
PulseFun NEON Security Review

Introduction

A time-boxed security review of the **NEON Token Ecosystem** contracts was conducted by **Stonewall**, with a focus on the security aspects of the smart contract implementation.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

About Stonewall

Stonewall is an independent smart contract security firm delivering immovable protection for Web3 protocols. Our team brings deep expertise in DeFi security, having reviewed DEXs, yield farming protocols, gaming contracts, and complex financial systems.

About NEON

NEON (NEXION) is a deflationary token on PulseChain with integrated DeFi features:

- **NEON Token:** ERC20 with transfer taxes (burn + LP rewards)
- **NEONStaking:** Single-sided staking with dual rewards (NEON + INC)
- **NEONAuction:** Token emission via auction mechanism
- **NEONVault:** Vault for auction proceeds
- **NEONBuynBurn:** Automated buyback and burn

Privileged Roles & Actors

Role	Description
Owner	Full control over fees, exclusions, staking address
Auction Contract	Can mint NEON tokens for auctions and LP
LPStaking Contract	Receives fee distributions

Observations

- Token has 6% total transfer tax (2% burn, 2% PLS rewards, 2% NEON rewards)
- Creates pairs with HEX, PLSX, INC, PDAI, ATROPA on deployment
- Automatic fee swapping when threshold reached
- Drip-based staking reward distribution

Risk Classification

	High Impact	Medium Impact	Low Impact
High Likelihood	Critical	High	Medium
Medium Likelihood	High	Medium	Low
Low Likelihood	Medium	Low	Low

Security Assessment Summary

Review Details	
Protocol Name	NEON Token Ecosystem
Repository	Private
Commit	0055958b44f63163dabf0acc419472d2832cbee0
Review Date	January 2026

Methods	Manual review, static analysis
Network	PulseChain

Project Links

Platform	Link
Twitter	@nexionpulse
Telegram	NexionPulse
DexScreener	NEON/PLS

Scope

Contract	SLOC
<code>NEON.sol</code>	~400
<code>NEONStaking.sol</code>	~330
<code>NEONVault.sol</code>	~150
<code>NEONBuynBurn.sol</code>	~100
<code>NEONAuction.sol</code>	~250

Findings Summary

ID	Title	Severity	Status
[H-01]	removeLiquidity temporarily grants fee exclusion to any user	High	Open
[M-01]	Division by zero possible in staking when totalStaked is zero	Medium	Open
[M-02]	Staking drip rate allows 0% which stops reward distribution	Medium	Open
[L-01]	Initialize function lacks access control after first call	Low	Open
[L-02]	Hardcoded pair addresses may fail on deployment	Low	Open

[L-03]	No slippage protection on liquidity operations	Low	Open
--------	--	-----	------

Findings

[H-01] removeLiquidity temporarily grants fee exclusion to any user

Severity: High

Impact: High - Any user can avoid transfer fees during liquidity removal.

Likelihood: High - Easy to exploit by calling removeLiquidity.

Location: `NEON.sol:266-293`

Description:

The `removeLiquidity` function temporarily sets fee exclusion for the caller:

```
function removeLiquidity(address pair, uint256 liquidity) external {
    require(isLiquidityPair[pair], "Liquidity pair is not correct");
    // ...

    bool wasExcluded = isExcludedFromFee[msg.sender];
    if (!wasExcluded) {
        isExcludedFromFee[msg.sender] = true; // @audit Anyone gets exclusion
    }

    pulseXRouter.removeLiquidity(...);

    if (!wasExcluded) {
        isExcludedFromFee[msg.sender] = false;
    }
}
```

While the exclusion is meant to be temporary for the liquidity removal, a malicious actor can use reentrancy (via the router callback) or simply front-run with a transfer during the exclusion window.

