Exiled Racers

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 11th, 2022 - April 22nd, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/11/2022	Roberto Reigada
0.2	Document Updates	04/22/2022	Roberto Reigada
0.3	Draft Review	04/22/2022	Gabi Urrutia
1.0	Remediation Plan	04/28/2022	Roberto Reigada
1.1	Remediation Plan Review	04/28/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Roberto Reigada	Halborn	Roberto.Reigada@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Exiled Racers engaged Halborn to conduct a security audit on their smart contracts beginning on April 11th, 2022 and ending on April 22nd, 2022. The security assessment was scoped to the smart contracts provided in the GitHub repository Sokoke-Labs/exr-contracts-public.

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contracts. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some security risks that were addressed by the Exiled Racers team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

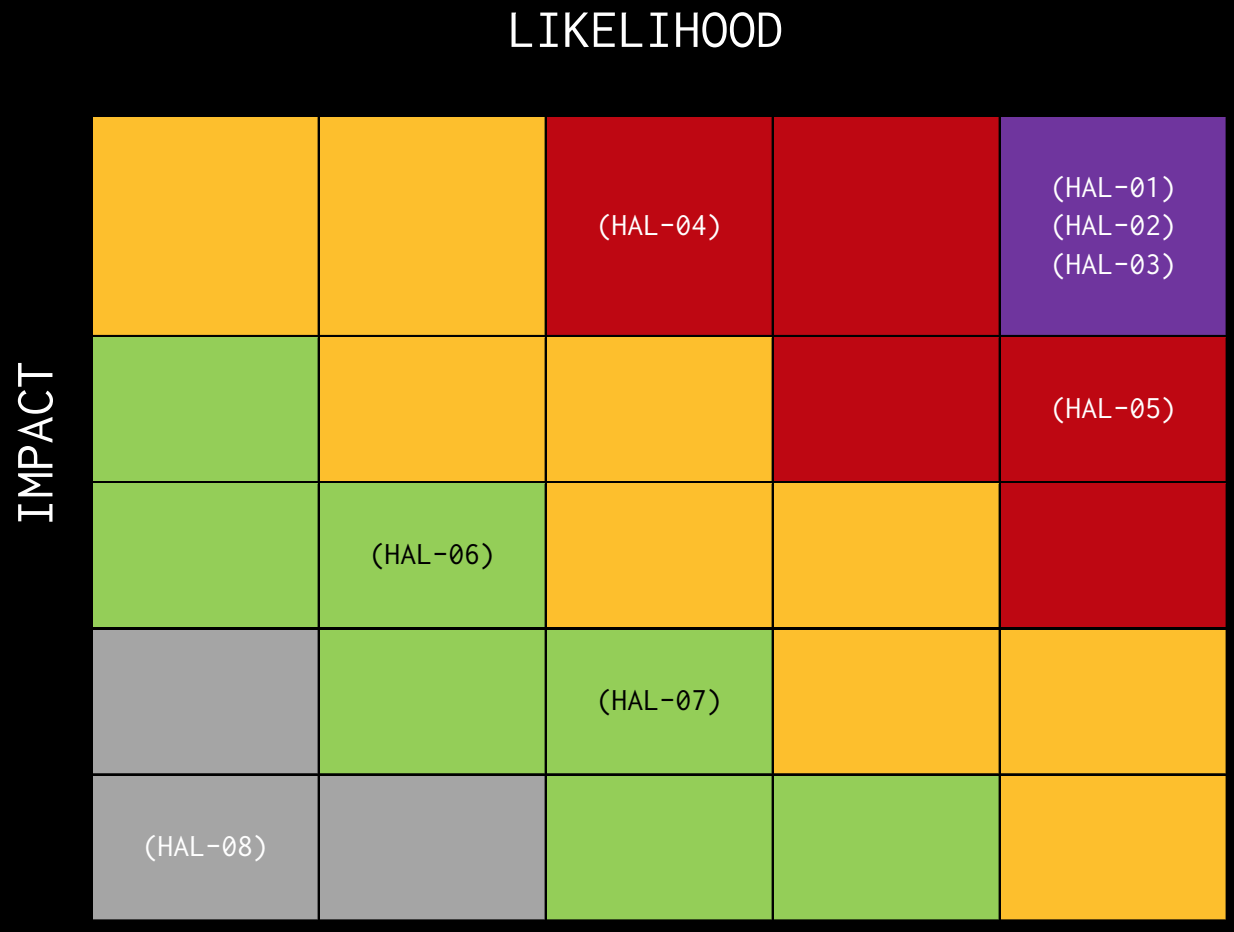
- EXRGameAssetERC721.sol
- EXRInventoryController.sol
- EXRInventoryERC1155.sol
- EXRMintPassERC1155.sol
- EXRSalesContract.sol

Initial Commit ID:
7d04aa3277631539cd5fa2acb4c4eaa388aefea1 - Private Repository

Fixed Commit ID: 55b75c192fb18a87b385f448b727b2d37fc64259

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
3	2	0	2	1



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - CLAIMREWARDITEMS AND BURNTOREDEEMINVENTORYITEMS FUNCTIONS CAN BE FRONTRUN	Critical	SOLVED - 04/28/2022
HAL02 - WEAK PRNG IN CLAIMRANDOMITEMS: USERS CAN ALWAYS REDEEM THE RAREST CATEGORY AND RARIRY	Critical	SOLVED - 04/28/2022
HAL03 - WEAK PRNG IN REDEEMPILOT AND REDEEMRACECRAFT FUNCTIONS	Critical	SOLVED - 04/28/2022
HAL04 - MULTIPLE FUNCTIONS ARE VULNERABLE TO REPLAY ATTACKS	High	SOLVED - 04/28/2022
HAL05 - SAME SEED AND SIGNATURE CAN BE REUSED IN REDEEMPILOT AND REDEEMRACECRAFT FUNCTIONS	High	SOLVED - 04/28/2022
HAL06 - ERC721FRAGMENTABLE MAX SUPPLY CAN BE EXCEEDED BY 1 IF THE FIRSTID OF FRAGMENT 0 IS 0	Low	SOLVED - 04/28/2022
HAL07 - INCORRECT EVENT EMISSION	Low	SOLVED - 04/28/2022
HAL08 - UNUSED EVENT	Informational	SOLVED - 04/28/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) CLAIMREWARDITEMS AND BURNTOREDEEMINVENTORYITEMS FUNCTIONS CAN BE FRONTRUN - CRITICAL

Description:

The `claimRewardItems()` and `burnToRedeemInventoryItems()` functions are used to claim different inventory items from the `EXRInventoryController` contract:

Listing 1: EXRInventoryController.sol (Line 89)

```
81 function claimRewardItems(
82     bytes32 seed,
83     uint256 qty,
84     Coupon calldata coupon
85 ) external whenNotPaused nonReentrant {
86     if (usedSeeds[seed]) revert InventoryReusedSeed();
87
88     usedSeeds[seed] = true;
89     bytes32 digest = keccak256(abi.encode(CouponType.Reward, qty,
   ↳ seed));
90     if (!_verifyCoupon(digest, coupon)) revert
   ↳ InventoryInvalidCoupon();
91
92     _claimRandomItems(seed, qty);
93     emit InventoryRewardClaimed(_msgSender(), qty);
94 }
```

Listing 2: EXRInventoryController.sol (Line 113)

```
103 function burnToRedeemInventoryItems(
104     bytes32 seed,
105     uint256 qty,
106     Coupon calldata coupon
107 ) external whenNotPaused nonReentrant {
108     if (mintpassContract.balanceOf(_msgSender(), inventoryPassId)
   ↳ == 0)
```

```
109             revert InventoryInsufficientPassBalance();
110         if (usedSeeds[seed]) revert InventoryReusedSeed();
111
112         usedSeeds[seed] = true;
113         bytes32 digest = keccak256(abi.encode(CouponType.Inventory,
 ↳ qty, seed));
114         if (!_verifyCoupon(digest, coupon)) revert
 ↳ InventoryInvalidCoupon();
115
116         mintpassContract.authorizedBurn(_msgSender(), inventoryPassId)
 ↳ ;
117         _claimRandomItems(seed, qty);
118     }
```

The digest is formed by:
- CouponType.Inventory
- qty
- seed

Since the digest is not linked to the caller's address (msg.sender), the following exploit would be possible:

1. The signer creates a signature and shares it with user1.
2. User1 calls claimRewardItems().
3. User1's transaction hits the public mempool.
4. The attacker detects the claimRewardItems() call in the mempool.
5. The attacker extracts the transaction signature of user1 in the mempool, and calls the claimRewardItems() functions front running user1.
6. The attacker manages to claim the items.
7. The User1 lost his claim. Since his transaction was front run, he gets the error InventoryReusedSeed.

