overflow-22.04

First, I ran checksec and determined that all security measures were enabled. Therefore, I could not use that to my advantage. I inspected the code and came up with the following exploit. First, I allocated 2 large spaces in memory. They were allocated such that they were adjacent. Then, I deleted the second allocation and overflowed from the first allocation to the second to see what was stored in the second allocation. This leaked a glibc address for me, after which I was able to calculate glibc base. Next, I used tcache poisoning to leak a stack address. First, I allocated 2 spaces in memory and deleted them so that I would have 2 items in fastbins. Then, I overflowed from one of the adjacent allocated memory spaces to one of the deleted memory sections to read the protected heap address that would give me the start of the heap. I unencrypted this address and got the front of the heap. Next, I overflowed from an adjacent memory allocation to overwrite (with the address of environ) the forward pointer of the later deleted fastbin item. I could determine what to write as the forward pointer because I could figure out the address of the chunk whose pointer I wanted to overwrite (due to me having the start of the heap). Using this information, I could give an encrypted forward pointer. After that, I reallocated items so that both fastbin items were used and read the last item I reallocated. When reading this item, whatever is at environ's address is printed. Thus, I was able to leak a stack address. From this stack address, I could figure out where rip would be. Finally, I can use tcache poisoning once again to write at the address of rip. Since I can write at rip, this means I can use ROP gadgets to spawn a shell. Thus, this challenge was completed.