School

I solved this challenge using a combination of static and dynamic analysis. First, I ran checksec and determined that NX is disabled, which means that instructions on the stack can be executed. Furthermore, stack canary is disabled, which means that we could try to use a buffer overflow attack. Also, we know the address of the start of the stack because the program prints it out for us. Now, we can write assembly that will give us the shell by running execve with the following arguments: a pointer to "/bin/sh", null (0), and null (0). We can turn this assembly into machine code and put it on the stack so that we can execute it. This will take up 37 bytes, and the remaining 3 bytes can be filled with '\x00's. We have overwritten the stack and the saved rbp at this point. Now, we can overwrite the return address to point to the start of the stack. This will cause the program to go to the start of the stack and execute our shellcode, giving us the shell. Thus, this challenge is solved.