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to add the address of the eligible user to claim the Coupon in the digest:

```
bytes32 digest = keccak256(abi.encode(msg.sender, CouponType.Inventory,
 qty, seed));
```

Remediation Plan:

SOLVED: The Exiled Racers team solved the issue by adding the address of the user eligible to claim the Coupon in the digest:

Listing 3: EXRInventoryController.sol (Lines 90-92)

```
82 function claimRewardItems(
83     bytes32 seed,
84     uint256 qty,
85     Coupon calldata coupon
86 ) external whenNotPaused nonReentrant hasValidOrigin {
87     if (usedSeeds[seed]) revert InventoryReusedSeed();
88
89     usedSeeds[seed] = true;
90     bytes32 digest = keccak256(
91         abi.encode(address(this), block.chainid, CouponType.Reward
↳, qty, seed, _msgSender())
92     );
93     if (!_verifyCoupon(digest, coupon)) revert
↳ InventoryInvalidCoupon();
94
95     _claimRandomItems(seed, qty);
96     emit InventoryRewardClaimed(_msgSender(), qty);
97 }
```

3.2 (HAL-02) WEAK PRNG IN CLAIMRANDOMITEMS: USERS CAN ALWAYS REDEEM THE RAREST CATEGORY AND RARIRY - CRITICAL

Description:

In the EXRInventoryController contract, the _claimRandomItems() internal function is called by claimRewardItems() and burnToRedeemInventoryItems():

Listing 4: EXRInventoryController.sol (Lines 232-234)

```
226 function _claimRandomItems(bytes32 seed, uint256 amount) internal
 ↳ {
227     uint256[] memory ids = new uint256[](amount);
228     uint256[] memory amounts = new uint256[](amount);
229
230     for (uint256 i; i < amount; i++) {
231         // Every category has an equal chance of being selected
232         uint256 randomCategorySelector = (uint256(
233             keccak256(abi.encode(_msgSender(), block.number - 1,
 ↳ seed, i))
234         ) % (categories.length * 100)) + 1;
235
236         uint256 id;
237         for (uint256 ii; ii < categories.length; ii++) {
238             if (randomCategorySelector < (ii + 1) * 100) {
239                 id = _selectIdByRarity(randomCategorySelector, ii)
 ↳ ;
240                 break;
241             }
242         }
243
244         ids[i] = id;
245         amounts[i] = 1;
246     }
247
248     inventoryContract.mintBatch(_msgSender(), ids, amounts, "");
```

```
249     emit InventoryItemsClaimed(ids, amounts);
250 }
```

Listing 5: EXRInventoryController.sol

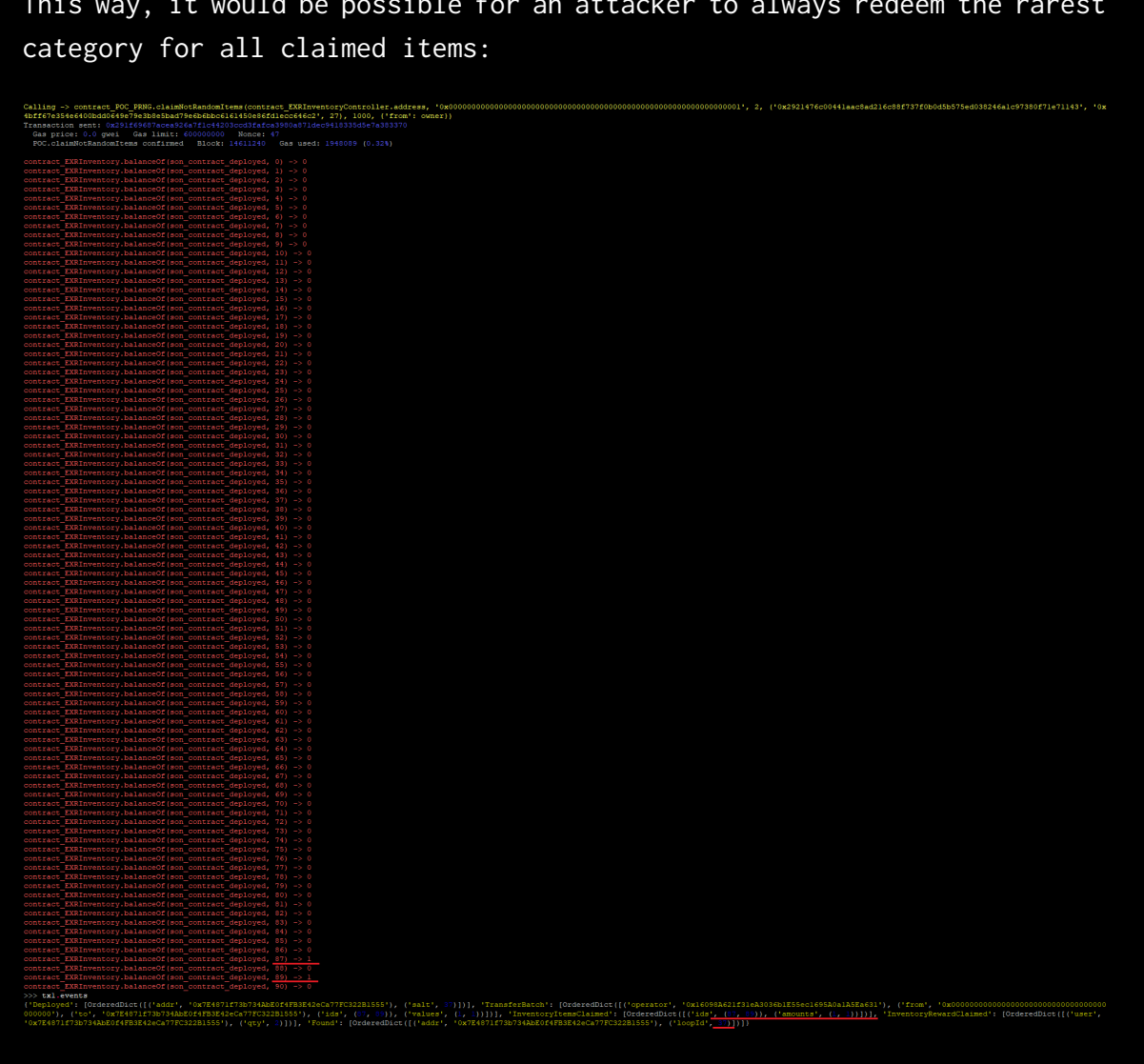
```
259 function _selectIdByRarity(uint256 seed, uint256 category)
 ↳ internal view returns (uint256) {
260     uint256 randomIdSelector = (uint256(keccak256(abi.encode(seed,
 ↳  category))) % 3000) + 1;
261     uint8[9] memory options = categories[category].tokenIds;
262 
263     if (randomIdSelector > 2500) {
264         return options[0]; // common ( 2500 - 3000)
265     } else if (randomIdSelector > 2000) {
266         return options[1]; // common (2000 - 2500)
267     } else if (randomIdSelector > 1500) {
268         return options[2]; // common ( 1500 - 2000)
269     } else if (randomIdSelector > 1150) {
270         return options[3]; // mid (1150 - 1500)
271     } else if (randomIdSelector > 800) {
272         return options[4]; // mid (800 - 1150)
273     } else if (randomIdSelector > 450) {
274         return options[5]; // mid ( 450 - 800)
275     } else if (randomIdSelector > 300) {
276         return options[6]; // rare ( 300 - 450 )
277     } else if (randomIdSelector > 150) {
278         return options[7]; // rare ( 150 - 300)
279     } else {
280         return options[8]; // rare ( 0 - 150)
281     }
282 }
```

The `_claimRandomItems()` is used to generate “random” token IDs. With the current implementation, an attacker could:

1. Create a malicious parent contract.
2. Precompute different contract addresses with different salts until finding a contract address that results in the rarest category: `uint256 randomCategorySelector = (uint256(keccak256(abi.encode( _msgSender(), block.number - 1, seed, i)))% (categories.length * 100))+ 1;`

3. Deploy the precomputed address using CREATE2.
4. Call the claimRewardItems() from the deployed child contract.

This way, it would be possible for an attacker to always redeem the rarest category for all claimed items:



Proof of Concept:

In the Proof of Concept below, we managed to claim 2 items, both with the rarest category, after precomputing 37 different addresses (37 iterations):

Exiled Racers - PRNG Exploitation Video

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to disallow calls from smart contracts to `claimRewardItems()` and `burnToRedeemInventoryItems()` functions by adding the following require statement:
`require(tx.origin == msg.sender);`

On the other hand, it is also recommended to remove `msg.sender` from the PRNG generation.

