POTENTIAL FRAUD
- ✓ PARTNERSHIPS WITH TOP LAUNCHPADS, INFLUENCERS AND CRYPTO PROJECTS
- ✓ CONSTANTLY BUILDING TOOLS TO HELP INVESTORS DO BETTER RESEARCH

To hire us, reach out to
contact@spywolf.co or
t.me/joe_SpyWolf

FIND US ONLINE

SPYWOLF.CO

@SPYWOLFNETWORK

@SPYWOLFNETWORK

Disclaimer

This report shows findings based on our limited project analysis, following good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall social media and website presence and team transparency details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report.
While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER:
By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.
No one shall have any right to rely on the report or its contents, and SpyWolf and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (SpyWolf) owe no duty of care towards you or any other person, nor does SpyWolf make any warranty or representation to any person on the accuracy or completeness of the report.
The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SpyWolf hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SpyWolf hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SpyWolf, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts, website, social media and team.
No applications were reviewed for security. No product code has been reviewed.