Security Audit Report ? OnChain RPG Battle

Prepared for: Confidential Client

**Prep

Executive Summary

AuditAid performed a full smart contract security review of the **OnChain RPG Battle**

codeb

Findings at a glance

| Severity | Count | Summary |

|-------

The **production contract** should not be treated as production-ready for real-value gameplay

until *

Code quality scored **62/100** after fuzz/invariant tests were added. Unit test line coverage

on v1

Scope

| In scope | Out of scope |

|-------

Context documents read: `target/README.md`, `target/legacy/README.md`.

Methodology

Multi-tier AuditAid pipeline: static analysis (Slither), parallel

manu

Build: Foundry at `target/`, Solidity 0.8.18+, optimizer 200 runs, IR pipeline enabled.

Findings

Critical.01 ? Legacy `expUp()` is public and bypasses all progression rules

Source ID: C-01

**Sev

Scope: Legacy contract only. Production v1 correctly uses `internal expUp`.

What the problem is

On the legacy contract, any wallet can call `expUp()` with an arbitrary experience value and

instan

How it works

The v1 contract restricts `expUp` to internal calls from combat paths. The legacy contract

expos

Step-by-step exploit

1. Attacker calls `registerPlayer` with the minimum fee.

2. Atta

Impact

Complete breakdown of progression economics on any deployment using legacy bytecode.

Why this severity

This is a direct, single-transaction unauthorized state change with no preconditions beyond

calling

Recommendation

Change `expUp` to `internal` to match v1, or remove legacy from deployment scope entirely.

Code reference

```
    function expUp(uint256 _exp) public returns(bool){
```

PoC: Confirmed ? `target/test/LegacyAuditPoC.t.sol` (PASS)

High.01 ? Combat randomness is predictable and identical within each transaction

Source ID: H-01

**Sev

What the problem is

All critical hits, damage variance, and ETH loot drops depend on a ?random? value derived from

block

How it works

`randomNumber()` hashes `block.timestamp`, `block.prevrandao`, `block.number`, and `msg.

Step-by-step exploit

1. Attacker registers and funds the contract.

2. Off

Impact

Systematic unfairness; selective extraction from the shared ETH fee pool. Undermines the core game

Why this severity

Exploitation is realistic for any motivated player with simulation tooling; impact is repeated econo

Recommendation

Integrate verifiable randomness (e.g., Chainlink VRF) with per-round request/fulfill, or at minim

Code reference

```
    function randomNumber() public view returns (uint256) {
```

PoC: Confirmed ? `target/test/AuditPoC.t.sol:test_PoC_randomNumber_sameValueWithinTx` (PASS

Medium.01 ? Players can re-register, bypass revive fees, and mint achievements without re-grind

Source ID: M-01

**Sev

What the problem is

Nothing prevents a registered player from calling `registerPlayer` again. Re-registration resets

How it works

`registerPlayer` unconditionally overwrites the `Player` struct. Auxiliary mappings live in separ

Step-by-step exploit

1. Player grinds 100 goblin kills (`monsterSlayeds[player][1] >= 100`).

2. Pla

Impact

Achievement NFT inflation; revive-fee bypass; accidental or griefing stat wipes.

Why this severity

Confirmed by PoC and failing invariant INV-04. Does not drain the full pool but breaks

docun

Recommendation

Add `require(players[msg.sender].lv == 0, "Already registered")` (or equivalent). On re-

regist

PoC: Confirmed ? `AuditPoC.t.sol:test_PoC_reregister_preservesMonsterSlayeds` (PASS)

Medium.02 ? Winning a mutual kill can leave a player ?alive? with zero HP

Source ID: M-02

**Sev

What the problem is

If the player kills the enemy and takes lethal damage in the **same round**, the code breaks

out of

Step-by-step exploit

1. Enter a battle where same-round mutual kill occurs (PRNG-dependent; easier with H-01

simula

Impact

Death/revive fee economics bypass; inconsistent game state.

Why this severity

Breaks a core lifecycle invariant; composes with predictable randomness. Structural line-proof;

dedica

Recommendation

After the combat loop, if `currentHp == 0`, set `isAlive = false`. Evaluate player death before

award

Medium.03 ? Smart contract wallets cannot claim achievement NFTs

Source ID: M-03

**Sev

What the problem is

Achievements are minted with OpenZeppelin `_safeMint`, which requires contract recipients to

impler

Impact

Documented product feature permanently unavailable to a significant wallet class; grind on that

addre

Why this severity

Real user-flow breakage for core feature, not theoretical.

Recommendation

Allow an explicit EOA recipient parameter, or reject contract addresses at registration with

clear

Medium.04 ? Legacy battle loops break when paying for more than 256 rounds

Source ID: M-04

**Sev

The loop counter is `uint8`. When `_battleRounds > 256`, the counter wraps and the transaction

runs o

Medium.05 ? Legacy PvP has no level-difference protection

Source ID: M-05

**Sev

Production v1 enforces a 10-level PvP cap. Legacy allows max-level players to farm low-level

target

Appendix ? Additional Observations

| Source ID | Severity | Title |

|-------

Recommendations Summary

1. **Do not deploy legacy** ? Critical `expUp` bypass (C-01).

2. **R

Residual Risk

- Branch coverage on v1 remains **35.82%**.

- Own

Disclaimer

This assessment reflects the codebase at the time of review. It is not a guarantee of future

securi

Tucker Sossaman

AuditA