SOLIDIFIED

Audit Report for Mito - March 24, 2022

Summary

Audit Report prepared by Solidified covering the Mito Ethereum smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief on 28 February 2022.

Audited Files

The source code has been supplied in the form of two GitLab repositories:

https://gitlab.com/rungie/mito-admin-contracts

Commit number: `ef890681f9a59f3474eff4cc1e83b07976def148`

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

```
contracts
├── IRegistry.sol
├── Registry.sol
├── marketplace
│   ├── BaseAuction.sol
│   ├── BaseEdition.sol
│   ├── EditionData.sol
│   ├── HubVentas.sol
│   ├── IAuction.sol
│   ├── IHub.sol
│   └── sale\ type
│       ├── Ofertas.sol
│       ├── Subasta.sol
│       ├── Subasta_v2.sol
│       ├── VentaDirecta.sol
│       ├── VentaPorCopias.sol
│       └── VentaPorCopias_buyLimit.sol
├── nft
│   ├── INft.sol
│   ├── INft_Mito.sol
│   ├── NFT.sol
│   └── NFT_Mito.sol
├── others
│   ├── BridgeNFT.sol
│   └── Rewards.sol
└── royalties
    ├── IRoyalties.sol
    └── Royalties.sol
```

Intended Behavior

The smart contracts implement ERC-1155-based NFT and a number of related sales and auction contracts.

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 in 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	Medium	Code readability is hindered by a mixture of English and Spanish naming.
Level of Documentation	Medium	-
Test Coverage	Medium	Unit test coverage does not seem to extend to the latest development iteration. Non-existent files are still referenced in the tests, whilst renamed versions are not covered.

Issues Found

Solidified found that the Mito contracts contain no critical issues, 1 major issue, 7 minor issues in addition to 5 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	BaseAuction.sol and BaseEdition.sol: Creator approval requirement means that auctions for a token might not be possible	Major	Acknowledged
2	Subasta.sol and Subasta_v2.sol: Admin's ability to set bid counters could lead to lost funds	Minor	Fixed
3	MitoNFT.sol: Admin's ability to set auction IDs could lead to duplicate/overwritten token IDs	Minor	Fixed
4	MitoNFT.sol: Admin's ability to set auction IDs could lead to duplicate/overwritten token IDs	Minor	Fixed
5	Dependencies with well-known security vulnerabilities in the build system	Minor	Acknowledged
6	HubVenta.sol: Auction with active lots can be removed or updated	Minor	Acknowledged
7	BaseEdition.sol: Users can overpay with funds getting stuck	Minor	Fixed
8	NFT_Mito.sol: transfer functions might result in incorrect token owner information	Minor	Fixed
9	Compiler Version	Note	-
10	NFT.sol and NFT_Mito.sol: Allow inactive minters to mint duplicate tokens	Note	-
11	Oferta.sol: Unused Modifier	Note	-
12	Subasta.sol, Subasta_V2.sol, VentaDirecta.sol,	Note	-

	VentaPorCopias_buyLimit.sol and VentaPorCopias.sol: Inconsistent Documentation		
13	NFT.sol and NFT_Mito.sol: Inconsistent Documentation	Note	-

Critical Issues

No critical issues have been identified

Major Issues

1. `BaseAuction.sol` and `BaseEdition.sol`: Creator approval requirement means that auctions for a token might not be possible

The functions `_winner_primary_sell` and `_bid` rely on the token creator having approved the contract. If this action has not been performed, the auction will fail. However, if an auction was to be used in a situation where the seller was not the original creator (re-sale), this may not be possible.

Recommendation
Consider adapting the logic, so that the seller does not rely on the creator's approval.

Minor Issues

2. `Subasta.sol` and `Subasta_v2.sol`: Admin's ability to set bid counters could lead to lost funds

The function `setBidCounter` allows the admin user to set bid counter to an arbitrary value. Since bid counters are incremented, this could lead to bids being overwritten and funds being lost.