Attack Scenario:

1. Attacker has NEON tokens to sell
2. Attacker provides minimal liquidity to get LP tokens
3. Attacker calls `removeLiquidity()` which sets their exclusion to true

4. During the router callback or via flash loan, attacker transfers large NEON amount fee-free
5. Exclusion is removed but attacker avoided 6% tax

Recommendation:

Use a different mechanism that doesn't expose exclusion status:

```
// Add a flag for internal operations
bool private _internalOperation;

modifier noFeeForInternal() {
    _internalOperation = true;
    _;
    _internalOperation = false;
}

function _transfer(...) internal override {
    bool takeFee = !isExcludedFromFee[sender] &&
                  !isExcludedFromFee[recipient] &&
                  !_internalOperation;

    // ...
}
```

[M-01] Division by zero possible in staking when totalStaked is zero

Severity: Medium

Impact: Medium - Contract could revert unexpectedly.

Likelihood: Low - Only occurs when pool is empty with pending rewards.

Location: `NEONStaking.sol:232-256`

Description:

```
function _updatePool() public {
    if (totalStaked != 0) {
        // ...
        accTokenPerShareNEON +=
            (multiplier * NEONDripPerBlock * PRECISION_FACTOR) /
            totalStaked; // Safe here
    }
}
```

The `_updatePool` function guards against division by zero, but `pendingRewards` view function doesn't have the same check in all paths:

```
function pendingRewards(...) external view returns (...) {
    // ...
    if (user.amount == 0 || totalStaked == 0) return (0, 0); // Guard exists

    // But if totalStaked was just set to 0 and this is called before update...
}
```

Recommendation:

Ensure all division operations have zero checks.

[M-02] Staking drip rate allows 0% which stops reward distribution

Severity: Medium

Impact: Medium - Rewards stop distributing if drip rate set to 0.

Likelihood: Low - Requires owner mistake.

Location: `NEONStaking.sol:320-328`

Description:

```
function setDripRatesN(uint256 rateNEON, uint256 rateINC) external onlyOwner {
    require(rateNEON >= 0 && rateNEON <= 20, "Invalid rateNEON");
    require(rateINC >= 0 && rateINC <= 20, "Invalid rateINC");
    DRIP_RATE_NEON = rateNEON;
    DRIP_RATE_INC = rateINC;
}
```

Rate of 0 is allowed, which would stop all reward distribution.

Recommendation:

Add minimum rate requirement: `require(rateNEON >= 1, "Rate too low");`

[L-01] Initialize function lacks access control after first call

Severity: Low

Location: `NEONStaking.sol:116-128`

Description:

While `initialize` has `onlyOwner`, it can be called multiple times, changing critical parameters.

Recommendation:

Add initialization guard: `require(address(stakedToken) == address(0), "Already initialized");`

[L-02] Hardcoded pair addresses may fail on deployment

Severity: Low

Location: `NEON.sol:86-115`

Description:

Constructor creates pairs with hardcoded token addresses (HEX, PLSX, INC, etc.). If any of these don't exist on deployment chain, constructor reverts.

[L-03] No slippage protection on liquidity operations

Severity: Low

Location: `NEON.sol:218-264`

Description:

All liquidity operations use `0` for minimum amounts:

```
pulseXRouter.addLiquidity(  
    token0, token1,  
    token0NewBalance, token1NewBalance,  
    0, // @audit No slippage protection  
    0,  
    msg.sender, block.timestamp  
);
```

Security Patterns Observed

Positive

- Solidity ^0.8.20 with overflow protection
- ReentrancyGuard on token contract
- Fee limits enforced (max 10%)
- SafeERC20 usage in staking

Concerns

- Temporary fee exclusion vulnerability
 - Hardcoded external dependencies
 - No slippage protection
 - Drip rate can be set to zero
-

Conclusion

The NEON token ecosystem has a complex fee and reward distribution system. The main concern is the temporary fee exclusion during liquidity removal which can be exploited.

We recommend addressing the High severity finding before mainnet deployment.

Overall Risk Assessment: Medium-High

This security review was conducted by Stonewall. For questions or clarifications, contact our team.