makersplace

MakersPlace Security Analysis

by Pessimistic

This report is public.

Published: February 6, 2021

Abstract

In this report, we consider the security of smart contracts of MakersPlace project. Our task is to find and describe security issues in smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we considered the security of smart contracts of MakersPlace project. We performed our audit according to the procedure described below.

The initial audit showed an Overpowered role issue, a few issues of low severity (code quality and gas consumption issues). The project uses old Solidity version.

The documentation of the project is incomplete and needs structuring.

After the initial audit, the code base was updated to the latest version. Some of low severity issues were fixed. Also, developers provided comments for Overpowered role issue.

General recommendations

We recommend updating Solidity version and completing the documentation to the project.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether code logic corresponds to the specification.
2. Whether the code is secure.
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
  - We scan project's code base with automated tool SmartCheck.
  - We manually verify (reject or confirm) all the issues found by tools.
- Manual audit
  - We compare the project's code to Compound's codebase and check whether the changes are consistent with documentation.
  - We manually analyze new code for security vulnerabilities.
  - We assess overall project structure and quality.
- Report
  - We reflect all the gathered information in the report.

Project overview

Project description

In our analysis we consider smart contracts of MakersPlace project on private GitHub repository, commit a57d086b6371c381491a6a6e5475797b440abc96.

The scope of the audit included **GlobalBidCore.sol** file and its dependencies.

The documentation for the project was provided as a readme file in the repository and a separate text file, sha1sum is 043cd506cb89cbf92b63e95ebe890f7e7f0856a8.

Latest version of the code

After the initial audit, the code base was updated. For the recheck, we were provided with commit 52d992caf8e65ad69664c6e29d9e0ca9f5c750c3.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence project operation in current implementation. We highly recommend addressing them.

Overpowered role

The operator of **GlobalBidCore** contract has the following powers:

- The operator can cancel user's bids at a will using `cancelUserGlobalBid()` function. In such case, the user gets his/her bid back.
- The operator can change fees up to 100% of the price without any confirmation from the token seller. The seller of the token gets the bid after the fee is subtracted.

  Since the operator can then withdraw the fee, this may result in undesirable consequences for token sellers if the operator's private keys become compromised.

  We recommend designing contracts in a trustless manner or implementing proper key management, e.g. multisig.

> *Comment form developer: this is by design. We want to onboard special artists who demand 20%~50% royalties for the first 5~10 years of the artwork lifecycle. So, we need to have a backdoor to tweak the payout amount for certain tokens. So, we resorted to this design where we can change payout for any token. Building this the blockchain way with multisig might not be worth it since we do custodial wallets, so we fire the request on behalf of them anyway.*

Low severity issues

Low severity issues can influence project operation in future versions of code. We recommend taking them into account.

Code quality

There are the following issues in **GlobalBidCore** contract:

- Mappings `mediaIdToUserBidId` and `bidIdToMediaBid` at lines 37–39 have implicit visibility. We recommend declaring visibility explicitly.

  The issue has been fixed and is not present in the latest version of the code.

- `_getNextBidId()` internal function does not incapsulate `bidIdCounter` logic. We recommend removing this function and using `bidIdCounter` directly or updating the function so that it updates `bidIdCounter` value automatically.

  The issue has been fixed and is not present in the latest version of the code.

- `cancelUserGlobalBid()` function can be used by both the bidder and the operator (`On Behalf Of` mode) which complicates the code logic. We recommend splitting this function into two lightweight `external` functions (one for bidders, one for the operator) that both call an `internal` function with cancellation logic.

  The issue has been fixed and is not present in the latest version of the code.

- The behavior of `acceptGlobalBid()` function is confusing as it uses auction terminology, however its logic differs from auction:
  - A seller can accept any of available bids, not only biggest ones.
  - Multiple tokens can be sold if there are multiple bids.

- There are fixed values used in `setPayoutPercentage()` function at lines 188–189. We recommend using `HUNDREDPERCENT` constant to improve code readability, i.e. `HUNDREDPERCENT/100` instead of `100` and `HUNDREDPERCENT/2` instead of `5000`.

  The issue has been fixed and is not present in the latest version of the code.

Old Solidity version

The project uses old Solidity version `pragma solidity 0.4.25;`.

We recommend updating the project to use newer Solidity version, since it performs better optimization and has many new features that improve code stability and readability.

Gas consumption

Mappings `mediaIdToUserBidId` and `bidIdToMediaBid` at lines 37–39 of **GlobalBidCore.sol** are mostly used together. Therefore, they can be combined into single mapping to reduce gas consumption, i.e.

```
{contract => mediaId => bidder => bidId} + {bidId => GlobalBid}
```

mappings can be replaced with

```
{contract => mediaId => bidder => GlobalBid}
```

Comment form developer: we need to assign unique bid ids for every bid.

No particular documentation

The provided documentation is split across different sources and does not cover all the details of the implementation.

We recommend combining it into a single file with complete description of the project architecture and details of the implementation.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer
Boris Nikashin, Analyst
Alexander Seleznev, Founder

February 6, 2021