**Q:** Try to change the functionality of printf. Give success and failure scenarios (if any)

**Sol:**

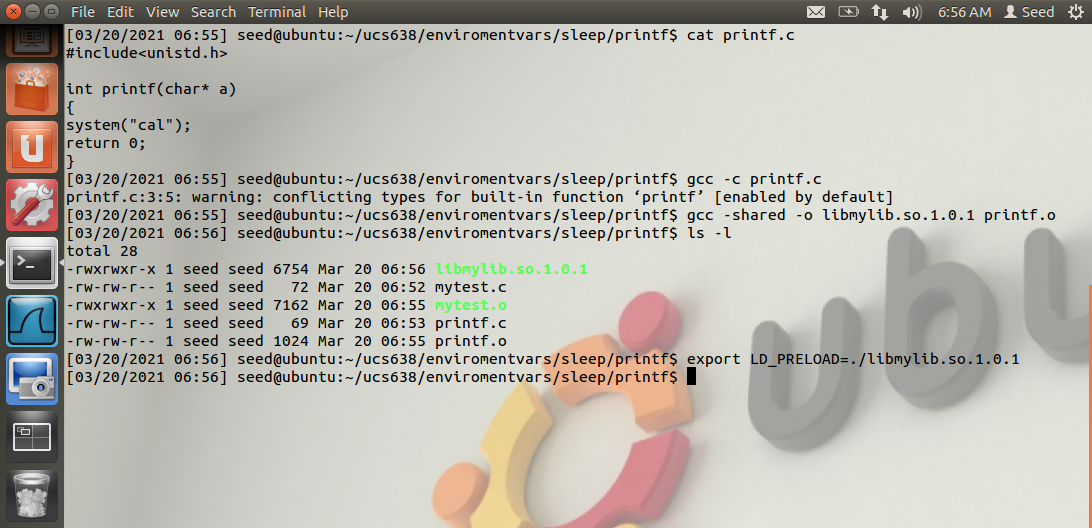
*Successful Attempt*

The following program (mytest.c) simply calls the printf function, which is present in libc.so, the standard libc shared library.



When we compile the above program, by default, the printf function is dynamically linked. Thus, when this program is run, the dynamic linker will find the function in the libc.so library. The program will print for the specified text as expected.

Molded printf function created by Attacker



Attacker needed to compile the above code, create a shared library, and add the shared library to the LD\_PRELOAD environment variable.



After that, if we run our previous mytest program again, we can see from the above result that attacker’s printf function is invoked instead of the one from libc.



If we unset the environment variable, everything goes back to normal.

*Unsuccessful Attempt*

The following program (mytest.c) simply calls the printf function, same as previous program. The only difference is of ‘\n’ at the end of string argument in printf function.



Following same procedure as previous program, this one is also compiled and run, as displayed above.

Molded printf function created by Attacker



Above procedure is same as previous one. Attacker needed to compile the above code, create a shared library, and add the shared library to the LD\_PRELOAD environment variable.



After that, if we run our mytest program again, we can see from the above result that attacker’s printf function is not invoked at all, but the one from libc is (as it should have in normal).



Although, the shared library has been added to the LD\_PRELOAD environment variable successfully, the attack is not successful.

The only difference made is ‘\n’ delimiter at the end of string argument, passed in printf function.