NFTCHANCE AUDIT
Hagia

Issues:

[HagiaCurations.47] `setPrice`'should' update one at a time as there may be times where one would be updated without the other.

[HagiaCurations.52] `getSaleStatus` returns the same thing as adding public to the Struct declaration. (May be in-house reason this is needed.)

[HagiaCurations.61] `setToken` is slightly 'dangerous'. Would like a way to make sure that the token id is not changed while there is a pending transaction that will be processed. (Again, unless intended.)

[HagiaCurations.66] `_claimToken` does not validate the token id so someone 'could' mint a token different than expected if the id was changed.

[HagiaCurations.67] tx.origin == msg.sender 'can' be bypassed. If it is expected that there will ever be high competition I would more strictly find a way to do this. This does also immediately prevent any Gnosis/Vault contracts being able to partake in the minting.

[HagiaCurations.72] Move this above the minting function. While _mint() is used and does not allow for re-entrancy, it is best practice to have the effects in the proper order. (Checks, Effects, Interactions Pattern)

[HagiaCurations.88] There is no total supply control here. General expectation is that infinite minting cannot take place, though this is a personal preference.

[HagiaCurations.0] Right now the implementation is .03 ETH for both prices but the functionality is built around them having different prices. If not actually different prices, will save a little money to remove.

[HagiaCurations.0] The general token implementation is rather limiting, though again, that may be fine. A more standard 1155 Token Type implementation would look like:

```
  function createToken(
    uint64 _maxSupply,
    string calldata _uri,
    uint128 _priceWei,
    bytes32 _root,
    bool _paused
  )
    public
    onlyOwner
    returns (uint256)
  {
    require(bytes(_uri).length > 0, 'URI required');
    require(_maxSupply > 0, 'Max supply must be more than 0');
    uint256 _tokenId = _currentTokenId;
    _currentTokenId++;

    Tokens[_tokenId] = Token(
           0,
           _maxSupply,
           _priceWei,
           _root,
           _paused,
           _uri
    );
    emit URI(_uri, _tokenId);
    return _tokenId;
  }
```

[HagiaCurations.0] If you desire to burn these tokens exists, it would be best practice to enter a controlling mechanism to do so.

[RaffleGraphy.43] Private will have no effect as the value can still be retrieved from the transaction when it was set.

[RaffleGraphy.57] index can just be derived from the current length of an array.

[RaffleGraphy.63] `setRoyalty` should be first called in the constructor.

[RaffleGraphy.100] tx.origin can be bypassed if this is a high-competition drop and there is enough incentive for someone to do it.

[RaffleGraphy.109] Naming conventions used in this function are odd, but I “think” everything is functioning the way it is intended.

[RaffleGraphy.109] The sale status could change while a transaction is pending and could result in a different amount being minted than anticipated by the minter.

[RaffleGraphy.111] By using a Merkle Proof you have already verified that they are someone that you want minting. Thus, the origin check is not needed. Could be moved to public mint only, but the issue of tx.origin still remains.

[RaffleGraphy.0] Minting without creators initialized errors out due to division by 0. This means that the initial stack of creators should be set in the constructor.

[RaffleGraphy.133] Held funds in contract does not have any impact on the active state. No need to close it until the sale has been finished.

Notes:

[CHANCE.P1] Requesting special functionality from OpenSea could not only take a lot of time, but be an interesting hurdle to get stuck on in the future. As we move forward into a more decentralized and fragmented market it may be best to design your support around the larger scope / broader market. Personally, I have pivoted from focusing on OS support, to more direct 'aggregate' support as that is the inevitable market future for high value and volume.

[CHANCE.P2] For this creator functionality, every single person that mints is paying additional money for something that is only supported on one marketplace.