**Boffin**

I solved this challenge using static 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 main, we can see that a string of 32 bytes is allocated space on the stack. We also know that there is a saved RBP 8 bytes long before this allocated space. We can overwrite all these values by sending 40 (0x28) "A"s into gets. Now that we have reached the return address, we can overwrite the current return address with the address of give\_shell, which gives us the shell. Thus, this challenge is solved.