**Lockbox**

I solved this challenge using static and dynamic analysis. For my solution to be effective, it's imperative that no canary is detected, as canaries would stop buffer overflow attacks. Furthermore, it is necessary that PIE (Position Independent Executable) is disabled, as an enabled PIE ensures that functions are not loaded into the same addresses each time, making it difficult to jump to different functions with a buffer overflow. I checked both requirements with checksec. If we examine the assembly, we can see that 0x40 bytes of space is allocated to the stack. We also know that there is a saved RBP 8 bytes long before this allocated space. So, we know that we must overwrite 0x48 bytes to get to the return address. We also know that we need to overwrite the return address with the address of the win function, since that will allow us to win. Now, if we look at the assembly, we can see that RBP-0x8 is a pointer to the third 8-byte item we send in through gets. We can also see that RBP-0x10 is a pointer to the fourth 8-byte item we send through gets. We also know that when we try to win (by jumping into the win function), we will need the key to be 0x00000000daddb0dd. Right now, the key is a different value. Furthermore, we know that, at the end of main, the program will overwrite the value that the third 8-byte item we send points to with the value of the fourth 8-byte item we send. Thus, we can send a pointer to the key as our third 8-byte item and 0x00000000daddb0dd as our fourth 8-byte item to get the flag. Thus, this challenge is solved.