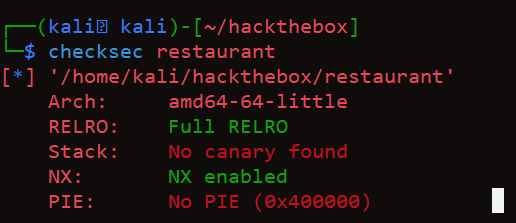
Writeup for the HACK THE BOX ROOM called “Restaurant”

This is indeed a challenging room in the HTB.

Please find the steps we can follow to exploit this room

Step1:

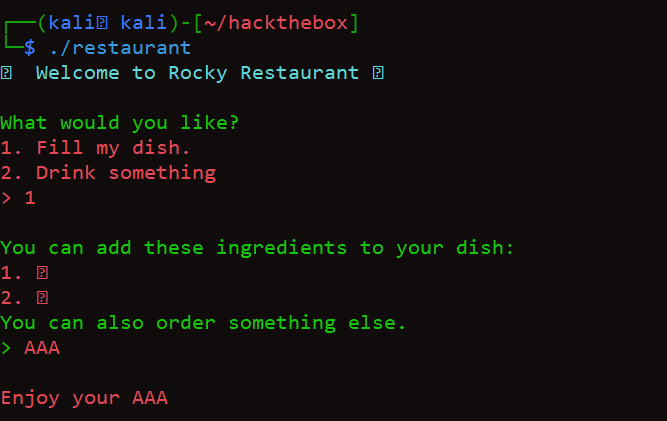
Look for the security properties of the binary



NX enabled – So stack is not executable, No use of Shellcode, Easiest way to overcome this is by overwriting GOT or using ret2libc

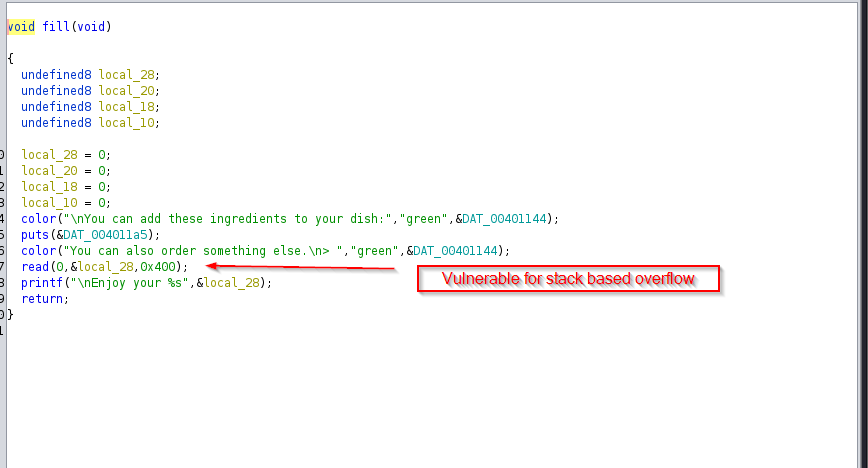
Full RELRO – Cant overwrite GOT because it is read only for Full RELRO , so have to use ret2libc

Step2 : Check the program if we can use any input to do stack based buffer overflows

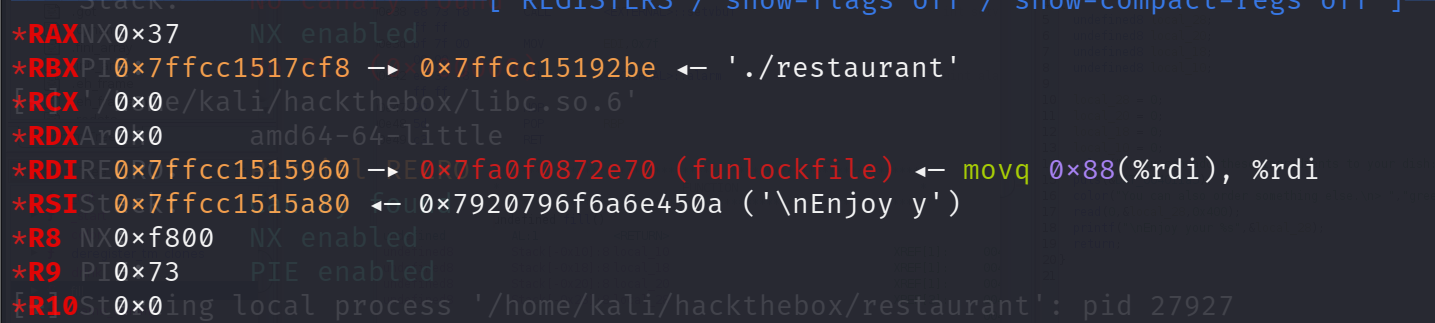


Step3: It takes the input from the user and displays, seems like there is a print function prone to format string vulnerability

