Stonewall
Smart Contract Security

# Global Internet Money (MoniesTree) Security Review

## Introduction

A time-boxed security review of the **Global Internet Money** protocol (MoniesTree) 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 Global Internet Money (MoniesTree)

MoniesTree is a DeFi protocol on PulseChain featuring:

- **Auction System**: 35-day daily auction where users deposit PLS to acquire tokens (each "day" = 4 hours, total 140 hours)
- **Token Vesting**: Auction participants receive vested tokens with 3% daily emissions (up to 111% return over 37 days)

- **Staking & Rewards**: Users stake MT tokens to earn rewards from reserve pool (10% of reserve distributed daily)
- **Buy-Back Mechanism**: 85% of collected PLS used to buy back tokens and build reserves
- **Buy/Sell Tax**: Configurable buy tax (max 10%) and sell tax (max 15%) on LP trades

### Privileged Roles & Actors

| Role | Description |
|---|---|
| Admin | Full control over protocol parameters, can pause auctions, change addresses |
| Team | Can set taxes, pause rewards, adjust slippage, manage exclusions |
| Fee Recipient | Receives 15% of collected PLS from daily auctions |

### Observations

- Complex three-contract architecture (MoniesTree + StakeContract + ReserveContract)
- Auction mechanism with 4-hour "days" (35 total = 140 hours auction period)
- LP tokens permanently burned (sent to zero address)
- No timelock or multisig on admin functions
- Tax only applies to LP buys/sells, not regular transfers

---

## 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** | Global Internet Money (MoniesTree) |
| **Repository** | Private |
| **Commit** | N/A |
| **Review Date** | January 2026 |
| **Methods** | Manual review, static analysis |
| **Network** | PulseChain |

### Scope

| Contract | SLOC |
|---|---|
| `new.sol` (MoniesTree) | ~650 |
| `StakeContract` (embedded) | ~290 |
| `ReserveContract` (embedded) | ~200 |

---

## Findings Summary

| ID | Title | Severity | Status |
|---|---|---|---|
| [C-01] | Unbounded loop in updateRewards() causes permanent DoS | Critical | Open |
| [H-01] | Admin can pause auctions indefinitely and trap user PLS | High | Open |

| | | | |
|---|---|---|---|
| [H-02] | No slippage protection on initial liquidity addition | High | Open |
| [M-01] | Integer division precision loss in reserve splits | Medium | Open |
| [M-02] | Missing zero-address checks on critical setters | Medium | Open |
| [M-03] | 7-day stake lock may be inconsistent with 4-hour "days" | Medium | Open |
| [L-01] | SafeMath unnecessary in Solidity 0.8+ | Low | Open |
| [L-02] | Missing events for critical state changes | Low | Open |
| [L-03] | LP tokens permanently burned without recovery option | Low | Open |

---

## Detailed Findings

### [C-01] Unbounded Loop in updateRewards() Causes Permanent DoS

**Severity:** Critical

**Location:** `StakeContract:467-469`

**Description:**

The `updateRewards()` function iterates through ALL stakers in the `allStakers` array:

```
// Loop through all users and update their pending rewards
for (uint256 i = 0; i < allStakers.length; i++) {
    updateUserRewards(allStakers[i]);
}
```

As the number of stakers grows, this loop will eventually exceed the block gas limit. Since `updateRewards()` is called by `deposit()`, `withdraw()`, `claimRewards()`, and `compoundRewards()`, the entire staking system becomes permanently unusable.

