



**Blackbirdz**

Security Assessment Report  
**BeaRex**

Sep 6, 2022

# Executive Summary

---

This report has been prepared for BeaRex to discover vulnerabilities and issues in the source code of the BeaRex smart contracts as well as any contract dependencies that were not part of an officially recognized library. A comprehensive assesment has been performed, utilizing Static Analysis, Dynamic Analysis and Manual Review techniques.

During the audit process, special attention is paid to the following considerations:

- Assessing the codebase to ensure compliance with current industry standards and best practices in smart contract development.
- Testing the smart contracts against common and uncommon attack vectors.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to low severity. We suggest addressing these findings to achive a higher level of security standards and best practices. We offer recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public repos;
- Provide more transparency on privileged activities once the protocol is live.

# Overview

---





## Project Summary

<b>Project name</b>	<b>BeaRex</b>
Protocol	Ethereum
Platform	EVM
Language	Solidity
Type	ERC721
Codebase	<a href="https://github.com/primadg/bearex">https://github.com/primadg/bearex</a>
Commits	ee13abadf4330cb0d48d98d5f4761734352eca1b a05ffa6ebfe6fb19a341a594cc00d76a6baf46b1 16850c35e1f44c894a78a9ab5e36db2ae223534f










## Audit Summary

<b>Timeline</b>	<b>Aug 23, 2022 - Sep 6, 2022</b>
Audit Methodology	Static Analysis, Manual Review, Dynamic Analysis, Architecture Review
Key Components	BearExBox.sol, BearExBoxBadge.sol, BearExNFT.sol
Change Log	Aug 25, 2022 - Initial Review Sep 5, 2022 - Second Review Sep 6, 2022 - Final Review

## Findings Summary

<b>Total Issues</b>	<b>9</b>
 Critical	2
 High	1
 Medium	3
 Low	3

# FINDINGS

ID	Title	Severity	Status
BRX-1	Broken redeem	 Critical	Resolved
BRX-2	The project has no tests	 Critical	Resolved
BRX-3	Double redeem	 High	Resolved
BRX-4	BearExNFT shadows <code>_tokenURIs</code>	 Medium	Resolved
BRX-5	Insecure random	 Medium	Acknowledged
BRX-6	<code>addTokensToRedeem</code> missing check	 Medium	Resolved
BRX-7	Save gas in <code>_redeemToken</code>	 Low	Resolved
BRX-8	<code>addRedeemGroup</code> can overwrite an existing group	 Low	Resolved
BRX-9	<code>rarityId</code> is not used in <code>mintPassContract.burnFromRedeem</code>	 Low	Resolved

## Finding Severity Breakdown

<b>Severity</b>	<b>Description</b>
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Other non-essential issues and recommendations reported to/ acknowledged by the team.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

<b>Status</b>	<b>Description</b>
Resolved	Recommended fixes have been made to the code and no longer affect project security.
Partially Resolved	Fixes have been made to the code but issue is still affect project security.
Acknowledged	The Customer is aware of the finding. The risk is not relevant to the project and accepted by the Customer. Or recommendations for the finding are planned to be resolved in the future.
Unresolved	Recommended fixes have not been made to the project code and affect project security.

## CRITICAL

### BRX-1: Broken redeem

---

The method `BearExNFT._redeemToken` removes the wrong NFT from the `tokensForRedeem[_].nftIds` array.

A user can get random NFT by calling `RedeemNFT` at

<https://github.com/primadg/bearex/blob/ee13abadf4330cb0d48d98d5f4761734352eca1b/BearExNFT.sol#L71>

`RedeemNFT` calls `_redeemToken` to extract a token id from `tokensForRedeem[rarity].nftIds` array.

It should remove a chosen element from the array but instead it simply decreases the array's length and leave a chosen element in place:

<https://github.com/primadg/bearex/blob/ee13abadf4330cb0d48d98d5f4761734352eca1b/BearExNFT.sol#L139-L142>

Thus, the next user who calls `BearExNFT._redeemToken` may receive a token that has already been assigned to another user because there is no check in `RedeemNFT` that the token has already been redeemed.

### Recommendations

1. To remove a chosen element from the array it is recommended to swap the chosen element with the last one and then decrease the array's length:

```
// choose an element
tokenPosition = random() % tokensForRedeem[rarity].totalTokensToRedeem;
tokenToRedeem = tokensForRedeem[rarity].nftIds[tokenPosition];

// remove the element
uint256 lastIndex = tokensForRedeem[rarity].totalTokensToRedeem - 1;
uint256 lastElement = tokensForRedeem[rarity].nftIds[lastIndex];

tokensForRedeem[rarity].nftIds[tokenPosition] = lastElement;
tokensForRedeem[rarity].nftIds[lastIndex] = 0; // gas refund
tokensForRedeem[rarity].totalTokensToRedeem--;
```

2. `RedeemNFT` should check that the token has already been redeemed before.