Remediation Plan:

SOLVED: The Exiled Racers team solved the issue by adding the `hasValidOrigin` modifier to the `claimRewardItems()` and `burnToRedeemInventoryItems()` functions:

Listing 6: EXRInventoryController.sol

```
303 modifier hasValidOrigin() {
304     require(
305         isTrustedForwarder(msg.sender) || msg.sender == tx.origin,
306         "Non-trusted forwarder contract not allowed"
307     );
308     _;
309 }
```

This modifier will restrict smart contract calls to these functions. On the other hand, `msg.sender` was removed from the PRNG generation:

Listing 7: EXRInventoryController.sol (Lines 246-248)

```
240 function _claimRandomItems(bytes32 seed, uint256 amount) internal
 ↳ {
241     uint256[] memory ids = new uint256[](amount);
```

```
242     uint256[] memory amounts = new uint256[](amount);
243
244     for (uint256 i; i < amount; i++) {
245         // Every category has an equal chance of being selected
246         uint256 randomCategorySelector = (uint256(
247             keccak256(abi.encode(seed, blockhash(block.number),
 ↳ block.basefee, i))
248         ) % (categories.length * 100)) + 1;
249
250         uint256 id;
251         for (uint256 ii; ii < categories.length; ii++) {
252             if (randomCategorySelector < (ii + 1) * 100) {
253                 id = _selectIdByRarity(randomCategorySelector, ii)
 ↳ ;
254                 break;
255             }
256         }
257
258         ids[i] = id;
259         amounts[i] = 1;
260     }
261
262     inventoryContract.mintBatch(_msgSender(), ids, amounts, "");
263     emit InventoryItemsClaimed(ids, amounts);
264 }
```

3.3 (HAL-03) WEAK PRNG IN REDEEMPILOT AND REDEEMRACECRAFT FUNCTIONS - CRITICAL

Description:

In the EXRSalesContract contract, the redeemPilot() and redeemRacecraft() functions are used to redeem a pilot or racecraft. These functions internally call the EXRGameAsset.mint() function:

Listing 8: EXRGameAsset.sol (Line 77)

```
62 function mint(
63     address recipient,
64     uint256 count,
65     uint8 fragment,
66     bytes32 seed
67 ) external whenNotPaused onlyRole(MINTER_ROLE) {
68     if (recipient == address(0)) revert GameAssetZeroAddress();
69     if (count == 0) revert GameAssetZeroCount();
70
71     Fragment memory assetFragment = fragments[fragment];
72     if (assetFragment.status != 1) revert GameAssetInvalidFragment
↳ (fragment);
73     if (assetFragment.publicTokens.issuedCount + count >
↳ assetFragment.publicTokens.supply)
74         revert GameAssetFragmentTokenPoolSupplyExceeded();
75
76     for (uint256 i; i < count; i++) {
77         uint256 tokenId = issueRandomId(fragment, seed, recipient)
↳ ;
78
79         if (tokenId < assetFragment.firstTokenId + assetFragment.
↳ reservedTokens.supply)
80             revert GameAssetTokenIdReserved({tokenId: tokenId});
81
82         if (tokenId > assetFragment.firstTokenId + assetFragment.
↳ supply - 1)
83             revert GameAssetInvalidFragmentTokenId();
84
85         _createAndMintGameAsset(recipient, tokenId, fragment);
```

```
86      }
87  }
```

As we can see above, the mint function makes use of `issueRandomId()` to choose the tokenID that will be minted:

Listing 9: ERC721Fragmentable.sol (Lines 236-238)

```
223 function issueRandomId(
224     uint256 fragment,
225     bytes32 seed,
226     address recipient
227 ) internal returns (uint256) {
228     Fragment storage currentFragment = fragments[fragment];
229 
230     uint256 remaining = currentFragment.publicTokens.supply -
231         currentFragment.publicTokens.issuedCount;
232     if (remaining == 0) revert TokenPoolEmpty();
233 
234     // returns a random number between 0 and the number of tokens
 ↳ remaining - 1, this will be
235     // used as the random ID if the slot in the matrix
 ↳ corresponding to the index is empty
236     uint256 randomIndex = uint256(
237         keccak256(abi.encodePacked(recipient, blockhash(block.
 ↳ number - 1), seed))
238     ) % remaining;
239 
240     // If the matrix is empty at the given random index (slot), we
 ↳  use the index as the token ID.
241     // However, if the slot contains an ID, we'll assign that
 ↳ instead.
242 
243     uint256 offset = fragmentPoolTokenMatrix[fragment][randomIndex
 ↳ ] == 0
244         ? randomIndex
245         : fragmentPoolTokenMatrix[fragment][randomIndex];
246 
247     currentFragment.publicTokens.issuedCount++;
248 
249     uint256 temp = fragmentPoolTokenMatrix[fragment][remaining -
 ↳ 1];
250 
```

```
251     if (temp == 0) {
252         fragmentPoolTokenMatrix[fragment][randomIndex] = remaining
 ↳  - 1;
253     } else {
254         fragmentPoolTokenMatrix[fragment][randomIndex] = temp;
255         delete fragmentPoolTokenMatrix[fragment][remaining - 1];
 ↳ // small gas refund
256     }
257
258     uint256 tokenId = currentFragment.publicTokens.startId +
 ↳ offset;
259     return tokenId;
260 }
```

The way the random number is generated, as in the HAL-02 issue, allows an attacker to, for example:

1. Create a malicious parent contract.
2. Send this contract to the Pilot MintPass.
3. Precompute different contract addresses with different salts until you find a contract address that results in the desired nftID: `uint256 randomIndex = uint256(keccak256(abi.encodePacked(recipient, blockhash(block.number - 1), seed)))% remaining;`
4. Deploy the precomputed address using CREATE2.
5. Send the Pilot MintPass from the parent contract to the child contract.
6. Call the `redeemPilot()` from the deployed child contract.

This way, it would be possible for an attacker to always redeem the Pilot with the desired NFTid. The same issue also applies to the `redeemRacecraft()` function.

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to disallow smart contracts calls to redeemPilot() and redeemRacecraft() functions by adding the following require statement: require(tx.origin == msg.sender);

On the other hand, it is also recommended to remove the recipient address from the PRNG generation.

Remediation Plan:

SOLVED: The Exiled Racers team solved the issue by adding the hasValidOrigin modifier to the redeemPilot() and redeemRacecraft() functions:

Listing 10: EXRSalesContract.sol

```
423 modifier hasValidOrigin() {
424     require(
425         isTrustedForwarder(msg.sender) || msg.sender == tx.origin,
426         "Non-trusted forwarder contract not allowed"
427     );
428     _;
429 }
```

This modifier will restrict smart contract calls to these functions. On the other hand, recipient was removed from the PRNG generation:

Listing 11: ERC721Fragmentable.sol (Lines 237-239)

```
224 function issueRandomId(
225     uint256 fragment,
226     bytes32 seed,
227     address recipient
228 ) internal returns (uint256) {
229     Fragment storage currentFragment = fragments[fragment];
230
231     uint256 remaining = currentFragment.publicTokens.supply -
232         currentFragment.publicTokens.issuedCount;
233     if (remaining == 0) revert TokenPoolEmpty();
234
```

```
235     // returns a random number between 0 and the number of tokens
 ↳ remaining - 1, this will be
236     // used as the random ID if the slot in the matrix
 ↳ corresponding to the index is empty
237     uint256 randomIndex = uint256(
238         keccak256(abi.encodePacked(block.basefee, blockhash(block.
 ↳ number - 1), seed))
239     ) % remaining;
240 
241     // If the matrix is empty at the given random index (slot), we
 ↳  use the index as the token ID.
242     // However, if the slot contains an ID, we'll assign that
 ↳ instead.
243 
244     uint256 offset = fragmentPoolTokenMatrix[fragment][randomIndex
 ↳ ] == 0
245         ? randomIndex
246         : fragmentPoolTokenMatrix[fragment][randomIndex];
247 
248     currentFragment.publicTokens.issuedCount++;
249 
250     uint256 temp = fragmentPoolTokenMatrix[fragment][remaining -
 ↳ 1];
251 
252     if (temp == 0) {
253         fragmentPoolTokenMatrix[fragment][randomIndex] = remaining
 ↳  - 1;
254     } else {
255         fragmentPoolTokenMatrix[fragment][randomIndex] = temp;
256         delete fragmentPoolTokenMatrix[fragment][remaining - 1];
 ↳ // small gas refund
257     }
258 
259     uint256 tokenId = currentFragment.publicTokens.startId +
 ↳ offset;
260     return tokenId;
261 }
```

3.4 (HAL-04) MULTIPLE FUNCTIONS ARE VULNERABLE TO REPLAY ATTACKS - HIGH

Description:

The claimRewardItems() and burnToRedeemInventoryItems() functions are used to claim different inventory items from the EXRInventory contract:

Listing 12: EXRInventoryController.sol (Line 89)

```
81 function claimRewardItems(
82     bytes32 seed,
83     uint256 qty,
84     Coupon calldata coupon
85 ) external whenNotPaused nonReentrant {
86     if (usedSeeds[seed]) revert InventoryReusedSeed();
87
88     usedSeeds[seed] = true;
89     bytes32 digest = keccak256(abi.encode(CouponType.Reward, qty,
   ↳ seed));
90     if (!_verifyCoupon(digest, coupon)) revert
   ↳ InventoryInvalidCoupon();
91
92     _claimRandomItems(seed, qty);
93     emit InventoryRewardClaimed(_msgSender(), qty);
94 }
```

Listing 13: EXRInventoryController.sol (Line 113)

```
103 function burnToRedeemInventoryItems(
104     bytes32 seed,
105     uint256 qty,
106     Coupon calldata coupon
107 ) external whenNotPaused nonReentrant {
108     if (mintpassContract.balanceOf(_msgSender(), inventoryPassId)
    ↳ == 0)
109         revert InventoryInsufficientPassBalance();
110     if (usedSeeds[seed]) revert InventoryReusedSeed();
111
```

```
112     usedSeeds[seed] = true;
113     bytes32 digest = keccak256(abi.encode(CouponType.Inventory,
 ↳ qty, seed));
114     if (!_verifyCoupon(digest, coupon)) revert
 ↳ InventoryInvalidCoupon();
115
116     mintpassContract.authorizedBurn(_msgSender(), inventoryPassId)
 ↳ ;
117     _claimRandomItems(seed, qty);
118 }
```

The digest is formed by:
- CouponType.Inventory
- qty
- seed

This does not follow the EIP 712 and it is vulnerable to multiple signature replay attacks. For example:

1. The EXRInventoryController contract is deployed on the testnet.
2. The signer with address '0x0000000000000000000000000000000000001337' creates different Coupons.
3. Various users use these coupons to call the claimRewardItems() function.
4. One month later, after the testing phase is completed, the same contract is deployed in the mainnet.
5. The same signer is set in the mainnet.
6. An attacker could collect the signatures used in the testnet and now reuse them in the mainnet contract.

