Stonewall
Smart Contract Security

# Lemonad Core Security Review

## Introduction

A time-boxed security review of the **Lemonad Core** 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 Lemonad Core

Lemonad Core consists of the foundational token and yield infrastructure:

- **LeMonad Token**: The native ERC20 token of the ecosystem
- **Yield Farming**: MasterChef-style farming with LEMON token rewards (LemonChef)
- **YieldBoostVault**: Single-sided staking vault with NFT boost mechanics
- **Treasury**: Centralized fee collection and distribution from all protocol activities

### Privileged Roles & Actors

| Role | Description |
| --- | --- |
| Owner | Can modify vault parameters, emergency withdraw, update fees and tax rates |
| Treasury Owner | Controls fee distribution and token recovery |

### Observations

- LemonChef follows standard MasterChef patterns
- YieldBoostVault implements time-based tax decay mechanism
- Treasury aggregates fees from DEX and gaming operations
- NFT integration for yield boosting

---

## Risk Classification

| | High Impact | Medium Impact | Low Impact |
| --- | --- | --- | --- |
| High Likelihood | Critical | High | Medium |
| Medium Likelihood | High | Medium | Low |
| Low Likelihood | Medium | Low | Low |

### Impact

- **High**: Leads to significant loss of user funds, protocol insolvency, or complete protocol failure
- **Medium**: Leads to partial loss of funds, temporary denial of service, or governance manipulation
- **Low**: Leads to minor issues, inconvenience, or suboptimal behavior

### Likelihood

- **High**: Attack is easy to perform and likely to happen
- **Medium**: Attack requires specific conditions but is feasible
- **Low**: Attack requires significant effort, resources, or unlikely conditions

---

## Security Assessment Summary

| Review Details | |
|---|---|
| **Protocol Name** | Lemonad Core |
| **Repository** | Private |
| **Commit** | `4bcdafa9703e197329cbc0cff193a50457decc6c` |
| **Review Date** | January 2026 |
| **Methods** | Manual review, static analysis |
| **Network** | Monad Mainnet (Chain ID: 143) |

### Deployed Contract Addresses

| Contract | Address |
|---|---|
| LeMonad Token | `0xfB5D8892297Bf47F33C5dA9B320F9D74c61955F7` |
| YieldBoostVault | `0x749F5fB1Ea41D53B82604975fd82A22538DaB65a` |
| LemonChef | `0x09C0B23c904ec03bFbf8B20686b4a42add71ad6a` |
| Treasury | `0xb64fE5E3650EBC5CD3f4816F1dcD4ce4cB52Eb3a` |

### Project Links

| Platform | Link |
|---|---|
| Website | lemonad.one |
| Twitter | @LeMONAD_Factory |
| Telegram | LeMONAD_Factory |
| Discord | Join Discord |
| Docs | GitBook |

### Scope

| Contract | SLOC |
| --- | --- |
| `LeMonad.sol` | ~50 |
| `farming/LemonChef.sol` | ~250 |
| `games/YieldBoostVault.sol` | ~320 |
| `games/Treasury.sol` | ~150 |

## Findings Summary

| ID | Title | Severity | Status |
| --- | --- | --- | --- |
| [H-01] | Emergency withdrawal allows owner to drain all user staked funds | High | Open |
| [M-01] | Treasury lacks recovery mechanism for accidentally sent tokens | Medium | Open |
| [M-02] | Vault reward calculation can permanently block user claims | Medium | Open |
| [L-01] | LemonChef allows duplicate pool creation | Low | Open |
| [L-02] | Tax rate can be set to 100% allowing seizure of all withdrawals | Low | Open |

## Findings

### [H-01] Emergency withdrawal allows owner to drain all user staked funds

**Severity:** High

**Impact:** High - Owner can withdraw all staked user funds, causing complete loss of deposits for all users.

**Likelihood:** Medium - Requires malicious or compromised owner, but no on-chain safeguards exist.

**Location:** `games/YieldBoostVault.sol:308`

**Description:**

The `emergencyWithdrawToken` function allows the owner to withdraw ANY ERC20 token from the contract, including the staked token that users have deposited:

```
function emergencyWithdrawToken(address _token, uint256 _amount) external onlyOwner {
    require(_token != address(0), "Invalid token address");
    IERC20(_token).safeTransfer(owner(), _amount);
}
```

This function has no restrictions on which tokens can be withdrawn or how much. The owner can call this with the stake token address and drain the entire contract balance, including all user deposits.

