Security Assessment

XBroker

Jul 7th, 2021

Table of Contents

Summary

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

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- 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 informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest 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 given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	XBroker
Platform	BSC
Language	Solidity
Codebase	https://github.com/xcarnival/XBroker
Commit	ffa1387c200a14d4b1f8d7491403652edef5b5c6

Audit Summary

Delivery Date	Jul 07, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
Critical	0	0	0	0	0	0
Major	2	0	0	1	1	0
Medium	2	0	0	1	1	0
Minor	4	0	0	3	1	0
Informational	3	0	1	2	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum
BXB	Broker.sol	ae521061f39501638b3381ee34cd8145a72c67e6991bf6ea117b030a2d5ff6ad

Centralized Risks

The owner of the contract `Broker` has the permission to:

- Admin can change the `redemptionPeriod` which is the time that pledger can repay his order via function `setRedemptionPeriod()`,
- Admin can change the `clearingPeriod` which is the time of auction via function `setClearingPeriod()`,
- Admin can change the `repayInterestCut` which will affect the cut of interest via function `setRepayInterestCut()`,
- Admin can change the `auctionPledgerCut` and `auctionDevCut` which will affect the cut of the profit of auction via function `setAuctionCut()`,
- Admin can change the `maxLendersCnt` of an order and the `defaultMaxLendersCnt` to limit the number of lenders via functions `setMaxLendersCnt()` and `setDefaultMaxLendersCnt()`,

without obtaining the consensus of the community.

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multi-sig governing procedure and let the community monitor in respect of transparency considerations.

Financial Models

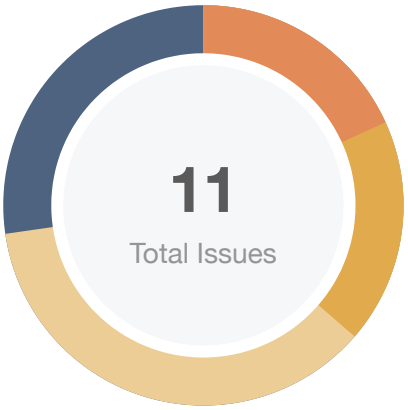
Financial models of blockchain protocols need to be resilient to attacks. It needs to pass simulations and verifications to guarantee the security of the overall protocol.

The XBroker protocol is XCarnival's original time-limited auction clearing mechanism. The pledge model is optimised for special types of assets with extremely poor liquidity, such as NFTs.

The pledger provides NFT assets as collateral for lending purpose. The fund lender provides funds. However, the real-value of NFT is hard to be estimated for lender.

The scope of this audit is not including tokenomics, hence we strongly recommend the team to take serious considerations.

Findings



Critical	**0** (0.00%)
Major	**2** (18.18%)
Medium	**2** (18.18%)
Minor	**4** (36.36%)
Informational	**3** (27.27%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
BXB-01	Lack of input validation	Logical Issue	Minor	Resolved
BXB-02	Lack of input validation	Logical Issue	Minor	Resolved
BXB-03	Missing emit events	Logical Issue	Minor	Resolved
BXB-04	Proper usage of “public” and “external” type	Coding Style	Informational	Resolved
BXB-05	Logical issue of function `setBeneficiary()`	Logical Issue	Informational	Resolved
BXB-06	Unreasonable upper boundary	**Centralization / Privilege**	**Major**	**Resolved**
BXB-07	Gas consumption in loop	Gas Optimization	Medium	Resolved
BXB-08	Privileged ownership	**Centralization / Privilege**	**Major**	**Acknowledged**
BXB-09	Logical Issue of `_profit` distribution	Logical Issue	Informational	Partially Resolved
BXB-10	External dependency	Logical Issue	Minor	Acknowledged
BXB-11	Risk of supported NFT digital assets	**Centralization / Privilege**	**Medium**	**Acknowledged**

BXB-01 | Lack of input validation

Category	Severity	Location	Status
Logical Issue	Minor	contracts/Broker.sol: 194	Resolved

