Three Sigma

Code Audit

PRC

Pixel Race Club Safari Racers

Disclaimer

Code Audit

Pixel Race Club Safari Racers

Disclaimer

The ensuing audit offers no assertions or assurances about the code's security. It cannot be deemed an adequate judgment of the contract's correctness on its own. The authors of this audit present it solely as an informational exercise, reporting the thorough research involved in the secure development of the intended contracts, and make no material claims or guarantees regarding the contract's post-deployment operation. The authors of this report disclaim all liability for all kinds of potential consequences of the contract's deployment or use. Due to the possibility of human error occurring during the code's manual review process, we advise the client team to commission several independent audits in addition to a public bug bounty program

Table of Contents

Code Audit

Pixel Race Club Safari Racers

Table of Contents

Summary

Code Audit

Pixel Race Club Safari Racers

Summary

Three Sigma audited Pixel Race Club in a 0.4 person week engagement. The audit was conducted on 29/04/2025.

Protocol Description

Pixel Race Club is an online multiplayer kart racing game on Abstract, designed in voxel graphical style. The game features engaging and intuitive racing mechanics and has been designed around building social interactions and encouraging intense competition. The game features exclusive benefits for PRC related NFT holders in game, to give them a powerful advantage in their races.

Scope

Code Audit

Pixel Race Club Safari Racers

Scope

Filepath	nSLOC
src/interfaces/IPRCSR.sol	12
src/PRCSR.sol	164
src/UUPSProxy.sol	10
Total	**186**

Methodology

Code Audit

Pixel Race Club Safari Racers

Methodology

To begin, we reasoned meticulously about the contract's business logic, checking security-critical features to ensure that there were no gaps in the business logic and/or inconsistencies between the aforementioned logic and the implementation. Second, we thoroughly examined the code for known security flaws and attack vectors. Finally, we discussed the most catastrophic situations with the team and reasoned backwards to ensure they are not reachable in any unintentional form.

Taxonomy

In this audit, we classify findings based on Immunefi's Vulnerability Severity Classification System (v2.3) as a guideline. The final classification considers both the potential impact of an issue, as defined in the referenced system, and its likelihood of being exploited. The following table summarizes the general expected classification according to impact and likelihood; however, each issue will be evaluated on a case-by-case basis and may not strictly follow it.

Impact / Likelihood	LOW	MEDIUM	HIGH
NONE	None		
LOW	Low		
MEDIUM	Low	Medium	Medium
HIGH	Medium	High	High
CRITICAL	High	Critical	Critical

Project Dashboard

Code Audit

Pixel Race Club Safari Racers

Project Dashboard

Application Summary

Name	Pixel Race Club
Commit	7e07b86d5de0179e1015811dc11bc3efa2e56692
Repository	https://github.com/developmentgamepxu/prc-sr-public
Language	Solidity
Platform	Abstract

Engagement Summary

Timeline	29/04/2025
№ of Auditors	2
Review Time	0.4 person weeks

Vulnerability Summary

Issue Classification	Found	Addressed	Acknowledged
Critical	0	0	0
High	0	0	0
Medium	1	1	0
Low	2	2	0
None	6	4	2

Category Breakdown

Suggestion	6
Documentation	0
Bug	3
Optimization	0
Good Code Practices	0

Risk Section

Code Audit

Pixel Race Club Safari Racers

Risk Section

- **Coarse-grained Access Control**: The lack of granular roles and permissions significantly increases the potential harm that can result from a compromised privileged account.
- **Upgradability Risk**: Administrators have the capability to update contract logic at any moment due to the project's upgradable design. Contract upgradability, while beneficial, also poses the risk of harmful modifications if administrative privileges or upgrade processes are compromised.

Findings

Code Audit

Pixel Race Club Safari Racers

Findings

3S-Pixel Race Club-M01

Missing reentrancy lock in **`safeMintBatch`** and **`safeMintArray`** allows bypassing supply limit

Id	3S-Pixel Race Club-M01
Classification	Medium
Impact	High
Likelihood	Low
Category	Bug
Status	Addressed in #38f9e33.

Description

The **`PRCSR::safeMintBatch`** and **`PRCSR::safeMintArray`** functions are designed to mint multiple NFTs to a specified address. However, these functions lack a reentrancy lock, which can lead to the bypass of the supply limit.