**Attack Scenario:**

1. Users stake 1,000,000 LEMON tokens in the vault
2. Owner calls `emergencyWithdrawToken(lemonToken, 1000000e18)`
3. All user funds are transferred to owner
4. Users cannot withdraw their stakes

**Recommendation:**

Add a check to prevent withdrawing staked tokens beyond available rewards:

```
function emergencyWithdrawToken(address _token, uint256 _amount) external onlyOwner {
    require(_token != address(0), "Invalid token address");

    if (_token == address(token)) {
        uint256 availableToWithdraw = token.balanceOf(address(this)) - totalStaked;
        require(_amount <= availableToWithdraw, "Cannot withdraw staked funds");
    }

    IERC20(_token).safeTransfer(owner(), _amount);
}
```

---

## [M-01] Treasury lacks recovery mechanism for accidentally sent tokens

**Severity:** Medium

**Impact:** Medium - Tokens sent outside normal game flows could be permanently locked.

**Likelihood:** Medium - Users occasionally send tokens to wrong addresses.

**Location:** `games/Treasury.sol`

**Description:**

The Treasury contract can receive funds via `receive()` but only has withdrawal functions for tracked game tokens. If tokens are accidentally sent directly to the treasury (outside normal game flows), they become permanently locked with no recovery mechanism.

**Recommendation:**

Add a generic token rescue function for non-tracked tokens:

```
function rescueToken(address _token, uint256 _amount) external onlyOwner {
    require(!isTrackedToken[_token], "Use designated withdrawal");
    IERC20(_token).safeTransfer(owner(), _amount);
}
```

---

## [M-02] Vault reward calculation can permanently block user claims

**Severity:** Medium

**Impact:** High - Users cannot claim legitimately earned rewards.

**Likelihood:** Low - Requires vault to become underfunded relative to promised rewards.

**Location:** `games/YieldBoostVault.sol:148`

**Description:**

If `totalRewardsAvailable` is less than a user's accumulated rewards, the claim function reverts:

```
require(totalRewardsAvailable >= rewards, "Insufficient vault rewards");
```

The reward rate continues accumulating based on APR regardless of actual available funds, creating a scenario where users are promised rewards they cannot claim.

**Recommendation:**

Allow partial claims up to available rewards:

```
uint256 claimableAmount = rewards > totalRewardsAvailable ? totalRewardsAvailable : rew
require(claimableAmount > 0, "Nothing to claim");
```

---

## [L-01] LemonChef allows duplicate pool creation

**Severity:** Low

**Location:** `farming/LemonChef.sol`

**Description:**

The `add()` function doesn't check if a staking token already exists, allowing duplicate pools for the same LP token.

**Recommendation:**

Track added tokens and prevent duplicates:

```
mapping(address => bool) public poolExists;

function add(...) external onlyOwner {
    require(!poolExists[_lpToken], "Pool exists");
    poolExists[_lpToken] = true;
    // ...
}
```

---

## [L-02] Tax rate can be set to 100% allowing seizure of all withdrawals

**Severity:** Low

**Location:** `games/YieldBoostVault.sol:285-296`

**Description:**

Owner can set `initialTaxRate` to 100% (10000 basis points), effectively seizing all user withdrawals.

**Recommendation:**

Add a reasonable maximum cap (e.g., 50%):

```
require(_taxRate <= 5000, "Tax rate too high");
```

## Informational Findings

### [I-01] High Daily Reward Rate Cap

Maximum 10% daily reward rate could deplete rewards rapidly. Consider lower default caps.

## Security Patterns Observed

### Positive

- Solidity ^0.8.20+ with overflow protection
- ReentrancyGuard on state-changing functions
- SafeERC20 for token transfers
- Standard MasterChef patterns in LemonChef

### Concerns

- Significant owner control over funds
- Emergency functions too powerful
- No maximum cap on tax rates

## Conclusion

The Lemonad Core contracts follow established patterns but have significant centralization risks. The main concern is the emergency withdrawal function that could allow complete drainage of user funds.

**We recommend addressing the High severity finding before deployment.**

**Overall Risk Assessment: Medium-High**

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