TITLES Publishing Protocol Report

Performed by: *0xShiki*

Table of Contents

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

Audit Scope Details

- In Scope:

```
src/TitlesCore.sol
src/editions/Edition.sol
src/fees/FeeManager.sol
src/graph/TitlesGraph.sol
src/shared/Common.sol
```

- Solc Version: 0.8.20
- Chain(s) to deploy contract to: Ethereum
- ERC20s:
  - USDC
  - DAI
  - LINK
  - WETH

Roles

ADMIN_ROLE (Trusted) => Granted by the deployer to internal, trusted addresses only.

- On TitlesCore, this role can:
  - Change the ERC-1155 Edition implementation contract to an arbitrary address (setEditionImplementation). No post-auth validation is performed.
  - Upgrade the contract to an arbitrary new implementation (via _authorizeUpgrade, inherited and overridden with auth check from Solady's UUPSUpgradeable)
- On TitlesGraph, this role can:

  - Create new Edges at will (createEdges). No post-auth validation is applied, except the typical uniqueness checks.
  - Upgrade the contract to an arbitrary new implementation (via _authorizeUpgrade, inherited and overridden with auth check from Solady's UUPSUpgradeable)
  - Grant or revoke any role to/from any address (grantRole, revokeRole).

- On FeeManager, this role can:
  - Set the protocol fees (setProtocolFees). All fees are constrained to a constant range.
  - Create or change a fee route for any work within any Edition (createRoute). This is the only way to change the fee route for a work after publication.
  - Withdraw any funds locked in the contract (withdraw). This is the only way to withdraw funds from the contract.
  - Grant or revoke any role to/from any address (grantRole, revokeRole).

EDITION_MANAGER_ROLE (Restricted)

- On an Edition, this role can:
  - Publish a new work with any desired configuration (publish). This is the only way to create new works after the Edition is created.
  - Mint promotional copies of any work (promoMint). There are no limitations on this action aside from the work's supply cap and minting period.
  - Set the Edition's ERC2981 royalty receiver (setRoyaltyTarget). This is the only way to change the royalty receiver for the Edition.
  - Grant or revoke any role to/from any address (grantRole, revokeRole).

EDITION_MINTER_ROLE (Restricted)

- On an Edition, this role can:
  - Mint promotional copies of any work (promoMint). There are no limitations on this action aside from the work's supply cap and minting period.

OTHER ROLES WHICH DON'T HAVE SPECIFIC ROLE IDs:

- Editions have an Ownable owner who can:
  - Mint promotional copies of any work (promoMint). There are no limitations on this action aside from the work's supply cap and minting period.

  - Grant or revoke EDITION_PUBLISHER_ROLE to/from any address (grantPublisherRole, revokePublisherRole).
  - Manage the ERC1155 contract in typical ways (e.g. transfer ownership). Notably, the owner CANNOT manage roles other than EDITION_PUBLISHER_ROLE.

- Works within an Edition have a creator who can:

  - Update the minting period for the work (setTimeframe). This is the only way to change the minting period for a work after publication.
  - Set the fee strategy for any work within the Edition (setFeeStrategy). This is the only way to change the fee strategy for a work after publication. The fee strategy is validated by the Fee Manager, and the final strategy (which may have been modified during validation) is applied immediately.
  - Set the metadata for their own works. This is the only way to change the metadata for a work after publication.
  - Transfer full ownership of the work to a new address (transferWork). This is the only way to change the creator for a work.

- FeeManager has an Ownable owner (essentially synonymous with ADMIN_ROLE, held by TitlesCore) who can:

  - Set the protocol fees (setProtocolFees). All fees are constrained to a constant range. This role is granted to the TitlesCore contract whose currently scoped version does not have a mechanism for leveraging this permission directly.
  - Create or change a fee route for any work within any Edition (createRoute). This is the only way to change the fee route for a work after publication.

Issues found

Severtity	Number of issues found
High	0
Medium	1
Low	0
Gas	0
Info	0
Total	1

Findings

[M-1] `msg.value` is consumed in the first iteration of a for-loop in mintBatch function, preventing follow-up iterations from executing

Relevant GitHub Links

```
https://github.com/sherlock-audit/2024-04-titles/blob/main/wallflower-contract-v2/src/editions/Edition.sol#L287
```

Description

`Editions` are ERC1155 contracts, in which each tokenId represents an individual work that has been published through the TITLES protocol. `mintBatch` function is used to mint to a `receiver` different amounts of different tokenIds to a user. However, when the `FeeManager` collects the mint fee for minting, it consumes the `msg.value` in the first iteration of the loop, which can exhaust the available ETH balance after the first iteration.

```
    function mintBatch(
        address to_,
        uint256[] calldata tokenIds_,
        uint256[] calldata amounts_,
        bytes calldata data_
    ) external payable {
        for (uint256 i = 0; i < tokenIds_.length; i++) {
            Work storage work = works[tokenIds_[i]];

            // wake-disable-next-line reentrancy
@>          FEE_MANAGER.collectMintFee{value: msg.value}(
                this, tokenIds_[i], amounts_[i], msg.sender, address(0)
                    , work.strategy
            );

            _checkTime(work.opensAt, work.closesAt);
            _updateSupply(work, amounts_[i]);
        }

        _batchMint(to_, tokenIds_, amounts_, data_);
        _refundExcess();
    }
```

Impact

Disrupts the expected behavior of the smart contract function logic, preventing minting different tokenIds to a user, and also preventing the `FeeManager` from collecting mint fees for every tokenId.

Proof of Concept

Paste this test in `Edition.t.sol` file:

```
    // In the `setUp` function
    edition.publish(
        address(2),
        10,
        0,
        0,
        new Node[](0),
        Strategy({
            asset: address(0xEeeeeEeeeEeEeeEeEeEeeEEEeeeeEeeeeeeeEEeE),
            mintFee: 0.01 ether,
            revshareBps: 2500,
            royaltyBps: 250
        }),
        Metadata({
            label: "Best2 Work2 Ever2",
            uri: "ipfs.io/best2-work2-ever2",
            data: new bytes(0)
        })
    );

    feeManager.createRoute(edition, 2, new Target[](0), address(0));
    .
    .
    .
    .
    .
    // Paste test
    function test_mintBatchMultipleWorksToReceiverReverts() public {
        uint256[] memory tokenIds = new uint256[](2);
        tokenIds[0] = 1;
        tokenIds[1] = 2;
        uint256[] memory amounts = new uint256[](2);
        amounts[0] = 1;
        amounts[1] = 1;
        edition.mintBatch{value: 0.0212 ether}(address(1), tokenIds,
           amounts, new bytes(0));
    }
```

The logs of the test show that the function reverts after the first iteration. If you divide the `msg.value` in the `mintBatch` function by 2, and run the test again, the test will pass.

Tools Used

Manual Review

Recommended Mitigation

Consider collecting the fees for each tokenId after minting all the tokenIds to the user.