shieldify

Roots of Embervault

Ruyui

SECURITY REVIEW

Date: 28 January 2026

CONTENTS

1. About Shieldify

Positioned as the first hybrid Web3 Security company, Shieldify shakes things up with a unique subscription-based auditing model that entitles the customer to unlimited audits within its duration, as well as top-notch service quality thanks to a disruptive 6-layered security approach. The company works with very well-established researchers in the space and have secured multiple millions in TVL across protocols, also can audit codebases written in Solidity, Vyper, Rust, Cairo, Move and Go.

Learn more about us at shieldify.org.

2. Disclaimer

This security review does not guarantee bulletproof protection against a hack or exploit. Smart contracts are a novel technological feat with many known and unknown risks. The protocol, which this report is intended for, indemnifies Shieldify Security against any responsibility for any misbehavior, bugs, or exploits affecting the audited code during any part of the project's life cycle. It is also pivotal to acknowledge that modifications made to the audited code, including fixes for the issues described in this report, may introduce new problems and necessitate additional auditing.

3. About Roots of Embervault - Ruyui

Embervault is an online action RPG featuring character progression through attribute-based builds, stamina-driven combat with risk-based resource loss on death, and a fully player-driven economy. Players craft, upgrade, and trade equipment and consumables via an integrated marketplace.

Seasonal competitive events distribute on-chain rewards, while a referral system provides direct wallet-based fee sharing.

Learn more about Embervault's concept and the technicalities behind it: here

4. Risk Classification

Severity	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** - results in a significant risk for the protocol's overall well-being. Affects all or most users
- **Medium** - results in a non-critical risk for the protocol affects all or only a subset of users, but is still unacceptable
- **Low** - losses will be limited but bearable - and covers vectors similar to griefing attacks that can be easily repaired

4.2 Likelihood

- **High** - almost certain to happen and highly lucrative for execution by malicious actors

- **Medium** – still relatively likely, although only conditionally possible
- **Low** – requires a unique set of circumstances and poses non-lucrative cost-of-execution to rewards ratio for the actor

5. Security Review Summary

The security review lasted 5 days with a total of 80 hours dedicated to the audit by the Shieldify team.

Overall, the code is well-written. The audit report contributed by identifying one High and three Low severity issues. They're related to flaws in the referral and purchase flow that allow bypassing signup requirements, enable malicious referees to disrupt purchases, and introduce DoS and signature replay risks.

The Ruyui team has done a great job with their test suite and provided support and responses to all of the questions that the Shieldify researchers had.

5.1 Protocol Summary

Project Name	**Roots of Embervault – Ruyui**
Repository	RuyuiMarketPlace
Type of Project	GameFi, Marketplace
Security Review Timeline	5 days
Review Commit Hash	66c6818e7765c74d6c16f887c1259411c09ad7a9
Fixes Review Commit Hash	42d3b627341da8d0699d12a4a732c956e43bc6b8

5.2 Scope

The following smart contracts were in the scope of the security review:

File	**nSLOC**
RuyuiShop.sol	350
RuyuiMarketplace.sol	428
Total	778

6. Findings Summary

The following number of issues have been identified, sorted by their severity:

- **High** issues: **1**
- **Low** issues: **3**
- **Info** issues: **2**

ID	Title	Severity	Status
[H-01]	It Is Possible to Purchase In-Game Items Without Signing Up Which Results in Loss of Commission Fee for Referee	High	Fixed
[L-01]	Malicious Referee Can DoS Shop Purchases for Their Referrals	Low	Fixed
[L-02]	Unbounded Array DoS Vulnerability	Low	Fixed
[L-03]	Signatures Lack Expiration Timestamp	Low	Fixed
[I-01]	An Unsigned Up Referrer Can Be Used as Referrer in the On-chain Path	Info	Fixed
[I-02]	Missing Zero Merkle Root Validation in Shop Configuration	Info	Fixed

7. Findings

[H-01] It Is Possible to Purchase In-Game Items Without Signing Up Which Results in Loss of Commission Fee for Referee

Severity

High Risk

Description

When purchasing a shop item and a battle pass, it is not checked whether the user has signed up or not. Signing up before purchase is necessary because the referee is set for the user during sign-up and at the time of purchasing an item, the referee gets their commission.

Calling purchase functions without signing up is dangerous because those purchase functions (`RuyuiShop.purchaseBattlePass()` and `RuyuiShop.purchaseShopItem()`) update states for the user:

```
userShopItemPurchases[msg.sender][itemId]++

hasPurchasedBattlePass[msg.sender][battlePassId] = true;
```

Those functions assume the caller is a user and update these mappings according to that. But without signing up, a caller can't be a user. If the protocol later plans some perks for users, for example, an airdrop, then the caller will get the airdrop without being a user of the game, additionally, he bypasses the signup fee payment.