NOTE:
This attack can also be done with contracts on the same network, since the digest does not contain the address of the contract. The same issue occurs in EXRSalesContract with the redeemPilot() and redeemRacecraft() functions:

Functions Affected:

- EXRInventoryController.claimRewardItems()
- EXRInventoryController.burnToRedeemInventoryItems()
- EXRSalesContract.redeemPilot()
- EXRSalesContract.redeemRacecraft()

Risk Level:

Likelihood - 3
Impact - 5

Recommendation:

It is recommended to implement the EIP 712 in the EXRInventoryController contract. For example:

Listing 14

```
bytes memory prefix = "\x19Ethereum Signed Message:\n32";
bytes32 messageHash = keccak256(abi.encode(address(this), block.
↳ chainid, CouponType.Reward, qty, seed));
bytes32 digest = keccak256(abi.encodePacked(prefix, messageHash));
if (!_verifyCoupon(digest, coupon)) revert InventoryInvalidCoupon
↳ ();
```

Remediation Plan:

SOLVED: The Exiled Racers team solved the issue by adding the smart contract address and the chainID in the digest:

Listing 15: EXRInventoryController.sol (Lines 90-92)

```
82 function claimRewardItems(
83     bytes32 seed,
84     uint256 qty,
85     Coupon calldata coupon
86 ) external whenNotPaused nonReentrant hasValidOrigin {
87     if (usedSeeds[seed]) revert InventoryReusedSeed();
```

```
88
89      usedSeeds[seed] = true;
90      bytes32 digest = keccak256(
91          abi.encode(address(this), block.chainid, CouponType.Reward
↳ , qty, seed, _msgSender())
92      );
93      if (!_verifyCoupon(digest, coupon)) revert
↳ InventoryInvalidCoupon();
94
95      _claimRandomItems(seed, qty);
96      emit InventoryRewardClaimed(_msgSender(), qty);
97  }
```

3.5 (HAL-05) SAME SEED AND SIGNATURE CAN BE REUSED IN REDEEMPILOT AND REDEEMRACECRAFT FUNCTIONS - HIGH

Description:

In the EXRSalesContract contract, the redeemPilot() and redeemRacecraft() functions do not check that the provided seed has not already been used:

Listing 16: EXRSalesContract.sol

```
188 function redeemPilot(bytes32 seed, Coupon calldata coupon)
 ↳ external nonReentrant {
189     if (state.redeemPilot == 0) revert
 ↳ SalesPilotRedemptionNotActive();
190     if (!pilotContract.fragmentExists(dedicatedFragment)) revert
 ↳ SalesNonExistentFragment();
191     if (pilotPassMaxSupply == 0) revert SalesMintpassQtyNotSet();
192
193     address caller = _msgSender();
194     if (mintPassContract.balanceOf(caller, pilotPassTokenId) == 0)
 ↳  revert SalesNoMintPass();
195
196     bytes32 digest = keccak256(abi.encode(CouponType.RandomSeed,
 ↳ seed));
197     if (!_verifyCoupon(digest, coupon)) revert SalesInvalidCoupon
 ↳ ();
198
199     mintPassContract.burnToRedeemPilot(caller, dedicatedFragment);
200     emit MintPassBurned(caller);
201
202     pilotContract.mint(caller, 1, dedicatedFragment, seed);
203     emit PilotRedeemed();
204 }
```

Listing 17: EXRSalesContract.sol

```
213 function redeemRacecraft(bytes32 seed, Coupon calldata coupon)
 ↳ external nonReentrant {
```

```
214     if (state.redeemRacecraft == 0) revert
 ↳ SalesRacecraftRedemptionNotActive();
215     if (!racecraftContract.fragmentExists(dedicatedFragment))
216         revert SalesNonExistentFragment();
217     if (racecraftPassMaxSupply == 0) revert SalesMintpassQtyNotSet
 ↳ ();
218
219     address caller = _msgSender();
220     if (mintPassContract.balanceOf(caller, racecraftPassTokenId)
 ↳ == 0)
221         revert SalesNoMintPass();
222
223     bytes32 digest = keccak256(abi.encode(CouponType.RandomSeed,
 ↳ seed));
224     if (!_verifyCoupon(digest, coupon)) revert SalesInvalidCoupon
 ↳ ();
225
226     mintPassContract.authorizedBurn(caller, racecraftPassTokenId);
227     racecraftContract.mint(caller, 1, dedicatedFragment, seed);
228     emit RacecraftRedeemed();
229 }
```

This allows any user to constantly reuse the same Coupon signature over and over again, and totally defeats the purpose of using signatures in the first place.

Risk Level:

Likelihood - 5
Impact - 4

Recommendation:

It is recommended to check that the seed has not been previously used in the smart contract, but this would introduce the issue mentioned in HAL-01. So once, the seed check is added, the same fix that was applied in HAL-01 needs to be implemented here, moreover to avoid the front running.

Remediation Plan:

SOLVED: The Exiled Racers team now checks in the redeemPilot() and redeemRacecraft() functions that the seed was not previously used:

Listing 18: EXRInventoryController.sol (Line 202)

```
194 function redeemPilot(bytes32 seed, Coupon calldata coupon)
195     external
196     nonReentrant
197     hasValidOrigin
198 {
199     if (state.redeemPilot == 0) revert
 ↳ SalesPilotRedemptionNotActive();
200     if (!pilotContract.fragmentExists(dedicatedFragment)) revert
 ↳ SalesNonExistentFragment();
201     if (pilotPassMaxSupply == 0) revert SalesMintpassQtyNotSet();
202     if (usedSeeds[seed]) revert SalesReusedSeed();
203
204     usedSeeds[seed] = true;
205
206     address caller = _msgSender();
207     if (mintPassContract.balanceOf(caller, pilotPassTokenId) == 0)
 ↳  revert SalesNoMintPass();
208
209     bytes32 digest = keccak256(
210         abi.encode(address(this), block.chainid, CouponType.Pilot,
 ↳  seed, caller)
211     );
212     if (!_verifyCoupon(digest, coupon)) revert SalesInvalidCoupon
 ↳ ();
213
214     mintPassContract.burnToRedeemPilot(caller, dedicatedFragment);
215     emit MintPassBurned(caller);
216
217     pilotContract.mint(caller, 1, dedicatedFragment, seed);
218     emit PilotRedeemed();
219 }
```

On the other hand, front running will not be possible as the Coupon digest includes the caller's address.

3.6 (HAL-06) ERC721FRAGMENTABLE MAX SUPPLY CAN BE EXCEEDED BY 1 IF THE FIRSTID OF FRAGMENT 0 IS 0 - LOW

Description:

In the ERC721Fragmentable contract, the createFragment() function allows you to create a new fragment in the collection:

Listing 19: ERC721Fragmentable.sol (Line 165)

```
158 function createFragment(
159     uint8 id,
160     uint64 fragmentSupply,
161     uint64 firstId,
162     uint64 reserved
163 ) external onlyRole(FRAGMENT_CREATOR_ROLE) {
164     if (fragmentSupply <= 1) revert FragmentInvalidSupply();
165     if (firstId + fragmentSupply - 1 > MAX_SUPPLY) revert
 ↳ FragmentExceedsCollectionSupply();
166     if (reserved > fragmentSupply) revert
 ↳ FragmentTokenSupplyExceeded();
167     if (fragments[id].status == 1) revert FragmentExists();
168
169     if (id > 0) {
170         Fragment memory previousFragment = fragments[id - 1];
171         if (id != previousFragment.fragmentId + 1) revert
 ↳ FragmentNotSequential();
172         if (firstId != previousFragment.firstTokenId +
 ↳ previousFragment.supply)
173             revert FragmentsTokenIdsNotSequential();
174     }
175     fragmentCount++;
176     fragments[id] = Fragment({
177         status: 1,
178         locked: 0,
179         fragmentId: id,
180         firstTokenId: firstId,
181         supply: fragmentSupply,
182         baseURI: "",
183         renderer: IRenderer(address(0)),
```

```
184         publicTokens: TokenPool({
185             issuedCount: 0,
186             startId: firstId + reserved,
187             supply: fragmentSupply - reserved
188         }),
189         reservedTokens: reserved > 0
190             ? TokenPool({issuedCount: 0, startId: firstId, supply:
 ↳  reserved})
191             : TokenPool(0, 0, 0)
192     });
193
194     emit FragmentCreated(id, fragmentSupply);
195 }
```

On the other hand, the contract has a hard-coded Maximum Supply of 8000 NFTs:

Listing 20: ERC721Fragmentable.sol

```
67 uint256 public constant MAX_SUPPLY = 8000;
```

