



Smart contracts security assessment

Final report

[Tariff: Standard](#)

Arbius Marketplace

October 2024



0xguard.com



hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	4
4. Known vulnerabilities checked	4
5. Classification of issue severity	5
6. Issues	6
7. Conclusion	11
8. Disclaimer	12

Introduction

The report has been prepared for **Arbius Marketplace**.

The project consists of upgradable contracts MarketplaceV1 and MarketplaceDataV1, used for jobs marketplace with an external escrow (Unicrow). The MarketplaceV1 is main contract responsible for job's workflow (creation, submission, assigning, finalization, disputing). The MarketplaceDataV1 contract is a helper that stores meta-information.

The code is available at the GitHub [repository](#) and was audited after the commit [d7bc73f96491380359eb0d49fd6efd8ce0ccda3a](#).

The audit scope excludes Unicrow contracts and interactions with Unicrow contracts.

Report Update.

The contract's code was updated according to this report and rechecked after the commit [77641cc762e58d814e4dcf2dd9152ffe33dd1a97](#).

Name	Arbius Marketplace
Audit date	2024-10-01 - 2024-10-28
Language	Solidity
Platform	Arbitrum Nova

Contracts checked

Name	Address
MarketplaceV1	
MarketplaceDataV1	
UnicrowTypes.sol	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
<u>Unencrypted Private Data On-Chain</u>	passed
<u>Code With No Effects</u>	passed
<u>Message call with hardcoded gas amount</u>	passed
<u>Typographical Error</u>	passed
<u>DoS With Block Gas Limit</u>	passed
<u>Presence of unused variables</u>	passed
<u>Incorrect Inheritance Order</u>	passed
<u>Requirement Violation</u>	passed
<u>Weak Sources of Randomness from Chain Attributes</u>	passed
<u>Shadowing State Variables</u>	passed

<u>Incorrect Constructor Name</u>	passed
<u>Block values as a proxy for time</u>	passed
<u>Authorization through tx.origin</u>	passed
<u>DoS with Failed Call</u>	passed
<u>Delegatecall to Untrusted Callee</u>	passed
<u>Use of Deprecated Solidity Functions</u>	passed
<u>Assert Violation</u>	passed
<u>State Variable Default Visibility</u>	passed
<u>Reentrancy</u>	passed
<u>Unprotected SELFDESTRUCT Instruction</u>	passed
<u>Unprotected Ether Withdrawal</u>	passed
<u>Unchecked Call Return Value</u>	passed
<u>Floating Pragma</u>	passed
<u>Outdated Compiler Version</u>	passed
<u>Integer Overflow and Underflow</u>	passed
<u>Function Default Visibility</u>	passed

Classification of issue severity

High severity

High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity

Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity

Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Rebasing payment tokens problem (MarketplaceV1)

Status: Open

Payment token is an arbitrary address provided by user. Rebasing tokens can cause locked funds problem or lack of liquidity problem due to discrepancy in marketplace contract's state (stored amounts) and token contract's state (actual balance).

Recommendation: Include whitelist for supported payment tokens or store each incoming payment in separate vault contract.

2. Incorrect outgoing ERC20 transfers (MarketplaceV1)

Status: Fixed

The function `safeTransferFrom` is used incorrectly for outgoing transfers, unless the allowance for itself is set to `type(uint256).max`. Affected functions: `updateJobPost`, `closeJob`, `withdrawCollateral`.

Recommendation: Use `SafeERC20.safeTransfer` for outgoing ERC20 transfers.

