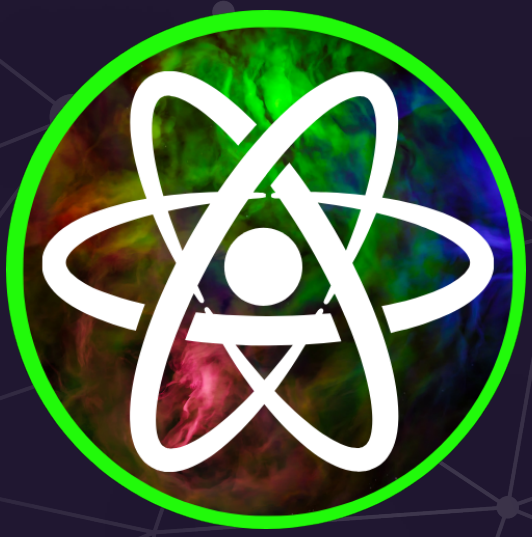
SB SECURITY

E280 NFT

Security Review



May 16, 2025

Contents

1. About SBSecurity

SBSecurity is a duo of skilled smart contract security researchers. Based on the audits conducted and numerous vulnerabilities reported, we strive to provide the absolute best security service and client satisfaction. While it's understood that 100% security and bug-free code cannot be guaranteed by anyone, we are committed to giving our utmost to provide the best possible outcome for you and your product.

Book a Security Review with us at sbsecurity.net or reach out on Twitter @Slavcheww.

2. Disclaimer

A smart contract security review can only show the presence of vulnerabilities **but not their absence**. Audits are a time, resource, and expertise-bound effort where skilled technicians evaluate the codebase and their dependencies using various techniques to find as many flaws as possible and suggest security-related improvements. We as a company stand behind our brand and the level of service that is provided but also recommend subsequent security reviews, on-chain monitoring, and high whitehat incentivization.

3. Risk classification

	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1. Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - leads to a moderate loss of assets in the protocol or some disruption of the protocol's functionality.
- **Low** - funds are not at risk.

3.2. Likelihood

- **High** - almost **certain** to happen, easy to perform, or highly incentivized.
- **Medium** - only **conditionally possible**, but still relatively likely.
- **Low** - requires specific state or **little-to-no incentive**.

3.3. Action required for severity levels

- High - **Must** fix (before deployment if not already deployed).
- Medium - **Should** fix.
- Low - **Could** fix.

4. Executive Summary

Overview

Project	E280 NFT
Repository	Private
Commit Hash	b8afe75952080e59e4dcd5d7d60243c483684d34
Resolution	7e4ac12b6daf7e8b4c2532c6c711661d27f22959
Timeline	May 14, 2025 Mitigation: May 16, 2025

Scope

ElmntNftTransformer.sol
E280NFT.sol
E280Vault.sol

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	1
Low/Info Risk	1

5. Findings

5.1. Medium severity

5.1.1. Smart wallets and contracts will lose their bridged NFTs

Severity: Medium Risk

Description: `ElmntNftTransformer::_transform` doesn't give ability to users to specify a `receiver` address on the chain they're bridging their tokens. As a result, some accounts, such as smart wallets (old gnosis safe versions doesn't use `create2` to deploy at the same address across the chains), smart contracts and account abstraction wallets, will end up losing their tokens, since their address on the destination chain might belong to someone else.

```
function _transform(uint256[] memory tokenIds, uint32 destination) internal {
    uint256 amount = tokenIds.length;
    uint8[] memory tiers = new uint8[](amount);
    IElmntNft nft = IElmntNft(ELMNT_NFT);
    IERC20 token = IERC20(ELMNT);

    for (uint256 i = 0; i < amount; i++) {
        tiers[i] = nft.getNftTier(tokenIds[i]);
        nft.safeTransferFrom(msg.sender, address(this), tokenIds[i]);
    }
    nft.redeemNFTs(tokenIds);
    token.safeTransfer(ELMNT_TARGET, token.balanceOf(address(this)));

    bytes memory payload = abi.encode(msg.sender, tiers);//<--- msg.sender used
    bytes memory options = _buildOptions(amount);
    MessagingFee memory fee = _quote(destination, payload, options, false);
    if (fee.nativeFee > msg.value) revert InsufficientFeeSent();

    MessagingReceipt memory receipt = _lzSend(
        destination,
        payload,
        options,
        MessagingFee(fee.nativeFee, 0),
        payable(msg.sender)
    );

    uint256 excessFee = msg.value - fee.nativeFee;
    if (excessFee > 0) {
        (bool success, ) = msg.sender.call{ value: excessFee }("");
        if (!success) revert RefundFailed();
    }

    emit NftTransform(receipt.guid, amount);
}
```

Recommendation:

1. Extend `batchTransform` to expect address `receiver` as an argument.

2. Use the `receiver` when constructing the payload in `_transform`.

Resolution: Fixed

5.2. Low/Info severity

5.2.1. Informational issues and code suggestions

Severity: Low Risk

Description:

1. `currentCycle` comment references Stax.

2. instead of passing `msg.sender` as argument to `_processTokenIdBurn`, get it from inside.

3. typo in `totalRewadsPaid`, must be `RewaRDs`.

4. `InsufficientFeeSent` isn't needed as `_payNative` does the verification.

5. Unclaimed `ELMNT` rewards are locked forever when NFT is bridged. (User mistake, but can add popup on the front end)

Resolution: Fixed