Preliminary Notes: Dexi-NFT

Brewlabs Services Hub

Summary: Dexi-NFT is a contract that allows traders to buy and bid (via auction) on NFTs. It is written in Solidity and compiled using build 0.9.0.
Audit package: Standard
Deployment Status: Not Deployed
Files Audited: DexiNFT.sol. All imported files for this contract were not audited.
Audit Result: **PASSED**

Disclaimer: *This audit documentation is for discussion purposes only. The scope of this audit was to analyze and document Dexi-NFT smart contract codebase for quality, security, and accuracy. This audit guarantees that your code has been revised by an expert.*

Overview of Audit

No critical or major security issues identified.
1 medium security issue identified.
Several minor/informational issues identified.

Security issues

Medium level security issues detected:

1. Violation of Check-Effects-Interaction pattern with state changes made in Buy and Sell functions after calls to an external contract are made. The contract sets the mapNFTAttribute[_id].listed value after transfers are made to an external contract which can potentially lead to reentrancy attacks, however risk is mitigated through modifier logic. Recommendation is to
    1. Maintain _validateSell and _validateBuy modifiers on functions
    2. Resolve all effects to the state of the contract prior to interacting with external contracts (move the mapNFTAttribute[_id].listed declaration before the transfers are executed).

Minor/Informational issues detected:

1. No upper limit for feePercent state variable leading to potential centralisation risk if contract owner sets the fee to 100%. Recommendation is to embed an upper limit, or ensure the Dev Team is doxxed or consider (Know Your Customer) KYC services.
2. Missing events for changes to state variables in setFee and setBulkMintBaseUrl functions. This means only the contract owner can alter state variables without the community being aware of changes. Recommendation is to add events for all changes to state variables to increase community transparency.
3. Lack of input validation for feeAddress which could lead to unexpected loss if an invalid address is set, for example, address(0).
4. Floating pragma set. If relying on byte-code level verification, best practice is to lock the compiler version prior to deployment.

Continued,

Code Style:

1. Revise contract for grammar eg. Line 243 return message: "Can not several bid” is not clear.
2. UInt256 set to 0 by default. Lines 178, 189 and 190 could be declared as just: uint256 private bulkMintLimit; to reduce gas usage.
3. bulkMint does not handle return variable from mintFixed. Dev team to confirm is return variable needs to be managed or not. Consider removing the return variable if it does not need to be managed.
4. Multiply before divide in lines 394, 395, 437 and 438 to avoid rounding errors.
5. Use of more comments to improve readability is encouraged.

Conclusion: The Brewlabs team thank you for the opportunity to review and audit your smart contact code. The data from this report will be formalised in the audit publication for your community. Keep in touch as we offer discounts for repeat business and on a range of other services!

Peach & Maverick
The Brewlabs Team.