The **`safeMintBatch`** function allows minting a batch of tokens to a user, and during this process, the **`_safeMint`** function is called. This function triggers the **`onERC721Received`** hook, which can be exploited by a malicious contract to re-enter the **`PRCSR`** contract. By doing so, the attacker can call other functions like **`PRCSR::publicMint`** or **`PRCSR::addToWL`** to mint additional tokens until the supply limit is reached. Then once the execution of **`safeMintBatch`** resumes, it keeps minting tokens effectively exceeding the supply limit, since the check for the supply limit is performed at the beginning of the **`safeMintBatch`** execution. The same issue is present in the **`safeMintArray`** function.

Recommendation

To prevent reentrancy attacks, apply the **`nonReentrant`** modifier to both **`PRCSR::safeMintBatch`** and **`PRCSR::safeMintArray`** functions. This will ensure that reentrant calls are blocked, maintaining the integrity of the supply limit checks.

3S-Pixel Race Club-L01

Missing zero-address checks in **PRCSR** contract functions

Id	3S-Pixel Race Club-L01
Classification	Low
Impact	Low
Likelihood	Low
Category	Bug
Status	Addressed in #0ec7b3a.

Description

The **PRCSR** contract contains multiple functions that accept address parameters without validating against the zero address (**`address(0)`**). This vulnerability exists across several key areas:

1. **Proxy Initialization**

   - **`UUPSProxy`** constructor allows deployment with zero implementation address.

2. **Role Assignment**

   - **`initialize()`** function accepts zero addresses for **`defaultAdmin`**, **`minter`**, and **`upgrader`**.

3. **Mint Operations**

   - **`safeMint()`**, **`safeMintBatch()`**, and **`publicMint()`** allow minting to **`address(0)`**.

Recommendation

Implement strict zero-address validation across all address parameters.

3S-Pixel Race Club-L02

Lack of refund mechanism in **`publicMint`** and **`wlMint`** functions

Id	3S-Pixel Race Club-L02
Classification	Low
Impact	Low
Likelihood	Medium
Category	Bug
Status	Addressed in #ae3f9c6.

Description

The **`PRCSR::publicMint`** and **`PRCSR::wlMint`** functions in the **`PRCSR`** contract are responsible for allowing users to mint NFTs during the public and whitelist sales, respectively. Both functions require users to send a payment equal to or greater than the price of the NFTs they wish to mint. However, neither function includes a mechanism to refund any excess payment beyond the required amount. This oversight can lead to users unintentionally overpaying and not receiving a refund for the excess amount.

In the **`PRCSR::publicMint`** function, the check **`require(msg.value >= price, "PRCSR: Invalid price");`** ensures that the user sends at least the required price, but does not handle the case where **`msg.value`** exceeds the price. Similarly, in the **`PRCSR::wlMint`** function, the check **`require(msg.value >= price * count, "PRCSR: Invalid price");`** ensures the user sends at least the total required price for the number of NFTs being minted, but again does not handle excess payment.

Recommendation

Implement a refund mechanism in both **`PRCSR::publicMint`** and **`PRCSR::wlMint`** functions to return any excess payment to the user. This can be achieved by calculating the excess amount and transferring it back to the sender after the minting process.

3S-Pixel Race Club-N01

Unbounded Loop Gas Consumption Risk in **`wlMint`** and **`safeMintArray`** Functions

Id	3S-Pixel Race Club-N01
Classification	None
Category	Suggestion
Status	Acknowledged

Description

The **`wlMint`** and **`safeMintArray`** functions contain unbounded loops that could consume excessive gas and potentially reach block gas limits.

`wlMint` performs a loop that executes **`count`** iterations without an upper bound on this parameter. This allows whitelisted users to mint an unlimited number of tokens in a single transaction, potentially exceeding the block gas limit. Since this function is accessible to any whitelisted user, it poses a risk of transaction failures that still consume gas fees.

`safeMintArray` similarly contains an unbounded loop that iterates through all addresses in the input array. While this function is restricted to the MINTER_ROLE, an administrator could still inadvertently trigger an out-of-gas error by providing an excessively large array of addresses.