However, this Maximum Supply value can be exceeded by 1 in case the `firstId` of the fragment 0 is 0. In total, the collection would have 8001 NFTs. The issue is due to the following check:

Listing 21: ERC721Fragmentable.sol

```
165 if (firstId + fragmentSupply - 1 > MAX_SUPPLY) revert
 ↳ FragmentExceedsCollectionSupply();`
```

Proof of Concept:

In the proof of concept below, we create 4 different fragments:
Fragment 0:
- fragmentSupply = 1500
- firstId = 0
- reservedPilots = 20
- reservedRacecrafts = 20

Fragment 1:
- fragmentSupply = 1500
- firstId = 1500
- reservedPilots = 20
- reservedRacecrafts = 20

Fragment 2:
- fragmentSupply = 1500
- firstId = 3000
- reservedPilots = 20
- reservedRacecrafts = 20

Fragment 3:
- fragmentSupply = 3501
- firstId = 4500
- reservedPilots = 20
- reservedRacecrafts = 20

```
Calling -> contract_EXRGameAsset_Pilot.createFragment(0, 1500, 0, 20, {'from': contract_EXRSalesContract1})
Transaction sent: 0x15ce78eda00dc9126dadc71e40e75a7ac7bf22ea6c2a1111bf979b0c22f1003a
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 1
  EXRGameAsset.createFragment confirmed   Block: 14628152   Gas used: 114481 (0.02%)

Calling -> contract_EXRGameAsset_Racecraft.createFragment(0, 1500, 0, 20, {'from': contract_EXRSalesContract1})
Transaction sent: 0xce2f5fd701a7a81280eb0eecd0ebcecf796701a9c9d699fe8ddaed2769ea9e33
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 2
  EXRGameAsset.createFragment confirmed   Block: 14628153   Gas used: 114481 (0.02%)

contract_EXRGameAsset_Pilot.fragments(0) -> (1, 0, 0, 0, 1500, '', '0x0000000000000000000000000000000000000000', (0, 0, 20), (0, 20, 1480))
contract_EXRGameAsset_Racecraft.fragments(0) -> (1, 0, 0, 0, 1500, '', '0x0000000000000000000000000000000000000000', (0, 0, 20), (0, 20, 1480))
Calling -> contract_EXRGameAsset_Pilot.createFragment(1, 1500, 1500, 20, {'from': contract_EXRSalesContract1})
Transaction sent: 0x470d7be5914a7fb4d74c9d9b1c4cca8abccd170fc265564e628b8424f3c43d6c
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 3
  EXRGameAsset.createFragment confirmed   Block: 14628154   Gas used: 105866 (0.02%)

Calling -> contract_EXRGameAsset_Racecraft.createFragment(1, 1500, 1500, 20, {'from': contract_EXRSalesContract1})
Transaction sent: 0x4ec7679e91b73850a6ad270e346b40ea9a982559f9217739b2eeb2f134a0668e
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 4
  EXRGameAsset.createFragment confirmed   Block: 14628155   Gas used: 105866 (0.02%)

contract_EXRGameAsset_Pilot.fragments(0) -> (1, 0, 0, 0, 1500, '', '0x0000000000000000000000000000000000000000', (0, 0, 20), (0, 20, 1480))
contract_EXRGameAsset_Racecraft.fragments(0) -> (1, 0, 0, 0, 1500, '', '0x0000000000000000000000000000000000000000', (0, 0, 20), (0, 20, 1480))
contract_EXRGameAsset_Pilot.fragments(1) -> (1, 0, 1, 1500, 1500, '', '0x0000000000000000000000000000000000000000', (0, 1500, 20), (0, 1520, 1480))
contract_EXRGameAsset_Racecraft.fragments(1) -> (1, 0, 1, 1500, 1500, '', '0x0000000000000000000000000000000000000000', (0, 1500, 20), (0, 1520, 1480))
Calling -> contract_EXRGameAsset_Pilot.createFragment(2, 1500, 3000, 20, {'from': contract_EXRSalesContract1})
Transaction sent: 0xd9200ef522b300b57009cd8e68b5c96788d5fa93b606db4f14ac2438d7fe836b
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 5
  EXRGameAsset.createFragment confirmed   Block: 14628156   Gas used: 105866 (0.02%)

Calling -> contract_EXRGameAsset_Racecraft.createFragment(2, 1500, 3000, 20, {'from': contract_EXRSalesContract1})
Transaction sent: 0x748f35e1e9e08d631e5bb9756b7346b4423ff4e1cb02f90b0b15e148f164ec22
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 6
  EXRGameAsset.createFragment confirmed   Block: 14628157   Gas used: 105866 (0.02%)

contract_EXRGameAsset_Pilot.fragments(0) -> (1, 0, 0, 0, 1500, '', '0x0000000000000000000000000000000000000000', (0, 0, 20), (0, 20, 1480))
contract_EXRGameAsset_Racecraft.fragments(0) -> (1, 0, 0, 0, 1500, '', '0x0000000000000000000000000000000000000000', (0, 0, 20), (0, 20, 1480))
contract_EXRGameAsset_Pilot.fragments(1) -> (1, 0, 1, 1500, 1500, '', '0x0000000000000000000000000000000000000000', (0, 1500, 20), (0, 1520, 1480))
contract_EXRGameAsset_Racecraft.fragments(1) -> (1, 0, 1, 1500, 1500, '', '0x0000000000000000000000000000000000000000', (0, 1500, 20), (0, 1520, 1480))
contract_EXRGameAsset_Pilot.fragments(2) -> (1, 0, 2, 3000, 1500, '', '0x0000000000000000000000000000000000000000', (0, 3000, 20), (0, 3020, 1480))
contract_EXRGameAsset_Racecraft.fragments(2) -> (1, 0, 2, 3000, 1500, '', '0x0000000000000000000000000000000000000000', (0, 3000, 20), (0, 3020, 1480))
Calling -> contract_EXRGameAsset_Pilot.createFragment(3, 3501, 4500, 20, {'from': contract_EXRSalesContract1})
Transaction sent: 0x718f97e0f6be33b33215843f662edd51471dd76fd1bc5ff5c70f8145e6e3aee6
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 7
  EXRGameAsset.createFragment confirmed   Block: 14628158   Gas used: 105866 (0.02%)

Calling -> contract_EXRGameAsset_Racecraft.createFragment(3, 3501, 4500, 20, {'from': contract_EXRSalesContract1})
Transaction sent: 0x148b0fed1f470073cb5239b08fa0a4f6e6cae36d7b58246f6d684d54e80e0452
  Gas price: 0.0 gwei   Gas limit: 600000000   Nonce: 8
  EXRGameAsset.createFragment confirmed   Block: 14628159   Gas used: 105866 (0.02%)

contract_EXRGameAsset_Pilot.fragments(0) -> (1, 0, 0, 0, 1500, '', '0x0000000000000000000000000000000000000000', (0, 0, 20), (0, 20, 1480))
contract_EXRGameAsset_Racecraft.fragments(0) -> (1, 0, 0, 0, 1500, '', '0x0000000000000000000000000000000000000000', (0, 0, 20), (0, 20, 1480))
contract_EXRGameAsset_Pilot.fragments(1) -> (1, 0, 1, 1500, 1500, '', '0x0000000000000000000000000000000000000000', (0, 1500, 20), (0, 1520, 1480))
contract_EXRGameAsset_Racecraft.fragments(1) -> (1, 0, 1, 1500, 1500, '', '0x0000000000000000000000000000000000000000', (0, 1500, 20), (0, 1520, 1480))
contract_EXRGameAsset_Pilot.fragments(2) -> (1, 0, 2, 3000, 1500, '', '0x0000000000000000000000000000000000000000', (0, 3000, 20), (0, 3020, 1480))
contract_EXRGameAsset_Racecraft.fragments(2) -> (1, 0, 2, 3000, 1500, '', '0x0000000000000000000000000000000000000000', (0, 3000, 20), (0, 3020, 1480))
contract_EXRGameAsset_Pilot.fragments(3) -> (1, 0, 3, 4500, 3501, '', '0x0000000000000000000000000000000000000000', (0, 4500, 20), (0, 4520, 3481))
contract_EXRGameAsset_Racecraft.fragments(3) -> (1, 0, 3, 4500, 3501, '', '0x0000000000000000000000000000000000000000', (0, 4500, 20), (0, 4520, 3481))
```

As we can see, we managed to create 4 different fragments successfully, which in total have a supply of 1500+1500+1500+3501 = 8001 NFTs.

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

It is recommended to force the first fragment to start with a firstId == 1.