Recommendation
Consider removing the ability to allow setting the bid counter. An alternative is to add checks to ensure the new auction ID is higher than the previous last ID.

3. MitoNFT.sol: Admin's ability to set auction IDs could lead to duplicate/overwritten token IDs

The function `setLastTokenId` allows the admin user to set the token last token id to an arbitrary value. Since token ids are assigned incrementally, this could lead to token IDs being overwritten.
Whilst this is only an issue if the admin assigns an already used ID, this may occur accidentally.

Recommendation
Consider removing the ability to set the token id counter.

4. HubVenta.sol: Lot counters and auction counters can be set to any value

The admin user can set `lotCounter_` and `auctionCounter_` to any value. This means that lots and auctions could be overwritten. These are used to assign lot ids and auction ids and setting them to non-incremental might lead to auctions and lots being overwritten.
Whilst this is only an issue if the admin assigns these values incorrectly, accidental misuse could lead to loss of funds.

Recommendation
Consider removing the ability to set lot numbers and auctions counters manually.

5. Dependencies with well-known security vulnerabilities in the build system

The build and test system relies on a number of outdated JavaScript libraries with well-known security vulnerabilities, some of which are critical. Since these are only used for deployment and testing they do not constitute a smart contract security risk. However, outdated dependencies in the build system make operational security incidents, such as key leakage more likely.

Recommendation
Consider Using `npm audit` to identify vulnerable dependencies and update them

6. HubVenta.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.

7. BaseEdition.sol: Users can overpay with funds getting stuck

The function `_bid()` accepts a `msg.value` higher than the price for direct sales. Any leftover ETH will be stuck in the contract.

Recommendation
Consider refunding excess amounts or requiring exact amounts.

8. NFT_Mito.sol: transfer functions might result in incorrect token owner information

Functions `safeTransferFrom()` and `safeBatchTransferFrom()` might result in incorrect token owner information.
The recipient (`to` parameter) could do a recursive `NFT` transfer to another address when handling `onERC1155Received`/`onERC1155BatchReceived` post transfer hook.
Subsequently, the `tokens_[_tokenID].currentOwner` would be set to an incorrect value once the post transfer hook has been executed.

Recommendation
Consider updating `tokens_[_tokenID].currentOwner` before calling `super.safeBatchTransferFrom`/`super.safeTransferFrom` functions.

Informational Notes

9. Compiler Version

The codebase uses Solidity version 0.7.5. However, this implies relying on a safe math library, since compiler versions lower than 0.8 did not implement automatic overflow protection. Using automatic overflow protection could significantly simplify the code. In addition, a number of important compiler bugs have recently been fixed. Moreover, any compiler version below 0.7.6 would not flag some kinds of source code attacks.

Recommendation
Consider using a compiler version greater than 0.8.4 or 0.7.6

10. `NFT.sol` and `NFT_Mito.sol`: Allow 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.

11. `Oferta.sol`: Unused Modifier

The modifier `onlyTokenOwner` is never used.

Recommendation
Consider removing the unused modifier.

12. Subasta.sol, Subasta_V2.sol, VentaDirecta.sol, VentaPorCopias_buyLimit.sol and VentaPorCopias.sol: Inconsistent Documentation

The documentation for the function `createLot` is incorrect. The documentation of the function states that "Only the Auction Hub is able to call this function", but in fact, `HubVentas` never calls this function. This function is called by an account which has previously requested a new auction lot.

Recommendation
Consider correcting the documentation or adjusting the implementation.

13. NFT.sol and NFT_Mito.sol: Inconsistent Documentation

The documentation for the functions `safeTransferFrom` and `safeBatchTransferFrom` do not list all parameters and seem to be a leftover from a previous version of the implementation.

Recommendation
Consider correcting the documentation.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of TrustToken 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 from legal and financial liability.

Oak Security GmbH