PIE TIME

The challenge is based on using the leaked function address, as well as the relative offsets, to exploit the Position Independent Executable (PIE) binary. What is PIE, you ask? PIE is an executable file feature that randomizes the base address of the program every time it is executed. In other words, the addresses of the functions within the program will keep changing every time it is executed. However, the relative offsets between functions will always remain the same. In other words, the difference between the addresses of any two functions will always remain the same. The program is taking advantage of this feature. It is printing the address of the main function. It is then prompting the user to input an address to jump to. This is possible only if the address is correctly calculated.

First of all, the binary was run on the local system. The offsets of the important functions in the binary were found out using the nm command. The address of the win function was 0x12a7, whereas the address of the main function was 0x133d. However, the difference between the two addresses was always constant at -0x96.

When the remote service was established through the netcat command, the program printed the runtime address of the main function. Due to the randomization of memory addresses through ASLR, the leaked address of the main function was different every time the program was run. With the help of the previously calculated offset, the runtime address of the win function could be determined by subtracting the value of 0x96 from the leaked address of the main function. This calculation had to be performed every time the program was run, as the address of the win function from the previous execution would result in a segmentation fault.

After the correct address of the win function was determined, the address was entered into the program, and the program proceeded to the win function and printed the flag, marking the success of the exploit. The most important thing learned from this problem is that although the PIE protection mechanism prevents the execution of the exploit, the presence of a memory leak ensures the success of the exploit. Therefore, the only thing required is to think in terms of offsets instead of addresses, and the exploit would have been successful every time the program was run.