Remediation Plan:

SOLVED: The Exiled Racers team states that the project is designed to start with firstId == 0. For this reason, the following fix was applied instead: if (firstId + fragmentSupply - 1 >= MAX_SUPPLY)revert FragmentExceedsCollectionSupply();

3.7 (HAL-07) INCORRECT EVENT EMISSION - LOW

Description:

In the EXRMintPass contract, the burnToRedeemPilot() function should emit a PilotRedeemed event with the address of the user who redeemed the pilot: event PilotRedeemed(address indexed user);:

Listing 22: EXRMintPass.sol (Line 104)

```
91 function burnToRedeemPilot(address account, uint256 fragment)
↳ external onlyRole(BURNER_ROLE) {
92     tokenBurnCounts[pilotPassTokenId]++;
93     tokenMintCountsByFragment[fragment][racecraftPassTokenId]++;
94     tokenMintCountsByFragment[fragment][inventoryTokenId]++;
95
96     _burn(account, pilotPassTokenId, 1);
97     emit PassBurned(account, pilotPassTokenId);
98
99     _mint(account, racecraftPassTokenId, 1, "");
100    emit PassMinted(account, racecraftPassTokenId, 1);
101
102    _mint(account, inventoryTokenId, 1, "");
103    emit PassMinted(account, inventoryTokenId, 1);
104    emit PilotRedeemed(msg.sender);
105 }
```

Since this function is always called from within a EXRSalesContract, msg.sender will always be here the address of the EXRSalesContract:

```
>>> tx2.events
{'TransferSingle': [OrderedDict([('operator', '0x170D6D18688022C17B0e44210B8174F4e15C88C7'), ('from', '0x0000000000000000000000000000000000000101'), ('to', '0x0000000000000000000000000000000000000000'), ('id', 3), ('value', 1)]), OrderedDict([('operator', '0x170D6D18688022C17B0e44210B8174F4e15C88C7'), ('from', '0x0000000000000000000000000000000000000000'), ('to', '0x0000000000000000000000000000000000000101'), ('id', 3), ('value', 1)]), OrderedDict([('operator', '0x170D6D18688022C17B0e44210B8174F4e15C88C7'), ('from', '0x0000000000000000000000000000000000000000'), ('to', '0x0000000000000000000000000000000000000101'), ('id', 3), ('value', 1)])], 'PassBurned': [OrderedDict([('account', '0x0000000000000000000000000000000000000101'), ('tokenId', 3)])], 'PassMinted': [OrderedDict([('recipient', '0x0000000000000000000000000000000000000101'), ('id', 2), ('qty', 1)]), OrderedDict([('recipient', '0x0000000000000000000000000000000000000101'), ('id', 3), ('qty', 1)])], 'PilotRedeemed': [OrderedDict([('user', '0x170D6D18688022C17B0e44210B8174F4e15C88C7')])], OrderedDict(), 'MintPassBurned': [OrderedDict([('user', '0x0000000000000000000000000000000000000101')])], 'Transfer': [OrderedDict([('from', '0x0000000000000000000000000000000000000000'), ('to', '0x0000000000000000000000000000000000000101'), ('tokenId', 33)])], 'GameAssetMinted': [OrderedDict([('recipient', '0x0000000000000000000000000000000000000101'), ('fragment', 1), ('tokenId', 33)])]}
>>> tx2.events['PilotRedeemed']
[OrderedDict([('user', '0x170D6D18688022C17B0e44210B8174F4e15C88C7')])], OrderedDict()]
>>> contract_EXRSalesContract1.address
'0x170D6D18688022C17B0e44210B8174F4e15C88C7'
```

Risk Level:

Likelihood - 3

Impact - 2

Recommendation:

It is recommended to use the `account` parameter instead of `msg.sender` when the event is emitted:
`emit PilotRedeemed(account);`

Also consider removing this event emission from the `burnToRedeemPilot()` function, as an identical event is emitted in the `redeemPilot()` function.

Remediation Plan:

SOLVED: The `Exiled Racers team` removed the `PilotRedeemed` event emission from the `burnToRedeemPilot()` function.

3.8 (HAL-08) UNUSED EVENT - INFORMATIONAL

Description:

In the EXRInventory contract, the event MintRoleGranted is declared but is not used anywhere in the code:

Listing 23: EXRInventoryERC1155.sol

```
33 event MintRoleGranted(address indexed minter);
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove the MintRoleGranted event declaration to reduce the deployment gas costs.

Remediation Plan:

SOLVED: The Exiled Racers team removed the MintRoleGranted event declaration from the EXRInventory contract.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

EXRGameAssetERC721.sol

```
ERC721Fragmentable.issueRandomId(uint256,bytes32,address) (contracts/extensions/ERC721Fragmentable.sol#223-260) uses a weak PRNG: "randomIndex = uint256(keccak256(bytes)(abi.encodePacked(recipient,blockhash(uint256)(block.number - 1),see
d))) % remaining (contracts/extensions/ERC721Fragmentable.sol#236-238)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

EXRGameAsset.mint(address,uint256,uint8,bytes32).i (contracts/EXRGameAssetERC721.sol#76) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#388-409) ignores return value by IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modul
es/@openzeppelin/contracts/token/ERC721/ERC721.sol#395-405)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

EXRGameAsset.constructor(string,string,string).name (contracts/EXRGameAssetERC721.sol#39) shadows:
        - ERC721.name() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#78-81) (function)
        - IERC721Metadata.name() (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#16) (function)
EXRGameAsset.constructor(string,string,string).symbol (contracts/EXRGameAssetERC721.sol#40) shadows:
        - ERC721.symbol() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#86-88) (function)
        - IERC721Metadata.symbol() (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#21) (function)
ERC721Fragmentable.walletOfOwner(address)._owner (contracts/extensions/ERC721Fragmentable.sol#202) shadows:
        - Ownable._owner (node_modules/@openzeppelin/contracts/access/Ownable.sol#21) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#395)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin
/contracts/token/ERC721/ERC721.sol#388-409) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#396)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#397)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin
/contracts/token/ERC721/ERC721.sol#388-409) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#398)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#397)' in ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin
/contracts/token/ERC721/ERC721.sol#388-409) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(reason)) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#402)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#388-409) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#401-403)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.9', '^0.8.0', '^0.8.1']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - 0.8.9 (contracts/EXRGameAssetERC721.sol#2)
        - 0.8.9 (contracts/extensions/ERC721Fragmentable.sol#3)
        - 0.8.9 (contracts/interfaces/IRenderer.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ERC721Enumerable._removeTokenFromAllTokensEnumeration(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#144-162) has costly operations inside a loop:
        - delete _allTokensIndex[tokenId] (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#160)
ERC721Enumerable._removeTokenFromAllTokensEnumeration(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#144-162) has costly operations inside a loop:
        - _allTokens.pop() (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#161)
ERC721Enumerable._removeTokenFromOwnerEnumeration(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#119-137) has costly operations inside a loop:
        - delete _ownedTokensIndex[tokenId] (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#135)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

AccessControl._setRoleAdmin(bytes32,bytes32) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#194-198) is never used and should be removed
AccessControl._setupRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#185-187) is never used and should be removed
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-57) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#95-101) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#174-176) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#147-149) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC721._baseURI() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#105-107) is never used and should be removed
ERC721._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#304-318) is never used and should be removed
ERC721._safeMint(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#248-250) is never used and should be removed
ERC721._safeMint(address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#256-266) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Enumerable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#4) allows old versions
```

```
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version0.8.9 (contracts/EXRGameAssetERC721.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.9 (contracts/extensions/ERC721Fragmentable.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.9 (contracts/interfaces/IRenderer.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#179) is not in mixedCase
Parameter ERC721Fragmentable.walletOfOwner(address)._owner (contracts/extensions/ERC721Fragmentable.sol#202) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#161-165)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
name() should be declared external:
        - ERC721.name() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#79-81)
symbol() should be declared external:
        - ERC721.symbol() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#86-88)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#93-98)
        - EXRGameAsset.tokenURI(uint256) (contracts/EXRGameAssetERC721.sol#164-170)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#112-122)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#136-138)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#150-159)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#164-170)
tokenByIndex(uint256) should be declared external:
        - ERC721Enumerable.tokenByIndex(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol#52-55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