Medium severity issues

1. Incorrect documentation (MarketplaceV1)

Status: Fixed

The function `publishJobPost` supports any ERC20 token as payment token and 0x00 for ETH payment (according to the function description). But the payment handling is performed via SafeERC20 library from OpenZeppelin, and it doesn't support ETH transfers.

```
/**
 * @notice Publish a new job post
 * @notice To assign the job to a specific worker, set multipleApplicants_ to false
and add the worker to the allowedWorkers_. In such a case, title and description can be
encrypted for the worker
 * @notice The function will request a collateral deposit in the amount_ and token_
from the caller.
 * @param title_ job title - must be not null
 * @param contentHash_ short job description published on IPFS
 * @param multipleApplicants_ do you want to select from multiple applicants or let
the first one take the job?
 * @param tags_ labels to help the workers search for the job. Each job must have
exactly one of the labels listed above, and any number of other labels
 * @param token_ token in which you prefer to pay the job with - must be a valid
ERC20 token or 0x00..00 for ETH
 * @param amount_ expected amount to pay for the job - must be greater than 0
 * @param maxTime_ maximum expected time (in sec) to deliver the job - must be
greater than 0
 * @param deliveryMethod_ preferred method of delivery (e.g. "IPFS", "Courier")
 * @param arbitrator_ address of an arbitrator preferred by the customer
 * @param allowedWorkers_ list of workers that can apply for the job. Leave empty
if any worker can apply
 */
function publishJobPost(
    string calldata title_,
    bytes32 contentHash_,
    bool multipleApplicants_,
    string[] calldata tags_,
    address token_,
    uint256 amount_,
    uint32 maxTime_,
    string calldata deliveryMethod_,
    address arbitrator_,
    address[] calldata allowedWorkers_
) public returns (uint256) {
    . . .
}
```

Recommendation: Consider removing 0x00 for ETH from description or modify the corresponding transfers.

2. Unusual ERC20 tokens may be unsupported (MarketplaceV1)

Status: Fixed

Job's payment can be set in an arbitrary ERC20 tokens, so its transfers are made with SafeERC20 library. However, Unicrow outgoing transfers are made via pull-payment (approve/transferFrom), but some ERC20 tokens may require unusual methods for allowance.

Recommendation: Consider using the `SafeERC20.forceApprove`.

3. Creator can set affiliated arbitrator (MarketplaceV1)

Status: Open

Job creator can set an affiliated arbitrator to dispute and refund payment for the job. Arbitrator reputation in the MarketplaceDataV1 can be increased intentionally before malicious job is created.

```
function publishJobPost(
    . . .
    address arbitrator_,
    . . .
) public returns (uint256) {
    checkParams(title_, tags_, amount_, arbitrator_, msg.sender);
    . . .
}

function checkParams(
    . . .
    address arbitrator_,
    . . .
) internal {
    . . .
    require(
        marketplaceData.arbitratorRegistered(arbitrator_),
        "arbitrator not registered"
    );
    require(
        arbitrator_ != creator_,
        "arbitrator and job creator can not be the same person"
    );
    . . .
}
```


Recommendation: Consider adding additional precautions against spam.

Low severity issues

1. Possible math underflow (MarketplaceV1)

Status: Fixed

Possible underflow in `updateJobPost` function, increasing job amount can fail due to `collateral0wed > difference`.

```
function updateJobPost(
    . . .
) public onlyJobCreator(jobId_) {
    if (job.amount != amount_) {
        if (amount_ > job.amount) {
            uint256 difference = amount_ - job.amount;
            SafeERC20.safeTransferFrom(
                IERC20(job.token),
                msg.sender,
                address(this),
                difference - job.collateral0wed
            );
        }
        . . .
    }
}
```

Recommendation: Include explicit check with human-readable error.

2. Unused variables (MarketplaceV1)

Status: Fixed

The variable `meceShortForm` is not used in the `checkParams` function.

The variable `treasury` is not used in any of audited contracts.

The variable `pauser` is not used for any function of the MarketplaceV1 contract.

3. Possible overflow (MarketplaceDataV1)

Status: Fixed

The function `updateUserRating` can experience math overflow if the `reviewRating_` is large enough to increase `averageRating` beyond `type(uint16).max`.

Recommendation: Consider adding explicit check for input parameter or use the OpenZeppelin's SafeCast library for `averageRating` calculations.

4. Possible overflow (UnicrowTypes.sol)

Status: Open

The function `abs8` can fail due to overflow for `type(int16).min` argument.

Recommendation: Be cautious using this function, consider substitute it with OpenZeppelin's `SignedMath.abs()`.

Conclusion

Arbius Marketplace MarketplaceV1, MarketplaceDataV1, UnicrowTypes.sol contracts were audited. 2 high, 3 medium, 4 low severity issues were found.

1 high, 2 medium, 3 low severity issues have been fixed in the update.

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 0xGuard 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 0xGuard 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.



 Guard