Security Audit report for RareMarket

Prepared by: **RareLabs Security team**
Date: **September 08 2025**
version: **2.2**

Abstract

This security audit, conducted by the RareLabs Security Team, is an updated version (2.2) following additional analysis of the RareMarket.sol smart contract and its dependencies. The audit re-evaluated the code to verify the effectiveness of prior fixes and identify new vulnerabilities using Slither static analysis and manual review.

Contents

1 Executive Summary

This security audit, conducted by the RareLabs Security Team, updates the previous audit (Version 2.1) with additional findings from the RareMarket.sol smart contract and its dependencies, including Royalties.s and OpenZeppelin libraries. The audit identified 21 issues, as summarized in Table 1.

Issue Type	Instances	Severity
Assembly Usage	3	Informational
Pragma Variability	6	Informational
Dead Code	4	Informational
Solidity Version Issues	5	Informational
Low-Level Calls	4	Informational
Naming Convention	3	Informational
Cache Array Length	1	Optimization

Table 1: Summary of Audit Findings

The audit confirms resolution of high- and low-severity issues from prior versions. Remaining concerns are informational, with optimization opportunities. Further testing is recommended.

2 Introduction

This Version 2.2 report updates the previous audit following additional analysis of the RareMarket.sol contract, which facilitates NFT listings, auctions, offers, and royalty distributions, and the Royalties.sol contract, which handles royalty management and liquidity pools.
The audited contracts include:

- RareMarket.sol: Manages NFT listings, auctions, offers, and payment processing.
- Royalties.sol: Handles royalty calculations and liquidity pool operations.
- Supporting OpenZeppelin libraries.

3 Audit Scope and Methodology

The audit re-evaluated the code using Slither static analysis and manual review, focusing on newly identified areas.

3.1 Methodology

- Static Analysis: Ran Slither to detect remaining vulnerabilities.
- Manual Review: Verified fixes and checked for new issues.

3.2 Scope

The audit covered updated components of RareMarket.sol and dependencies, based on the codebase snapshot dated September 05, 2025.

4 Findings and Recommendations

Findings are categorized by severity, with detailed descriptions, impacts, and mitigation strategies.

4.1 Informational Findings

4.1.1 I1: Assembly Usage

Description: Inline assembly increases code complexity and risk of errors.
Instances:

- ID-0 (SafeERC20.sol, Lines 173191): In _callOptionalReturn

```
    // INLINE ASM (Lines 176-186)
```

- ID-1 (SafeERC20.sol, Lines 201211): In _callOptionalReturnBool

```
    // INLINE ASM (Lines 205-209)
```

- ID-2 (Address.sol, Lines 138148): In _revert

```
    // INLINE ASM (Lines 142-144)
```

Impact: Reduced readability and potential for subtle bugs.
Recommendation: Replace assembly with high-level Solidity constructs where possible.

4.1.2 I2: Pragma Variability

Description: Multiple Solidity versions complicate maintenance.
Instances:

- $^{0}.8.20, >= 0.8.4, >= 0.6.2, >= 0.4.16, >= 0.5.0, ^{0}.8.28$ **Impact:** Inconsistent compiler behavior.

Recommendation: Standardize to a single, recent version (e.g., $^{0}.8.28$).

4.1.3 I3: Dead Code

Description: Unused functions increase contract size.
Instances:

- ID-3 (AccessControl.sol, Lines 168172): _setRoleAdmin
- ID-4 (Context.sol, Lines 2527): _contextSuffixLength

- ID-5 (Context.sol, Lines 2123): _msgData
- ID-6 (ReentrancyGuard.sol, Lines 8486): _reentrancyGuardEntered

Impact: Unnecessary gas costs.
Recommendation: Remove unused functions.

4.1.4 I4: Solidity Version Issues

Description: Older Solidity versions contain known bugs.
Instances:

- $^{0}.8.20 : Verbatim Invalid Deduplication, etc. >= 0.8.4 : Full Inliner Non Expression Split Argum$

- >=0.6.2: MissingSideEffectsOnSelectorAccess, etc.
- >=0.4.16: DirtyBytesArrayToStorage, etc.
- >=0.5.0: DirtyBytesArrayToStorage, etc.

Impact: Potential exploitation of known compiler bugs.
Recommendation: Upgrade to a stable, recent version (e.g., $^{0}.8.28$).

4.1.5 I5: Low-Level Calls

Description: Low-level calls increase risk due to reduced safety.
Instances:

- ID-7 (Address.sol, Lines 3342): In sendValue

```
(success, returndata) = recipient.call{value: amount}(); // Line 38
```

- ID-8 (Address.sol, Lines 7581): In functionCallWithValue

```
(success, returndata) = target.call{value: value}(data); // Line 79
```

- ID-9 (Address.sol, Lines 8790): In functionStaticCall

```
(success, returndata) = target.staticcall(data); // Line 88
```

- ID-10 (Address.sol, Lines 9699): In functionDelegateCall

```
(success, returndata) = target.delegatecall(data); // Line 97
```

Impact: Reduced safety and error handling.
Recommendation: Use high-level function calls or ensure robust checks.

4.1.6 I6: Naming Convention

Description: Non-standard naming conventions reduce code clarity.
Instances:

- ID-11 (market.sol, Line 189): setRoyaltyEngineAddress
- ID-12 (market.sol, Line 195): setRareMarket
- ID-13 (market.sol, Line 83): NATIVE$_{C}URRENCY$ **Impact:** Reduced maintainability.

Recommendation: Adopt mixedCase naming (e.g., newRoyaltyEngine, nativeCurrency).

4.2 Optimization Findings

4.2.1 O1: Cache Array Length

Description: Repeatedly accessing array length in loops increases gas costs.
Instances:

- ID-14 (market.sol, Line 658): In loop condition i < allPoolIdentifie

Impact: Higher gas costs due to redundant storage reads.
Recommendation: Cache array length:

```
uint256 length = allPoolIdentifiers.length;
for (uint256 i = 0; i < length; i++) { ... }
```

5 Conclusion

This Version 2.2 audit confirms ongoing improvements, identifying 21 issues. All high- and low-severity issues from prior audits remain resolved. The RareLabs team recommends addressing informational findings and optimizing the codebase.

This audit does not guarantee the absence of all vulnerabilities. Continuous security practices, including regular audits, are essential.