EXRInventoryController.sol

```
EXRInventoryController.removeCategory(uint8).i (contracts/EXRInventoryController.sol#181) is a local variable never initialized
EXRInventoryController._claimRandomItems(bytes32,uint256).ii (contracts/EXRInventoryController.sol#237) is a local variable never initialized
EXRInventoryController._claimRandomItems(bytes32,uint256).id (contracts/EXRInventoryController.sol#236) is a local variable never initialized
EXRInventoryController.addCategory(uint8,uint8[9]).i (contracts/EXRInventoryController.sol#156) is a local variable never initialized
EXRInventoryController._claimRandomItems(bytes32,uint256).i (contracts/EXRInventoryController.sol#230) is a local variable never initialized
EXRInventoryController.removeCategory(uint8).removed (contracts/EXRInventoryController.sol#180) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

CouponSystem.constructor(address).signer (contracts/extensions/CouponSystem.sol#29) lacks a zero-check on :
                - _adminSigner = signer (contracts/extensions/CouponSystem.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in EXRInventoryController._claimRandomItems(bytes32,uint256) (contracts/EXRInventoryController.sol#226-250):
        External calls:
        - inventoryContract.mintBatch(_msgSender(),ids,amounts,) (contracts/EXRInventoryController.sol#248)
        Event emitted after the call(s):
        - InventoryItemsClaimed(ids,amounts) (contracts/EXRInventoryController.sol#249)
Reentrancy in EXRInventoryController.burnToRedeemInventoryItems(bytes32,uint256,CouponSystem.Coupon) (contracts/EXRInventoryController.sol#103-118):
        External calls:
        - mintpassContract.authorizedBurn(_msgSender(),inventoryPassId) (contracts/EXRInventoryController.sol#116)
        - _claimRandomItems(seed,qty) (contracts/EXRInventoryController.sol#117)
                - inventoryContract.mintBatch(_msgSender(),ids,amounts,) (contracts/EXRInventoryController.sol#248)
        Event emitted after the call(s):
        - InventoryItemsClaimed(ids,amounts) (contracts/EXRInventoryController.sol#249)
                - _claimRandomItems(seed,qty) (contracts/EXRInventoryController.sol#117)
Reentrancy in EXRInventoryController.claimRewardItems(bytes32,uint256,CouponSystem.Coupon) (contracts/EXRInventoryController.sol#81-94):
        External calls:
        - _claimRandomItems(seed,qty) (contracts/EXRInventoryController.sol#92)
                - inventoryContract.mintBatch(_msgSender(),ids,amounts,) (contracts/EXRInventoryController.sol#248)
        Event emitted after the call(s):
        - InventoryRewardClaimed(_msgSender(),qty) (contracts/EXRInventoryController.sol#93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC2771Context._msgSender() (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#24-33) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#27-29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.9', '^0.8.0', '^0.8.9']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.9 (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - 0.8.9 (contracts/EXRInventoryController.sol#2)
        - 0.8.9 (contracts/extensions/CouponSystem.sol#2)
        - 0.8.9 (contracts/interfaces/IEXRInventory.sol#2)
        - 0.8.9 (contracts/interfaces/IEXRMintPass.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

EXRInventoryController.removeCategory(uint8) (contracts/EXRInventoryController.sol#179-194) has costly operations inside a loop:
        - categories.pop() (contracts/EXRInventoryController.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

AccessControl._setRoleAdmin(bytes32,bytes32) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#194-198) is never used and should be removed
AccessControl._setupRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#185-187) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC2771Context._msgData() (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#35-41) is never used and should be removed
EXRInventoryController._msgData() (contracts/EXRInventoryController.sol#304-312) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.9 (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version0.8.9 (contracts/EXRInventoryController.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.9 (contracts/extensions/CouponSystem.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.9 (contracts/interfaces/IEXRInventory.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.9 (contracts/interfaces/IEXRMintPass.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Constant EXRInventoryController.inventoryPassId (contracts/EXRInventoryController.sol#38) is not in UPPER_CASE_WITH_UNDERSCORES
Variable CouponSystem._adminSigner (contracts/extensions/CouponSystem.sol#14) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#161-165)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

EXRInventoryERC1155.sol

```
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#445) is a local variable never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#450) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#427) is a local variable never initialized
EXRInventory.batchSetTokenURIs(uint256[],string[]).i (contracts/EXRInventoryERC1155.sol#146) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#414-433) ignores return value by IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#435-456) ignores return value by IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#444-454)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423)' in ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#414-433) potentially used before declaration: response != IERC1155Receiver.onERC1155Received.selector (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#424)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#427)' in ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#414-433) potentially used before declaration: revert(string)(reason) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#428)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#445)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#435-456) potentially used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#447)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#450)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#435-456) potentially used before declaration: revert(string)(reason) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.9', '^0.8.0', '^0.8.1']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155Burnable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155Supply.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155MetadataURI.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - 0.8.9 (contracts/EXRInventoryERC1155.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControl._setRoleAdmin(bytes32,bytes32) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#194-198) is never used and should be removed
AccessControl._setupRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#185-187) is never used and should be removed
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#85-87) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#95-101) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#174-176) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#147-149) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed

Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155Burnable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155Supply.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155MetadataURI.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version0.8.9 (contracts/EXRInventoryERC1155.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable EXRInventory._itemURIs (contracts/EXRInventoryERC1155.sol#31) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#161-165)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
uri(uint256) should be declared external:
        - ERC1155.uri(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#59-61)
        - EXRInventory.uri(uint256) (contracts/EXRInventoryERC1155.sol#104-107)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155.balanceOfBatch(address[],uint256[]) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#82-98)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155.setApprovalForAll(address,bool) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#103-105)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#117-125)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#134-146)
burn(address,uint256,uint256) should be declared external:
        - ERC1155Burnable.burn(address,uint256,uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Burnable.sol#15-26)
burnBatch(address,uint256[],uint256[]) should be declared external:
        - ERC1155Burnable.burnBatch(address,uint256[],uint256[]) (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Burnable.sol#28-39)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

EXRMintPassERC1155.sol

```
Reentrancy in EXRMintPass.burnToRedeemPilot(address,uint256) (contracts/EXRMintPassERC1155.sol#91-105):
        External calls:
        - _mint(account,racecraftPassTokenId,1,) (contracts/EXRMintPassERC1155.sol#99)
          - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
        - _mint(account,inventoryTokenId,1,) (contracts/EXRMintPassERC1155.sol#102)
          - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
        State variables written after the call(s):
        - _mint(account,inventoryTokenId,1,) (contracts/EXRMintPassERC1155.sol#102)
          - _balances[id][to] += amount (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#272)
        - _mint(account,inventoryTokenId,1,) (contracts/EXRMintPassERC1155.sol#102)
          - _totalSupply[ids[i]] += amounts[i] (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Supply.sol#48)
          - _totalSupply[ids[i_scope_0]] -= amounts[i_scope_0] (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Supply.sol#54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#445) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423) is a local variable never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#450) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#427) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#414-433) ignores return value by IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#435-456) ignores return value by IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,amounts,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#444-454)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423)' in ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#414-433) potentially used before declaration: response != IERC1155Receiver.onERC1155Received.selector (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#424)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#427)' in ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#414-433) potentially used before declaration: revert(string)(reason) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#428)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#445)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#435-456) potentially used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#447)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#450)' in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#435-456) potentially used before declaration: revert(string)(reason) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
```

```
Reentrancy in EXRMintPass.burnToRedeemPilot(address,uint256) (contracts/EXRMintPassERC1155.sol#91-105):
        External calls:
        - _mint(account,racecraftPassTokenId,1,) (contracts/EXRMintPassERC1155.sol#99)
            - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
        Event emitted after the call(s):
        - PassMinted(account,racecraftPassTokenId,1) (contracts/EXRMintPassERC1155.sol#100)
Reentrancy in EXRMintPass.burnToRedeemPilot(address,uint256) (contracts/EXRMintPassERC1155.sol#91-105):
        External calls:
        - _mint(account,racecraftPassTokenId,1,) (contracts/EXRMintPassERC1155.sol#99)
            - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
        - _mint(account,inventoryTokenId,1,) (contracts/EXRMintPassERC1155.sol#102)
            - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
        Event emitted after the call(s):
        - PassMinted(account,inventoryTokenId,1) (contracts/EXRMintPassERC1155.sol#103)
        - PilotRedeemed(msg.sender) (contracts/EXRMintPassERC1155.sol#104)
        - TransferSingle(operator,address(0),to,id,amount) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#273)
            - _mint(account,inventoryTokenId,1,) (contracts/EXRMintPassERC1155.sol#102)
Reentrancy in EXRMintPass.mint(address,uint256,uint256,uint256) (contracts/EXRMintPassERC1155.sol#72-83):
        External calls:
        - _mint(recipient,tokenId,qty,) (contracts/EXRMintPassERC1155.sol#81)
            - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
        Event emitted after the call(s):
        - PassMinted(recipient,tokenId,qty) (contracts/EXRMintPassERC1155.sol#82)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.9', '^0.8.0', '^0.8.1']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Burnable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Supply.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155MetadataURI.sol#4)
        - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - 0.8.9 (contracts/EXRMintPassERC1155.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControl._setRoleAdmin(bytes32,bytes32) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#194-198) is never used and should be removed
AccessControl._setupRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#165-167) is never used and should be removed
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#85-87) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#95-101) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#114-120) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#174-176) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#147-149) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#201-221) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC1155._mintBatch(address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#287-307) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155Receiver.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Burnable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Supply.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155MetadataURI.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version0.8.9 (contracts/EXRMintPassERC1155.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Constant EXRMintPass.pilotPassTokenId (contracts/EXRMintPassERC1155.sol#31) is not in UPPER_CASE_WITH_UNDERSCORES
Constant EXRMintPass.racecraftPassTokenId (contracts/EXRMintPassERC1155.sol#32) is not in UPPER_CASE_WITH_UNDERSCORES
Constant EXRMintPass.inventoryTokenId (contracts/EXRMintPassERC1155.sol#33) is not in UPPER_CASE_WITH_UNDERSCORES
Variable EXRMintPass._tokenURIs (contracts/EXRMintPassERC1155.sol#45) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#161-165)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-56)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/contracts/access/Ownable.sol#62-65)
uri(uint256) should be declared external:
        - ERC1155.uri(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#59-61)
        - EXRMintPass.uri(uint256) (contracts/EXRMintPassERC1155.sol#166-170)
balanceOfBatch(address[],uint256[]) should be declared external:
        - ERC1155.balanceOfBatch(address[],uint256[]) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#82-98)
setApprovalForAll(address,bool) should be declared external:
        - ERC1155.setApprovalForAll(address,bool) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#103-105)
safeTransferFrom(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#117-129)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should be declared external:
        - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#134-146)
burn(address,uint256,uint256) should be declared external:
        - ERC1155Burnable.burn(address,uint256,uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Burnable.sol#15-26)
burnBatch(address,uint256[],uint256[]) should be declared external:
        - ERC1155Burnable.burnBatch(address,uint256[],uint256[]) (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Burnable.sol#28-39)

exists(uint256) should be declared external:
        - ERC1155Supply.exists(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/ERC1155Supply.sol#29-31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

EXRSalesContract.sol