**Impact:**

- Complete denial of service for all staking operations
- Users unable to withdraw their staked tokens
- Users unable to claim accumulated rewards
- Protocol becomes unusable after ~500-1000 users

**Recommendation:**

Implement a pull-based reward system where users update only their own rewards:

```
function updateRewards() public {
    uint256 currentDay = _auction.readCurrentDay();
    if (!rewardsActive || lastRewardDay >= currentDay || rewardsPaused) return;

    // Update global accRewardPerStake only
    for (uint256 day = lastRewardDay; day < currentDay; day++) {
        // ... existing reward calculation per day
    }
    lastRewardDay = currentDay;
}

// Users call updateUserRewards(msg.sender) themselves before actions
```

---

## [H-01] Admin Can Pause Auctions Indefinitely and Trap User PLS

**Severity:** High

**Location:** `MoniesTree:1268-1287`

**Description:**

The admin can pause auctions via `pauseUnpauseAuctions()`. While paused:

- Users cannot enter new auctions
- Users who deposited PLS for future days cannot claim tokens
- The `buybackAndBuild()` function cannot execute
- PLS remains trapped in the contract

Although `permanentlyDisableAuctionPause()` exists, there's no mechanism to force the admin to call it, and no emergency withdrawal for users.

**Impact:**

- Users' PLS deposits can be held hostage
- No recourse for users if admin becomes malicious or loses keys

**Recommendation:**

1. Add maximum pause duration with automatic unpause

2. Implement emergency withdrawal mechanism for users
3. Add timelock for pause functionality

---

## [H-02] No Slippage Protection on Initial Liquidity Addition

**Severity:** High

**Location:** `MoniesTree:1310-1319`

**Description:**

The `addFirstDaysLiq()` function adds initial liquidity with zero slippage protection:

```
(, , uint256 amtLiquidity) = _router.addLiquidityETH{
    value: collectedETHtoLiq
}(
    contrAddr,
    amountAuctionTokenToAdd,
    0, // amountTokenMin = 0
    0, // amountETHMin = 0
    contrAddr,
    block.timestamp + 100
);
```

**Impact:**

- Sandwich attack can steal significant value during initial liquidity provision
- MEV bots can front-run the transaction and extract value
- First-day auction participants receive less value than expected

**Recommendation:**

Calculate minimum amounts based on expected price:

```
uint256 amountTokenMin = amountAuctionTokenToAdd * 95 / 100;
uint256 amountETHMin = collectedETHtoLiq * 95 / 100;
```

---

## [M-01] Integer Division Precision Loss in Reserve Splits

**Severity:** Medium

**Location:** Multiple locations

**Description:**

Several places divide amounts by 2, losing precision for odd amounts:

```
// MoniesTree:1108-1109
taxBuyBackReserveContract.updateReserveAuction(true, tokenAmount / 2);
taxBuyBackReserveContract.updateReserveStake(true, tokenAmount / 2);

// MoniesTree:1172-1173
taxBuyBackReserveContract.updateReserveAuction(true, taxAmount / 2);
taxBuyBackReserveContract.updateReserveStake(true, taxAmount / 2);
```

For odd amounts, 1 wei is lost per split. Over many transactions, this accumulates.

**Impact:**

- Slow token leak over time
- Reserve balances don't match expected values

**Recommendation:**

Track remainder and distribute it:

```
uint256 half = tokenAmount / 2;
uint256 remainder = tokenAmount - (half * 2);
taxBuyBackReserveContract.updateReserveAuction(true, half + remainder);
taxBuyBackReserveContract.updateReserveStake(true, half);
```

---

## [M-02] Missing Zero-Address Checks on Critical Setters

**Severity:** Medium

**Location:** `MoniesTree:1075-1081`

**Description:**

The `setAdminAddresses()` function allows setting admin and fee recipient to any address:

```
function setAdminAddresses(
    address payable admin,
```

```
    address payable feeRecipient
) external onlyAdmin {
    _admin = admin;
    _feeRecipient = feeRecipient;
}
```

**Impact:**

- Setting admin to zero address permanently locks all admin functions
- Setting fee recipient to zero address burns 15% of all PLS collected
- Irreversible loss of protocol control

**Recommendation:**

```
require(admin != address(0), "Invalid admin");
require(feeRecipient != address(0), "Invalid fee recipient");
```

---

## [M-03] 7-Day Stake Lock May Be Inconsistent with 4-Hour "Days"

**Severity:** Medium

**Location:** `StakeContract:335`

**Description:**

The stake lock time is defined as `7 days` in real time:

```
uint256 public stakeLockTime = 7 days; // TODO change to 7 days
```

However, the protocol uses 4-hour "days" for auctions ( `oneDay = 4 hours` ). Users may expect the lock to be 7 auction "days" (28 hours) but get locked for 7 real days (168 hours).

