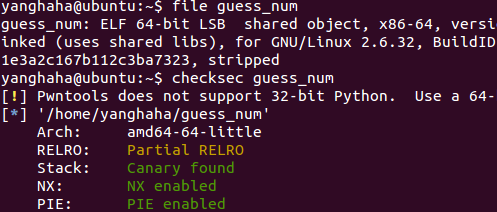
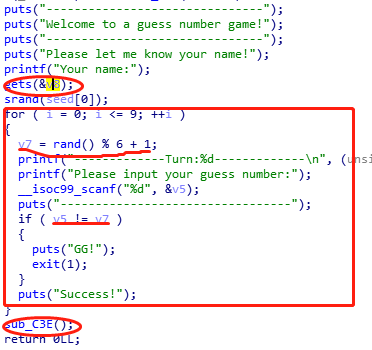
题目：

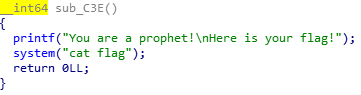


首先检查文件类型：

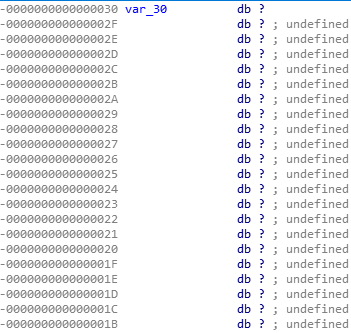


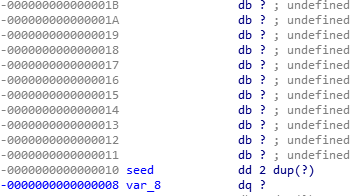
64位ELF文件，该有的保护都开了，用IDA看看源码，进去之后找到main函数：





如上图所示，这里的关键问题在于for循环的执行，v7每次产生一个随机数要保证与v5相等，只有顺利执行十次for循环才能最终调用sub\_C3E函数，该函数是获取flag的关键。因此这里的关键是想办法修改随机数种子，让每次生成的数字一样。而这里又开了栈保护，不能直接溢出到sub\_C3E函数，因此，我们来看看变量v8：





很幸运的发现v8与seed之间内存连续且相差了0x20字节，因此我们可以通过覆盖随机数种子从而控制随机数的产生。这里学习了一下随机数生成函数rand()与srand()的联系，简单概括为：**随机函数生成的随机数并不是真的随机数，他们只是在一定范围内随机，实际上是一段数字的循环，这些数字取决于随机种子。在调用rand()函数时，必须先利用srand()设好随机数种子，如果未设随机数种子，rand()在调用时会自动设随机数种子为1**。

有了这样的理论基础和前面的面的偏移量，便可以通过指定随机数种子，然后通过rand()函数模拟随机数产生并发送给接收方，这样就能保证我们的输入与程序自身产生的随机数一致。

因此，这里指定随机数为“1”，构造payload=‘a’\*0x20+p64(1)，利用的脚本如下：



最后就可以获取flag：

**收获：**进一步学习了栈溢出的题目，开了基础的保护措施。绕了个弯于是顺带温习了随机数的产生原理！