Description

The function `initialize()` parameters are not sanitized for zero address validation.

```
function initialize(
    address _usdxc,
    address _beneficiary,
    uint256 _redemptionPeriod,
    uint256 _clearingPeriod,
    uint256 _repayInterestCut,
    uint256 _auctionPledgerCut,
    uint256 _auctionDevCut
) public initializer {
    __ReentrancyGuard_init();
    admin = msg.sender;
    usdxc = _usdxc;
    beneficiary = _beneficiary;
    redemptionPeriod = _redemptionPeriod;
    clearingPeriod = _clearingPeriod;
    repayInterestCut = _repayInterestCut;
    auctionPledgerCut = _auctionPledgerCut;
    auctionDevCut = _auctionDevCut;
}
```

Recommendation

Consider adding checks as following example:

```
require(_usdxc != address(0), "_usdxc is zero address");
require(_beneficiary != address(0), "_beneficiary is zero address");
```

Alleviation

The team heeded our advice and resolved this issue in commit eb097e6174deda87e1827ce28e5887ed04f793fa.

BXB-02 | Lack of input validation

Category	Severity	Location	Status
Logical Issue	Minor	contracts/Broker.sol: 702	Resolved

Description

The function `setBeneficiary()` parameter is not sanitized for zero address validation.

```
function setBeneficiary(address _beneficiary) external onlyBeneficiary {
    beneficiary = _beneficiary;
}
```

Recommendation

Consider adding checks as following example:

```
require(_beneficiary != address(0), "_beneficiary is zero address");
```

Alleviation

The team removed this function in commit eb097e6174deda87e1827ce28e5887ed04f793fa.

BXB-03 | Missing emit events

Category	Severity	Location	Status
Logical Issue	Minor	contracts/Broker.sol: 673, 679, 685, 690	Resolved

Description

Several sensitive actions are defined without event declarations.

1. Fucntions `setRedemptionPeriod()`, `setClearingPeriod()`, `setRepayInterestCut()`, `setAuctionCut()` will decide the important metrics.
2. Function `setBeneficiary()` will change the beneficiary address of this contract.
3. Function `initialize()` will decide several sensitive state variables.

Recommendation

Consider adding events for sensitive actions, and emit it in the function.

Alleviation

The team heeded our advice and resolved this issue in commit eb097e6174deda87e1827ce28e5887ed04f793fa.

BXB-04 | Proper usage of “public” and “external” type

Category	Severity	Location	Status
Coding Style	Informational	contracts/Broker.sol: 194, 646, 663	Resolved

Description

“public” functions that are never called by the contract should be declared “external”. When the inputs are arrays, “external” functions are more efficient than “public” functions.

Examples:

Functions like : `initialize()`, `lenderOfferInfo()`, `t1()`

Recommendation

Consider using the “external” attribute for functions never called from the contract.

Alleviation

The team heeded our advice and resolved this issue in commit eb097e6174deda87e1827ce28e5887ed04f793fa.

BXB-05 | Logical issue of function `setBeneficiary()`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/Broker.sol: 702	Resolved

Description

The modifier `onlyBeneficiary` is used on the function `setBeneficiary()` which is used to change the beneficiary.

```
function setBeneficiary(address _beneficiary) external onlyBeneficiary {
    beneficiary = _beneficiary;
}
```

In case `_beneficiary` is a wrong address and `beneficiary` can't be changed anymore, a pending variable of `beneficiary` and a function to accept the pending variable is recommended.

Recommendation

Consider adding a pending variable of `beneficiary` and a function to accept the pending variable.

Alleviation

The team heeded our advice and resolved this issue in commits eb097e6174deda87e1827ce28e5887ed04f793fa and 218796a852206f057d9c96b66969bb3ec17450c7.

[XCarnival Team]:

When changing the beneficiary, we need two steps.

Step 1: propose a new Beneficiary address through the function proposeBeneficiary.

Step 2: through the function claimBeneficiary, the new Beneficiary address will be confirmed.

BXB-06 | Unreasonable upper boundary

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/Broker.sol: 685, 690	**Resolved**

Description

In the functions `setRepayInterestCut()` and `setAuctionCut()`, the admin can set `repayInterestCut`, `auctionPledgerCut` and `auctionDevCut` to be a too large value which will make most interest and premium transferred to the platform.

Recommendation

Consider adding a reasonable upper boundary.

Alleviation

The team heeded our advice and resolved this issue in commits eb097e6174deda87e1827ce28e5887ed04f793fa and 218796a852206f057d9c96b66969bb3ec17450c7.