```
EXRSalesContract.airdropMintpass(uint256,uint256[],address[]).i_scope_0 (contracts/EXRSalesContract.sol#313) is a local variable never initialized
EXRSalesContract.airdropMintpass(uint256,uint256[],address[]).i (contracts/EXRSalesContract.sol#305) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

CouponSystem.constructor(address).signer (contracts/extensions/CouponSystem.sol#29) lacks a zero-check on :
        - _adminSigner = signer (contracts/extensions/CouponSystem.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

EXRSalesContract.airdropMintpass(uint256,uint256[],address[]) (contracts/EXRSalesContract.sol#295-317) has external calls inside a loop: mintPassContract.mint(recipients[i_scope_0],qtys[i_scope_0],tokenId,dedicatedFragment) (contracts/EXRSalesContract.sol#314)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Reentrancy in EXRSalesContract.claimPilotPass(CouponSystem.Coupon,uint256,uint256,uint256) (contracts/EXRSalesContract.sol#134-171):
        External calls:
        - mintPassContract.incrementPilotPassClaimCount(caller,dedicatedFragment,qty) (contracts/EXRSalesContract.sol#164)
        - mintPassContract.mint(caller,qty,pilotPassTokenId,dedicatedFragment) (contracts/EXRSalesContract.sol#165)
        Event emitted after the call(s):
        - MintPassClaimed(caller,qty) (contracts/EXRSalesContract.sol#166)
Reentrancy in EXRSalesContract.claimPilotPass(CouponSystem.Coupon,uint256,uint256,uint256) (contracts/EXRSalesContract.sol#134-171):
        External calls:
        - mintPassContract.incrementPilotPassClaimCount(caller,dedicatedFragment,qty) (contracts/EXRSalesContract.sol#164)
        - mintPassContract.mint(caller,qty,pilotPassTokenId,dedicatedFragment) (contracts/EXRSalesContract.sol#165)
        - refundCaller(caller,paid - amountOwed) (contracts/EXRSalesContract.sol#169)
            - (success) = buyer.call{value: amount}() (contracts/EXRSalesContract.sol#380)
        External calls sending eth:
        - refundCaller(caller,paid - amountOwed) (contracts/EXRSalesContract.sol#169)
            - (success) = buyer.call{value: amount}() (contracts/EXRSalesContract.sol#380)
        Event emitted after the call(s):
        - RefundIssued(buyer,amount) (contracts/EXRSalesContract.sol#382)
            - refundCaller(caller,paid - amountOwed) (contracts/EXRSalesContract.sol#169)
Reentrancy in EXRSalesContract.createFragments(uint64,uint64,uint64,uint64) (contracts/EXRSalesContract.sol#330-348):
        External calls:
        - pilotContract.createFragment(dedicatedFragment,fragmentSupply,firstId,reservedPilots) (contracts/EXRSalesContract.sol#340)
        - racecraftContract.createFragment(dedicatedFragment,fragmentSupply,firstId,reservedRacecrafts) (contracts/EXRSalesContract.sol#341-346)
        Event emitted after the call(s):
        - SalesFragmentCreated(fragmentSupply,firstId,reservedPilots,reservedRacecrafts) (contracts/EXRSalesContract.sol#347)
```

AUTOMATED TESTING

```
Reentrancy in EXRSalesContract.redeemPilot(bytes32,CouponSystem.Coupon) (contracts/EXRSalesContract.sol#188-204):
        External calls:
        - mintPassContract.burnToRedeemPilot(caller,dedicatedFragment) (contracts/EXRSalesContract.sol#199)
        Event emitted after the call(s):
        - MintPassBurned(caller) (contracts/EXRSalesContract.sol#200)
Reentrancy in EXRSalesContract.redeemPilot(bytes32,CouponSystem.Coupon) (contracts/EXRSalesContract.sol#188-204):
        External calls:
        - mintPassContract.burnToRedeemPilot(caller,dedicatedFragment) (contracts/EXRSalesContract.sol#199)
        - pilotContract.mint(caller,1,dedicatedFragment,seed) (contracts/EXRSalesContract.sol#202)
        Event emitted after the call(s):
        - PilotRedeemed() (contracts/EXRSalesContract.sol#203)
Reentrancy in EXRSalesContract.redeemRacecraft(bytes32,CouponSystem.Coupon) (contracts/EXRSalesContract.sol#213-229):
        External calls:
        - mintPassContract.authorizedBurn(caller,racecraftPassTokenId) (contracts/EXRSalesContract.sol#226)
        - racecraftContract.mint(caller,1,dedicatedFragment,seed) (contracts/EXRSalesContract.sol#227)
        Event emitted after the call(s):
        - RacecraftRedeemed() (contracts/EXRSalesContract.sol#228)
Reentrancy in EXRSalesContract.refundCaller(address,uint256) (contracts/EXRSalesContract.sol#379-383):
        External calls:
        - (success) = buyer.call{value: amount}() (contracts/EXRSalesContract.sol#380)
        Event emitted after the call(s):
        - RefundIssued(buyer,amount) (contracts/EXRSalesContract.sol#382)
Reentrancy in EXRSalesContract.withdrawBalance() (contracts/EXRSalesContract.sol#365-369):
        External calls:
        - (success) = msg.sender.call{value: address(this).balance}() (contracts/EXRSalesContract.sol#366)
        Event emitted after the call(s):
        - BalanceWithdrawn() (contracts/EXRSalesContract.sol#368)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC2771Context._msgSender() (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#24-33) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#27-29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.9', '^0.8.0', '^0.8.9']
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4)
        - ^0.8.9 (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - 0.8.9 (contracts/EXRSalesContract.sol#2)
        - 0.8.9 (contracts/extensions/CouponSystem.sol#2)
        - 0.8.9 (contracts/interfaces/IEXRGameAsset.sol#2)
        - 0.8.9 (contracts/interfaces/IEXRMintPass.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

AccessControl._setRoleAdmin(bytes32,bytes32) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#194-198) is never used and should be removed
AccessControl._setupRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#185-187) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
ERC2771Context._msgData() (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#35-41) is never used and should be removed
EXRSalesContract._msgData() (contracts/EXRSalesContract.sol#401-409) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#15-35) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/IAccessControl.sol#4) allows old versions
Pragma version^0.8.9 (node_modules/@openzeppelin/contracts/metatx/ERC2771Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version0.8.9 (contracts/EXRSalesContract.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.9 (contracts/extensions/CouponSystem.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.9 (contracts/interfaces/IEXRGameAsset.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.9 (contracts/interfaces/IEXRMintPass.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in EXRSalesContract.withdrawBalance() (contracts/EXRSalesContract.sol#365-369):
        - (success) = msg.sender.call{value: address(this).balance}() (contracts/EXRSalesContract.sol#366)
Low level call in EXRSalesContract.refundCaller(address,uint256) (contracts/EXRSalesContract.sol#379-383):
        - (success) = buyer.call{value: amount}() (contracts/EXRSalesContract.sol#380)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Constant EXRSalesContract.pilotPassTokenId (contracts/EXRSalesContract.sol#49) is not in UPPER_CASE_WITH_UNDERSCORES
Constant EXRSalesContract.racecraftPassTokenId (contracts/EXRSalesContract.sol#50) is not in UPPER_CASE_WITH_UNDERSCORES
Variable CouponSystem._adminSigner (contracts/extensions/CouponSystem.sol#14) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

grantRole(bytes32,address) should be declared external:
        - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#130-132)
revokeRole(bytes32,address) should be declared external:
        - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#143-145)
renounceRole(bytes32,address) should be declared external:
        - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#161-165)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

- The weak PRNG was correctly flagged by Slither.
- The reentrancies flagged by Slither are all false positives.

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

EXRGameAssetERC721.sol

No issues found by MythX.

EXRInventoryController.sol

Report for contracts/EXRInventoryController.sol
https://dashboard.mythx.io/#/console/analyses/2cf0d038-e635-4998-ab6f-3b5e22412919

Line	SWC Title	Severity	Short Description
233	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

EXRInventoryERC1155.sol

Report for contracts/EXRInventoryERC1155.sol
https://dashboard.mythx.io/#/console/analyses/27f720b5-cf1f-4bb8-a9d5-d509e9de156a

Line	SWC Title	Severity	Short Description
22	(SWC-123) Requirement Violation	Low	Requirement violation.

EXRMintPassERC1155.sol

Report for contracts/EXRMintPassERC1155.sol
https://dashboard.mythx.io/#/console/analyses/0299f56e-14de-4705-85e6-c0ec17e34de0

Line	SWC Title	Severity	Short Description
26	(SWC-123) Requirement Violation	Low	Requirement violation.

EXRSalesContract.sol

No issues found by MythX.

- The weak source of randomness issue was correctly flagged by MythX.
- The requirement violations are all false positives.

THANK YOU FOR CHOOSING

// HALBORN