Location of Affected Code

File: RuyuiShop.sol#L440-L463

File: RuyuiShop.sol#L383-L438

Impact

- The referee will not get his commission.
- Incorrect state update.
- Avoids signup fee.
- Breaks the rules of the game, where signing up is important.

Recommendation

Consider implementing this check in `purchaseBattlePass()` and `purchaseShopItem()` functions:

```
require(hasSignedUp(msg.sender), "Has not signed up yet");
```

Team Response

Fixed.

[L-01] Malicious Referee Can DoS Shop Purchases for Their Referrals

Severity

Low Risk

Description

When a user has a referee set, `purchaseShopItem()` pays the referee commission using a low-level call that must succeed. A malicious referee can deploy a contract whose `receive()` accepts the first commission transfer (to look legitimate during setup or onboarding) and then reverts on all subsequent transfers.

Because the `userReferee` mapping permanently ties users to that referee, the contract effectively bricks purchases for all mapped users and offers no way to recover. So if the user tricks the admin api to sign the message with the malicious contract address as referee for other accounts and tricks them into using this account as referee, then all subsequent purchases will result in DoS.

Location of Affected Code

File: RuyuiShop.sol#L383-L438

```
function purchaseShopItem(...) external payable nonReentrant {
    // code
    address referee = userReferee[msg.sender];
    if (referee != address(0)) {
        uint256 commission = (price * shopItemRefereeCommissionBps) /
           10000;
        uint256 adminAmount = price - commission;

        if (commission > 0) {
            (bool commissionSuccess, ) = referee.call{value: commission}(
              "");
            require(commissionSuccess, "Commission transfer failed");
        }

        (bool withdraw, ) = admin.call{value: adminAmount}("");
        require(withdraw, "Failed to withdraw ETH");
    } else {
        (bool withdraw, ) = admin.call{value: price}("");
        require(withdraw, "Failed to withdraw ETH");
    }
    // code
}
```

Impact

A referee can permanently block shop purchases for any users mapped to them, resulting in denied access to purchases and lost protocol revenue from those blocked transactions, even if the referee accepted the first commission transfer.

Proof of Concept

1. Attacker deploys a referee contract that accepts only the first ETH transfer:

```
contract SelectiveReferee {
    mapping(address => bool) private hasReceived;

    receive() external payable {
        if (hasReceived[msg.sender]) revert("refuse commissions");
        hasReceived[msg.sender] = true;
    }
}
```

2. Users sign up using this referee.
3. The first commission transfer succeeds, making the referee appear legitimate.
4. All subsequent shop purchases by referred users revert when commission is sent.

Recommendation

Put a check in the contract that the admin can deactivate a malicious referee, in that case, the 10% fee will not be deducted and the whole amount will be transferred to the admin.

Or Pull design is preferred instead of push design, where creating specialised functions like `claim()` is recommended to track the balances of referrers.

Team Response

Fixed.

[L-02] Unbounded Array DoS Vulnerability

Severity

Low Risk

Description

The `allSellers` array grows indefinitely without bounds, causing DoS when gas limits are exceeded during array iterations. Although there is an admin signature requirement, there is no limit on the number of sellers that can be added to the marketplace.

Location of Affected Code

File: RuyuiMarketplace.sol#L387

```
address[] public allSellers;

function _createListing( bytes32 key, address seller, uint256 itemId,
    uint256 qty, uint256 price, uint8 currencyType, string calldata info )
     internal {
   // code
   if (!sellerExists[seller]) {
       sellerExists[seller] = true;
       allSellers.push(seller); // DoS risk
   }
}
```

Impact

Once the array reaches a size, the `_createListing()` function's array push operation may consume excessive gas, preventing new sellers from joining the marketplace entirely.

Recommendation

It's recommended NOT to use arrays for unbounded on-chain enumeration.** Remove the `allSellers` array entirely and use mappings with off-chain indexing.

Recommended Architecture:

On-Chain (Smart Contract):

```
// Remove: address[] public allSellers;
// Keep only mappings - O(1) operations:
mapping(address => bool) public sellerExists;
mapping(address => uint256[]) public sellerItemIds;
mapping(bytes32 => Listing) public listings;

// Emit event for off-chain indexing
event SellerRegistered(address indexed seller, uint256 timestamp);
event SellerRemoved(address indexed seller, uint256 timestamp);
```

Off-Chain (Backend/Indexer):

- Use **The Graph Protocol** or a custom indexer (e.g., Ponder, Subsquid, or simple event listener)
- Index `SellerRegistered` and `ItemListed` events to build seller database
- Store seller addresses and metadata in a traditional database (PostgreSQL, MongoDB)
- Implement pagination at the database level with efficient SQL queries
- Provide GraphQL/REST API for front-end queries with offset/limit pagination

Team Response

Fixed.