Step4: Analyze the code using GHIDRA



Step5: set the breakpoint at fill+162 and check it via GDB



Step6: We could see the Code although vulnerable to format string vulnerability , there is nothing on the stack or registers to display , Ideally , this string prints Enjoy your %s and this %s points to RDI

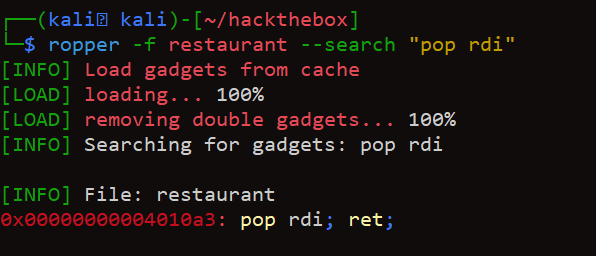
Step7: As ASLR is enabled on the remote machine we need to leak the address onto RDI so that the base address of libc is calculated

Step8: I have framed a payload to pop the GOT address of puts to RDI and the base address of libc is calculated from that

Step9: The first payload to leak the GOT address of puts function is as follows

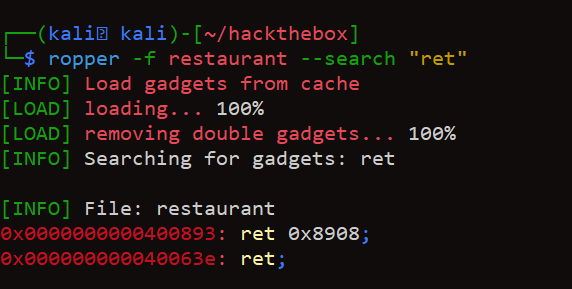
40\*b’A’ ------------------🡪 40 bytes of overflow, that fills till the return pointer of the vulnerable function mentioned above.

POP RDI Gadget address , Have used “ropper” to find this out

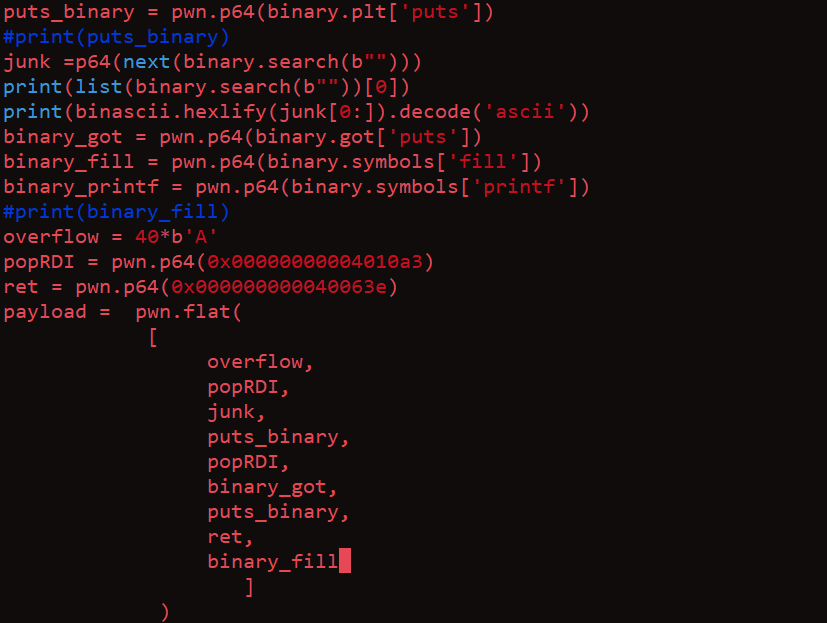


Step10: We need to find the libc version used on the remote machine, however in this task as the libc information is provided we can mention that libc version in pwntools.

Step11: Address of ret gadget is used for stack alignment



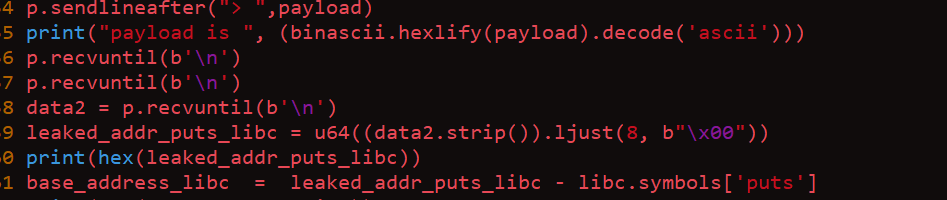
Step12: Even before popping the GOT address of puts, we need to pop newline to separate the GOT address from overflow. Puts() function with null string prints a newline. So, the payload is framed as below.