### Status

Resolved

## BRX-2: The project has no tests

---

Any changes to the code before the contract is deployed to the network can break the intended logic of the contract and no one will know about it until users encounter a problem.

### **Recommendations**

It is recommended to write tests covering the core business logic of the contract:

- deploying the contract;
- minting an NFT;
- redeeming an NFT;
- ensuring that the redeemed NFT was correctly removed from the queue of redeemable tokens.

### **Status**

Resolved

## HIGH

### BRX-3: Double redeem

---

An owner of the contract can call `addTokensToRedeem` to add an already redeemed NFT back to the queue of redeemable tokens:

<https://github.com/primadg/bearex/blob/ee13abadf4330cb0d48d98d5f4761734352eca1b/BearExNFT.sol#L126>

The method `RedeemNFT` doesn't check that the NFT was already redeemed by some user and just assigns it to another.

### Recommendations

1. `addTokensToRedeem(rarity, tokenId)` should revert if the token was already added to the queue or if it was redeemed.
2. `RedeemNFT` should revert if the user gets a token that has already been redeemed by another user.

### Status

Resolved



## MEDIUM

### BRX-4: BearExNFT shadows `_tokenURIs`

---

BearExNFT contract has a public variable `_tokenURIs` that shadows a private variable `_tokenURIs` from ERC721URIStorage:

<https://github.com/primadg/bearex/blob/ee13abadf4330cb0d48d98d5f4761734352eca1b/BearExNFT.sol#L18>

This may confuse the developer which may lead to mistakes in the code. Removing this variable and replacing it with the existing ERC721URIStorage `tokenURI` and `_setTokenURI` methods will also save gas.

#### **Recommendations**

Remove the public variable `_tokenURIs` and use ERC721URIStorage `tokenURI` and `_setTokenURI` methods instead.

#### **Status**

Resolved

## BRX-5: Insecure random

---

A random token is selected for the user using the `random` function:

<https://github.com/primadg/bearex/blob/ee13abadf4330cb0d48d98d5f4761734352eca1b/BearExNFT.sol#L95>

It uses `block.difficulty` and `block.timestamp` as a source of randomness. Those parameters are predictable and the last one can easily be manipulated by a miner.

### Recommendations

Either use random oracles (Chainlink) or accept the risk that a miner can non randomly redeem tokens for himself.

### Status

Acknowledged

## BRX-6: `addTokensToRedeem` missing check

---

The method `addTokensToRedeem` doesn't check that a `tokenId` was minted:

<https://github.com/primadg/bearex/blob/ee13abadf4330cb0d48d98d5f4761734352eca1b/BearExNFT.sol#L126>

This means that a non-existent `tokenId` could be added to the queue of redeemable tokens.

### Recommendations

Add a check.

### Status

Resolved

 **LOW****BRX-7: Save gas in `_redeemToken`**

---

The line

<https://github.com/primadg/bearex/blob/ee13abadf4330cb0d48d98d5f4761734352eca1b/BearExNFT.sol#L142>  
doesn't do anything:

```
tokensForRedeem[rarity].nftIds[tokensForRedeem[rarity].totalTokensToRedeem];
```

**Recommendations**

Fix or remove the line.

**Status**

Resolved

## BRX-8: `addRedeemGroup` can overwrite an existing group

---

The method `addRedeemGroup(rarityId)` at <https://github.com/primadg/bearex/blob/ee13abadf4330cb0d48d98d5f4761734352eca1b/BearExNFT.sol#L142> doesn't check that `rarityId` may already exist.

### Recommendations

Add a check.

### Status

Resolved

**BRX-9: `rarityId` is not used in `mintPassContract.burnFromRedeem`**

---

The method `burnFromRedeem` always uses `rarityId=1`:

<https://github.com/primadg/bearex/blob/ee13abadf4330cb0d48d98d5f4761734352eca1b/BearExBox.sol#L45>

```
_burn(account, 1, 1);
```

**Recommendations**

It seems that the developers forgot to add the `rarityId` parameter to the `burnFromRedeem` method.

**Status**

Resolved

## Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

## Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Blackbirdz. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Blackbirdz. The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.