SOLIDIFIED

Audit Report for Linum Labs - July 15, 2021

Summary

Audit Report prepared by Solidified covering the Mito smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief on 15 July 2021.

Audited Files

The source code has been supplied in the form of a GitLab repository:

https://gitlab.com/linumlabs/mito-admin/-/tree/master/contracts

Commit number: `d0ceea23a7c24088b66fea402e7d8778730f89e6`

The scope of the audit was limited to the following files:

```
contracts
├── Registry.sol
├── auctions
│   ├── AuctionHub.sol
│   ├── BaseAuction.sol
│   ├── BaseEdition.sol
│   ├── EditionData.sol
│   ├── FirstPrice.sol
│   ├── IAuction.sol
│   ├── IHub.sol
│   └── LimitedTimedEdition.sol
├── nft
│   ├── INft.sol
│   └── NFT.sol
└── royalties
    ├── IRoyalties.sol
    └── Royalties.sol
```

Intended Behavior

The smart contracts implement the smart contracts for an NFT marketplace with several auction models.

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

Issues Found

Solidified found that the Mito contracts contain no critical issues, 3 major issues, 3 minor issues, in addition to 4 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	Royalties.sol: Missing validations can cause loss of funds	Major	Pending
2	AuctionHub.sol: Auction with active lots can be removed or updated	Major	Pending
3	NFT.sol:Token transfer restrictions can be bypassed	Major	Pending
4	NFT.sol: Missing Event Emission	Minor	Pending
5	NFT.sol Token owner is not updated in all transfers	Minor	Pending
6	NFT.sol Duplicate tokens can be minted for invalid basic token	Minor	Pending
7	Compiler Version	Note	-
8	Consider using pull over push for payment	Note	-
9	NFT.sol: Allows inactive minters to mint duplicate tokens	Note	-
10	Code cleanup	Note	-

Critical Issues

No critical issues have been found.

Major Issues

1. Royalties.sol: Missing validations can cause loss of funds

The function `updateAddress()` is missing a few important validations. The method never checks if the `_newAddress` already has any existing royalty. Overwriting an existing address with royalty will make the funds to be lost forever and can never be withdrawn.

Furthermore, the method also allows the owner to use any address as input which gives the owner complete control over any account balance stored in the contract.

Recommendation
It is recommended to sum the existing and new royalty together and store rather than completely overwriting the existing royalty balance.

The method acts more like a backup plan but can be easily used by malicious owners to claim all royalty. It is recommended to either remove this method or inform the user beforehand if it's offered as a feature.

2. AuctionHub.sol: Auction with active lots can be removed or updated

The function `removeAuction()` and `updateAuction()` do not check if there are any existing active Lots. Any such removed auction will lock the token transferred to it permanently.

Recommendation
Consider checking if there are any lots already present in the auction before updating or removing it from the auction hub.

3. `NFT.sol`:Token transfer restrictions can be bypassed

`NFT.sol` implements checks that `onlyAuctions` can transfer the tokens. However these checks can be easily bypassed by interacting with `safeTransferFrom()` and `safeBatchTransferFrom()` functions implemented in base `ERC1155.sol` contract.
This issue is related to issue 5.

Recommendation
Consider overriding the method in the inherited contract to make sure the restriction is checked.

Minor Issues

4. `NFT.sol`: Missing Event Emission

The function `_changeOwner()` does not emit an event, even though a change of ownership should be looked at according to a TODO comment.

Recommendation
Emit missing event

5. `NFT.sol` Token owner is not updated in all transfers

The function `transfer()` updates the current owner for every transfer. But the methods `safeTransferFrom` and `safeBatchTransferFrom` inherited from the `ERC1155` contract are exposed as public methods and can be called by anyone to transfer the token - which does not update the current owner in the `NFT` contract.

Recommendation
Consider overriding those methods in the inherited contract to track the ownership change in all cases.

6. NFT.sol Duplicate tokens can be minted for invalid basic token

The function `batchDuplicateMint()` does not validate whether the base token is a valid one or not. This allows the duplicate minter to mint any token with an invalid base token id, resulting in invalid creator address for those.

Recommendation
Consider checking the validity of the base token id before allowing the minter to mint new tokens.

Informational Notes

7. Compiler Version

The codebase locks the compiler version to 0.7.3. In general it is good practise to lock the version pragma to a specific version. However, Solidity version 0.7.3 contained several compiler bugs, including a security relevant code generation bug that can lead to data corruption when copying empty arrays to storage.

Recommendation
Consider using compiler version 0.7.4 or 0.7.5

8. Consider using pull over push for payment

The function `_insecureHandlePayment()` sends the sale amount to the seller in the same method. This allows the seller to annoy/manipulate the system to some extent. For example, the seller can maintain a whitelist of their own or make the buyer pay more gas.

Recommendation
Consider using the existing royalty module to pay the seller as well.

9. NFT.sol: Allows inactive minters to mint duplicate tokens

The modifier `onlyBatchDuplicateMinter()` does not check if a minter is active and this allows an invalid minter to mint new duplicate tokens by calling the method `batchDuplicateMint()`. Whilst this can only happen after calling the `updateMinter()` with a duplicate minter address, it introduces a potential source of error.

Recommendation
Consider checking if a minter is active before allowing the address to mint new tokens.

10. Code cleanup

Consider cleaning up the code based on the following recommendations.

1. Remove the `Testable.sol` import from all contracts before deploying to the main net.
2. `BaseAuction.sol`: Function `_isLotInBiddableState` the condition `status != IHub.LotStatus.AUCTION_CANCELED` is redundant.
3. `AuctionHub.sol`: misspelled `actionAddress_` contract variable
4. `NFT.sol`: Unused variable `circulatingSupply_`
5. `NFT.sol`: Remove hardhat `console.log` import.
6. The code contains several TODOs. It is recommended to implement or remove them.
7. `BaseEdition.sol`: Remove duplicate code in L163 - L166
```
   auctionHubInstance_.lotAuctionCompleted(auctionID_, _lotID);(
         auctionID_,
         _lotID
      );
```
8. `NFT.sol`: Includes unreachable code in `_isValidCreator` method. The `else-if` path of the condition statement is redundant, because it will never be executed.
9. `AuctionHub.sol`: function `getOwner()` has exactly the same functionality as `owner()` inherited from `Ownable.sol` base contract.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Linum Labs or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.