Recommendation

Implement reasonable upper bounds on loop iterations in both functions.

3S-Pixel Race Club-N02

Incorrect token ID range in **`setBaseURI`**

Id	3S-Pixel Race Club-N02
Classification	None
Category	Suggestion
Status	Addressed in #b7d5df4.

Description

The **`PRCSR::setBaseURI`** function in the **`PRCSR`** contract is responsible for updating the base URI for all tokens in the collection. This function emits a **`BatchMetadataUpdate`** event to signal that the metadata for a range of tokens has been updated. However, the current implementation of the function emits the event with a starting token ID of 1, while the tokens in the collection start at ID 0. This results in the metadata update event not covering token ID 0, potentially leading to inconsistencies in metadata handling for this token.

Recommendation

To ensure that the metadata update event covers all tokens, including token ID 0, modify the starting token ID in the **`BatchMetadataUpdate`** event emission to 0.

3S-Pixel Race Club-N03

Typo in parameter name in **`withdraw`** function

Id	3S-Pixel Race Club-N03
Classification	None
Category	Suggestion
Status	Addressed in #d645375.

Description

In the **`PRCSR::withdraw`** function, there is a typographical error in the parameter name:

```
function withdraw(uint256 amount, address reciever) external override
onlyRole(DEFAULT_ADMIN_ROLE) {
```

The parameter name **`reciever`** is misspelled and should be **`receiver`** (with two 'e's). While this does not affect contract functionality since parameter names are only used within the contract scope, it could lead to confusion in documentation, comments, or for developers working with the codebase.

Recommendation

Correct the parameter name to use the standard English spelling.

3S-Pixel Race Club-N04

Coarse grained access control

Id	3S-Pixel Race Club-N04
Classification	None
Category	Suggestion
Status	Acknowledged

Description

The **PRCSR** contract is an upgradeable ERC721 token contract that includes functionality for whitelist and public sales. The **DEFAULT_ADMIN_ROLE** is currently used for administrative operations such as adding users to the whitelist, adjusting NFT price, and setting sale start blocks. This role also has the authority to upgrade the contract. If the **DEFAULT_ADMIN_ROLE** is compromised, an attacker could upgrade the contract to maliciously transfer all NFTs, leading to a significant security risk. The use of **DEFAULT_ADMIN_ROLE** for administrative tasks increases the likelihood and impact of potential compromise.

Recommendation

To mitigate this risk, the **DEFAULT_ADMIN_ROLE** should be restricted to assigning roles only. Separate roles should be created and used for administrative operations to minimize the impact of a compromised admin account. Ensure that only the **UPGRADER_ROLE** can authorize contract upgrades, and that its only role admin is **DEFAULT_ADMIN_ROLE**.

3S-Pixel Race Club-N05

Misleading function name and description in **`getTotalMintedNFTs`**

Id	3S-Pixel Race Club-N05
Classification	None
Category	Suggestion
Status	Addressed in #736370f.

Description

The function **`IPRCSR::getTotalMintedNFTs`** is intended to return the total number of NFTs minted by a specific user. However, the current implementation returns the balance of NFTs held by the user, not the number of NFTs minted by them. This discrepancy between the function's name, its NatSpec description in the interface, and its actual behavior can lead to confusion and incorrect assumptions about the data being returned.

Recommendation

It's suggested to rename the function to reflect its actual behavior or update its logic to match the intended functionality. To accurately track the number of NFTs minted by each address, implement a mapping to store this data and update it within the mint functions.

3S-Pixel Race Club-N06

Missing validation in sale start time configuration

Id	3S-Pixel Race Club-N06
Classification	None
Category	Suggestion
Status	Addressed in #7f25060.

Description

The contract **`PRCSR`** implements a whitelist and public sale mechanism for NFT minting, where whitelist sale should start before public sale. The contract exposes three functions to configure these timings: **`initialize`**, **`setWLStart`**, and **`setPublicSaleStart`**. However, none of these functions validate that **`wlStart`** is less than **`publicSaleStart`**, which could lead to incorrect sale timing configuration.

Recommendation

Add validation to ensure **`wlStart`** is always less than **`publicSaleStart`** in the three mentioned functions.