**Impact:**

- Users locked longer than expected
- Confusion between auction days and real days
- Potential for user funds to be inaccessible during volatile periods

**Recommendation:**

Align lock period with protocol's day definition or clearly document the difference:

```
uint256 public stakeLockTime = 7 * 4 hours; // 7 protocol "days" = 28 hours
```

---

## [L-01] SafeMath Unnecessary in Solidity 0.8+

**Severity:** Low

**Location:** Throughout all contracts

**Description:**

The contracts use SafeMath library despite being Solidity 0.8.0+, which has built-in overflow/underflow protection. This adds unnecessary gas costs.

**Recommendation:**

Remove SafeMath and use native arithmetic operators.

---

## [L-02] Missing Events for Critical State Changes

**Severity:** Low

**Location:** Multiple functions

**Description:**

Several important state changes don't emit events:

- `setBuyAndSellTax()` - tax rate changes
- `setAdminAddresses()` - admin/fee recipient changes
- `setExcludedFromTax()` - tax exclusion changes
- `permanentlyDisableAuctionPause()` - permanent state change
- `setStakeRewardFactor()` / `setAuctionTokenFactor()` - reserve factor changes

**Impact:**

- Reduced transparency for users monitoring the protocol
- Difficult to track historical changes
- Limits off-chain monitoring capabilities

**Recommendation:**

Add appropriate events for all state-changing functions.

---

### [L-03] LP Tokens Permanently Burned

**Severity:** Low

**Location:** `MoniesTree:1326`

**Description:**

Initial LP tokens are transferred to the zero address:

```
IERC20(liquidityPoolAddress).transfer(address(0), amtLiquidity);
```

**Impact:**

- LP tokens permanently lost
- Cannot remove liquidity in emergency
- No upgrade path if issues are discovered

**Recommendation:**

Consider sending to a dead address with known properties or implementing governance over LP tokens.

---

## Centralization Risks

| Risk | Description | Severity |
|---|---|---|
| Admin Key Compromise | Single admin controls all protocol parameters | High |
| Tax Manipulation | Team can set buy tax up to 10%, sell tax up to 15% | Medium |
| Pause Abuse | Admin can pause auctions trapping user funds | High |
| No Timelock | All admin changes take effect immediately | Medium |
| Reserve Factor Control | Team can set reward/auction factors from 1-100% | Medium |

## Recommendations

1. **Implement Pull-Based Rewards**: Replace the unbounded loop with a system where users update only their own rewards
2. **Add Timelock**: Implement a timelock contract for all admin functions with at least 24-48 hour delay
3. **Emergency Withdrawal**: Add mechanism for users to withdraw auction deposits if paused beyond threshold
4. **Multisig Admin**: Replace single admin key with multisig wallet
5. **Slippage Protection**: Add minimum output amounts to liquidity operations
6. **Zero-Address Validation**: Add checks to prevent setting critical addresses to zero
7. **Event Logging**: Add events for all state-changing admin functions
8. **Clarify Lock Period**: Document or align stake lock with protocol's 4-hour day definition

## Conclusion

Global Internet Money (MoniesTree) implements an auction and vesting mechanism on PulseChain, but contains a critical issue that should be addressed before production use. The unbounded loop in the staking contract will cause permanent protocol failure at scale. The centralization risks and lack of timelock also present significant trust assumptions for users.

We recommend addressing the Critical severity finding before deployment, and implementing the recommended mitigations for High and Medium severity issues.