* In the above payload, we have added a newline before plt address of puts which in turn executes puts(“\n”).
* Then puts GOT value is given and the PLT of PUTS is called to resolve the offset of GOT
* Ret is used for stack alignment as the payload is less than 16 bytes
* Fill() function is called to continue with the next exploitation

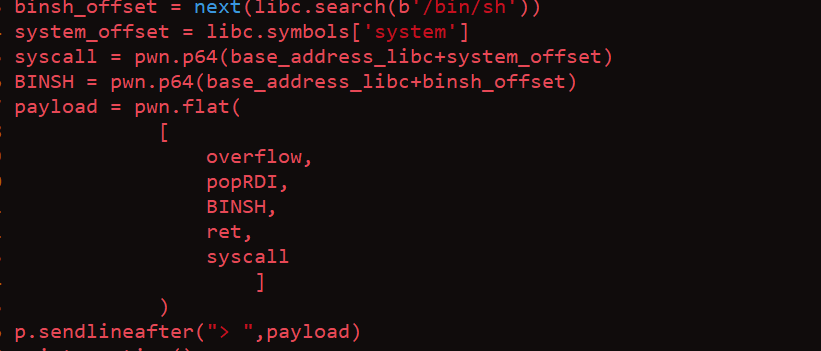
Step13:

Calculating the libc base address

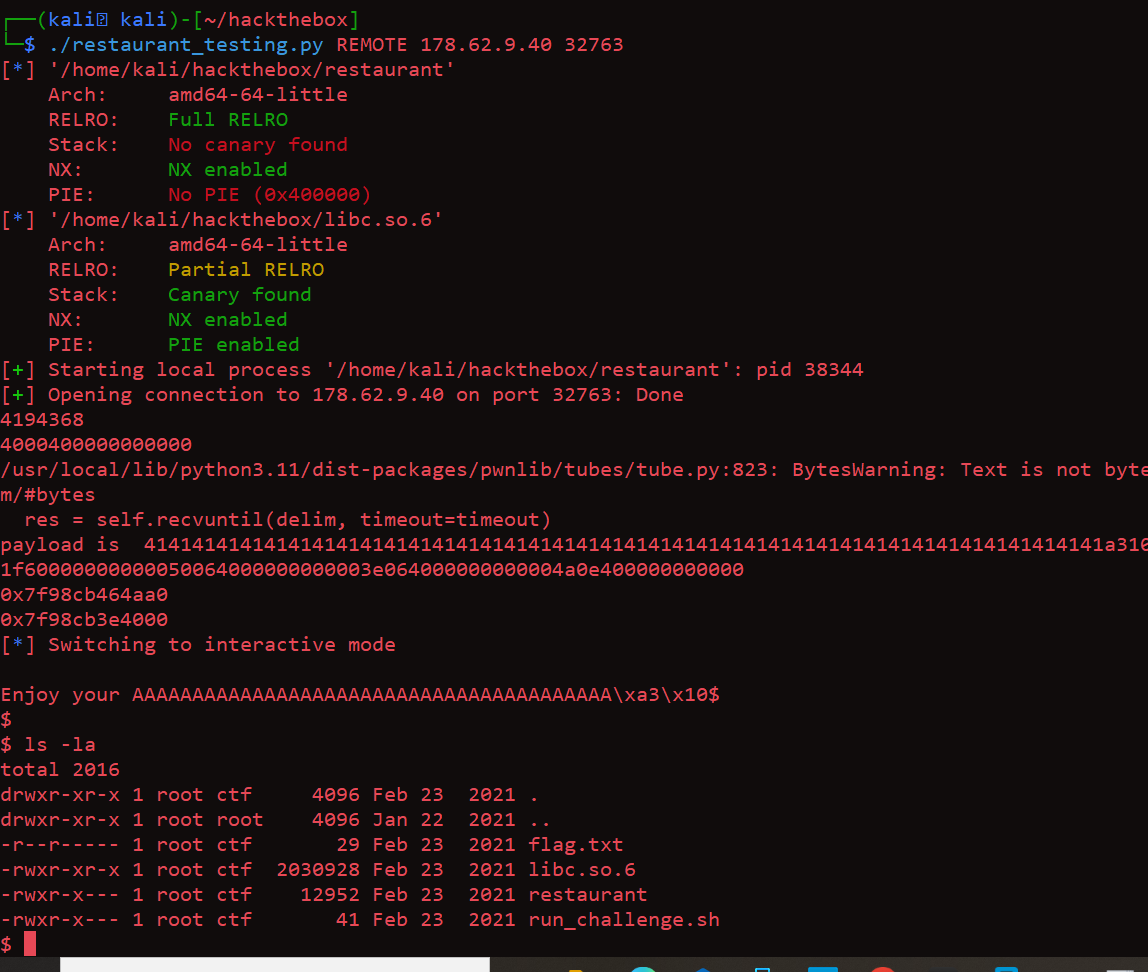


The Base address of LIBC is calculated by subtracting the puts offset address from leaked GOT address of PUTS

Step14: Forming the second payload



* Offset of BINSH is added with the libc base address to find the virtual address of binsh
* Offset of system is added with the libc base address to find the virtual address of system function
* Used the ROP chain to override the return pointer with the system function that spawns the shell



* The shell is obtained