[L-03] Signatures Lack Expiration Timestamp

Severity

Low Risk

Description

Admin signatures for listings (`RuyuiMarketplace`) and signups (`RuyuiShop`) have no expiration. Once generated, signatures remain valid indefinitely, preventing graceful revocation.

Location of Affected Code

File: RuyuiMarketplace.sol#L274-L295

```
function _hashListingForSig(address seller, uint256 itemId, uint256 qty,
    uint256 price, uint8 currencyType, bytes32 nonce) internal view
    returns (bytes32) {
     return
         keccak256(
             abi.encode(
                 seller,
                 itemId,
                 qty,
                 price,
                 currencyType,
                 nonce,
                 block.chainid,
                 address(this)
             )
         );
}
```

File: RuyuiShop.sol#L489-L495

```
function signUp(address referee, bytes calldata signature) external
    payable nonReentrant {
   // code
   bytes32 messageHash = keccak256(
       abi.encodePacked(
           msg.sender,
           referee,
           block.chainid,
           address(this)
       )
   );
   // code
}
```

Impact

Since signatures never expire, stale signatures can be used long after market conditions have changed, potentially allowing users to create listings at outdated prices. Additionally, there is no way to invalidate unused signatures without changing the admin key entirely, which would invalidate all outstanding signatures.

Proof of Concept

- Admin signs a listing approval for seller Alice at 1 ETH price
- Protocol later updates policy to ban certain items or increase minimum prices
- Alice still uses the old signature to create the now-disallowed listing
- Admin cannot revoke the signature without changing the entire admin key

Recommendation

Add a `deadline` parameter to signature hashes and validate with `require(block.timestamp <= deadline, "Signature expired")`.

Team Response

Fixed.

[I-01] An Unsigned Up Referrer Can Be Used as Referrer in the On-chain Path

Severity

Informational Risk

Description

The `signUp()` flow accepts a non-zero `referee` as long as an admin signature is provided, but does not validate on-chain that the `referee` has actually signed up. This allows referral rewards to be paid to arbitrary addresses that are not genuine game accounts.

Location of Affected Code

File: RuyuiShop.sol#L488-L511

```
function signUp(address referee, bytes calldata signature) external
   payable nonReentrant {
  // code
  if (referee != address(0)) {
      bytes32 messageHash = keccak256(
          abi.encodePacked(
              msg.sender,
              referee,
              block.chainid,
              address(this)
          )
      );
      bytes32 ethSignedMessageHash = messageHash.toEthSignedMessageHash()
         ;
      require(!usedSignatures[ethSignedMessageHash], "Signature already
         used");
      address recoveredSigner = ethSignedMessageHash.recover(signature);
      require(recoveredSigner == admin, "Not signed by admin");
      usedSignatures[ethSignedMessageHash] = true;
      commission = (signUpFee * refereeCommissionBps) / 10000;
      (bool success, ) = referee.call{value: commission}("");
      require(success, "Commission transfer failed");
      actualReferee = referee;
      userReferee[msg.sender] = referee;
  }
  // code
}
```

Impact

Referral commissions can be paid to addresses that never created a game account on-chain, weakening referral integrity and enabling abuse if off-chain checks fail.

Recommendation

Require that the referee has a valid on-chain signup before paying commission and storing the referral:

- `require(hasSignedUp[referee], "Referee not signed up");`

Place this check inside the `if (referee != address(0))` block before transferring commission.

Team Response

Fixed.

[I-02] Missing Zero Merkle Root Validation in Shop Configuration

Severity

Informational Risk

Description

The `setShopItemsMerkleRoot()` function lacks validation against `bytes32(0)`, allowing uninitialized or invalid merkle root state.

Location of Affected Code

File: RuyuiShop.sol#L379-L381

```
function setShopItemsMerkleRoot(bytes32 _merkleRoot) external onlyAdmin {
    shopItemsMerkleRoot = _merkleRoot; // No zero validation
}
```

Impact

Admin could accidentally set `bytes32(0)` as the Merkle root, causing a shop malfunction. Users would receive misleading "Invalid proof" errors instead of "Shop not configured", making diagnosis difficult and reducing contract robustness.

Recommendation

Add validation to prevent zero Merkle roots:

```
function setShopItemsMerkleRoot(bytes32 _merkleRoot) external onlyAdmin {
    require(_merkleRoot != bytes32(0), "Invalid merkle root");
    shopItemsMerkleRoot = _merkleRoot;
}

function purchaseShopItem(...) external payable nonReentrant {
    require(shopItemsMerkleRoot != bytes32(0), "Shop not configured");
    // code
}
```

Team Response

Fixed.

, our shielding · Your smart contracts, our shielding · Your smart c

shieldify

Thank you!