BXB-07 | Gas consumption in loop

Category	Severity	Location	Status
Gas Optimization	Medium	contracts/Broker.sol: 256, 318, 483	Resolved

Description

In the ethereum, each transaction consumes a certain amount of gas, and the actual amount consumed is determined by the complexity of the transaction, the greater the number of loops, the higher the complexity of the transaction, and when the maximum amount of gas consumption allowed is exceeded, it will cause the transaction to fail.

In the functions `cancelPledge()`, `pledgerDeal()` and `lenderDeal()`, the call of `IERC20Upgradeable(usdxc).safeTransfer()` is used in a loop which will consume a large amount of gas consumption, which might lead to the call of function `cancelPledge()`/`pledgerDeal()`/`lenderDeal()` failed. It's recommended to provide withdraw function to allow users to get back their tokens.

Recommendation

Consider adding withdraw function and recording funds correctly.

Alleviation

The team replied: We have limited the max number of offer list , to prevent a large amount of gas consumption.

BXB-08 | Privileged ownership

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/Broker.sol: 673, 679, 685, 690	Acknowledged

Description

The owner of the contract `Broker` has the permission to:

- Admin can change the `redemptionPeriod` which is the time that pledger can repay his order via function `setRedemptionPeriod()`,
- Admin can change the `clearingPeriod` which is the time of auction via function `setClearingPeriod()`,
- Admin can change the `repayInterestCut` which will affect the cut of interest via function `setRepayInterestCut()`,
- Admin can change the `auctionPledgerCut` and `auctionDevCut` which will affect the cut of the profit of auction via function `setAuctionCut()`,
- Admin can change the `maxLendersCnt` of an order and the `defaultMaxLendersCnt` to limit the number of lenders via functions `setMaxLendersCnt()` and `setDefaultMaxLendersCnt()`,

without obtaining the consensus of the community.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multi-sig governing procedure and let the community monitor in respect of transparency considerations.

Alleviation

The team replied: After deploy the broker contract, we will transfer the ownership to a multi-sign contract.

BXB-09 | Logical Issue of `_profit` distribution

Category	Severity	Location	Status
Logical Issue	Informational	contracts/Broker.sol: 568	Partially Resolved

Description

According to the whitepaper, in case the auction price is higher than the lender's loan and the premium will be shared by the lender and the platform. However, the premium is shared by the lender, pledger and platform in the below codes.

```
568         uint256 _profit = _price.sub(detail.price).sub(detail.interest);
569         uint256 _beneficiaryCommissionOfInterest =
570             detail.interest.mul(repayInterestCut).div(
571                 MAX_REPAY_INTEREST_CUT
572             );
573         uint256 _beneficiaryCommissionOfProfit =
574             _profit.mul(autionDevCut).div(MAX_REPAY_INTEREST_CUT);
575         uint256 _beneficiaryCommission =
576             _beneficiaryCommissionOfInterest.add(
577                 _beneficiaryCommissionOfProfit
578             );
579         uint256 _pledgerCommissionOfProfit =
580             _profit.mul(autionPledgerCut).div(MAX_REPAY_INTEREST_CUT);
581
```

Can you please explain the intention here?

Alleviation

[XCarnival Team]:

We have changed this point. The premium will be shared by the lender, pledger and platform.

[Certik Response]:

Consider publishing a new version of whitepaper, and post a medium article or twitter publicly.

BXB-10 | External dependency

Category	Severity	Location	Status
Logical Issue	Minor	contracts/Broker.sol: 205	Acknowledged

Description

The value of address `usdxc` will be initialized by parameter. Although there is a mock file help understand the logic, it's still uncertain about the real value.

The scope of this audit would treat the external dependency entity as black boxes and assume the functional correctness.

Recommendation

Consider ensuring the address of `usdxc` correct.

BXB-11 | Risk of supported NFT digital assets

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	contracts/Broker.sol: 214~247	**Acknowledged**

Description

In function `pledge()`, there is no restriction on the type of collateral NFT assets to be pledged. However, it is hard for lenders to estimate the real-value of any type of NFTs.

Recommendation

Consider using a whitelist for NFT assets to be pledged in the contract to mitigate the potential risks.

Alleviation

The team replied: The lender should estimate the value by himself, and broker dapp will give some price source to help the lender make his offer.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.