# 577.https://stackoverflow.com/questions/69783556/is-using-nonreentrant-modifier-with-a-payable-function-causes-errors

**T:**Is using nonReentrant modifier with a payable function causes errors?

**Q:**I created an ERC-721 contract that has a mint function that is payable. I have used a nonReentrant modifier with it which is found in the Openzeppelin contracts under Renterancy. Does this cause errors?  
  
Does payable invoke the nonreenterant modifier?

**C1:**Please edit the question to limit it to a specific problem with enough detail to identify an adequate answer.

1 **Answer**

**A1:**The OpenZeppelin nonReentrant modifier (link) prevents the reentrancy attack (link).  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
But it does not affect the function state mutability (such as payable).  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]

**C1:**The reason I'm asking is because I have deployed a contract with a mint function that is payable and added the reenterant modifier on it. The function was working normally, then it stopped, giving enormous gas when a user tries to use that function! This is the contract: etherscan.io/address/…

**C2:**The linked implementation seems all right. Can you specify what tokenId and msg.value was the sender passing, and what was the tokenPrice value at the time? My guess is that any of the require() conditions (within the mintBinaries() function or its dependencies) was failing, which would cause the transaction to revert... It's tricky to calculate gas requirement of a reverting tx (the wallet doesn't know if it's reverting because of a condition or because of not supplying enough gas) so some wallets are trying to spin up the gas recommendation all the way up to the available limit.

**C3:**TokenID is 523 for example (but it is any token ID really, as only IDs 8500, 2345, 8, 88, 888, 1, and 5585 got minted) and the msg.value is 0.01. The tokenPrice is 10000000000000000 wei (0.01 ETH) which haven't changed since the contract creation. I don't understand why this issue came up when some users minted successfully before! Also I have another function called ownerClaim that allows me to mint without paying, it works fine for the same tokenID mentioned above (523). So the problem is with either the extra require statement (which wasn't an issue before) or the modifier.