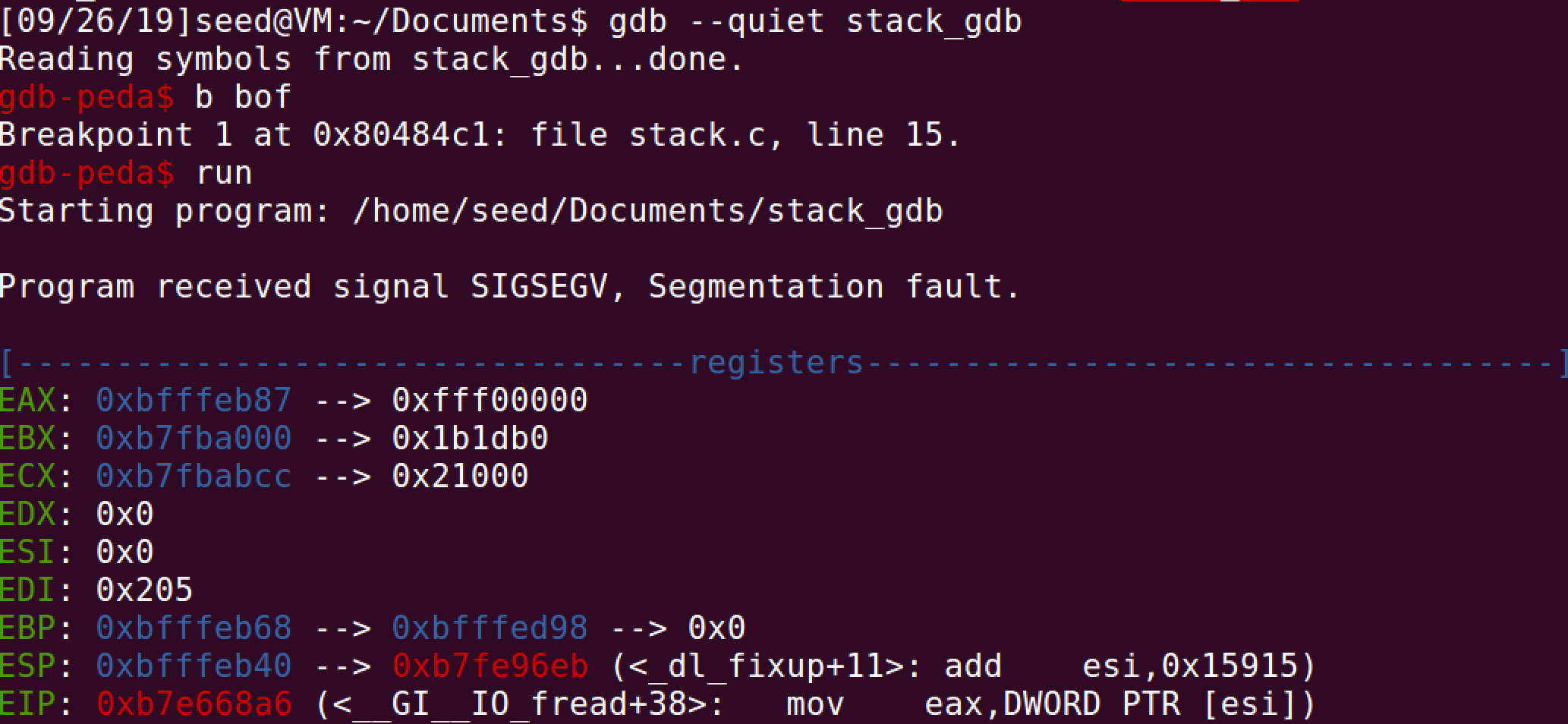
Lab3 Buffer\_overflow

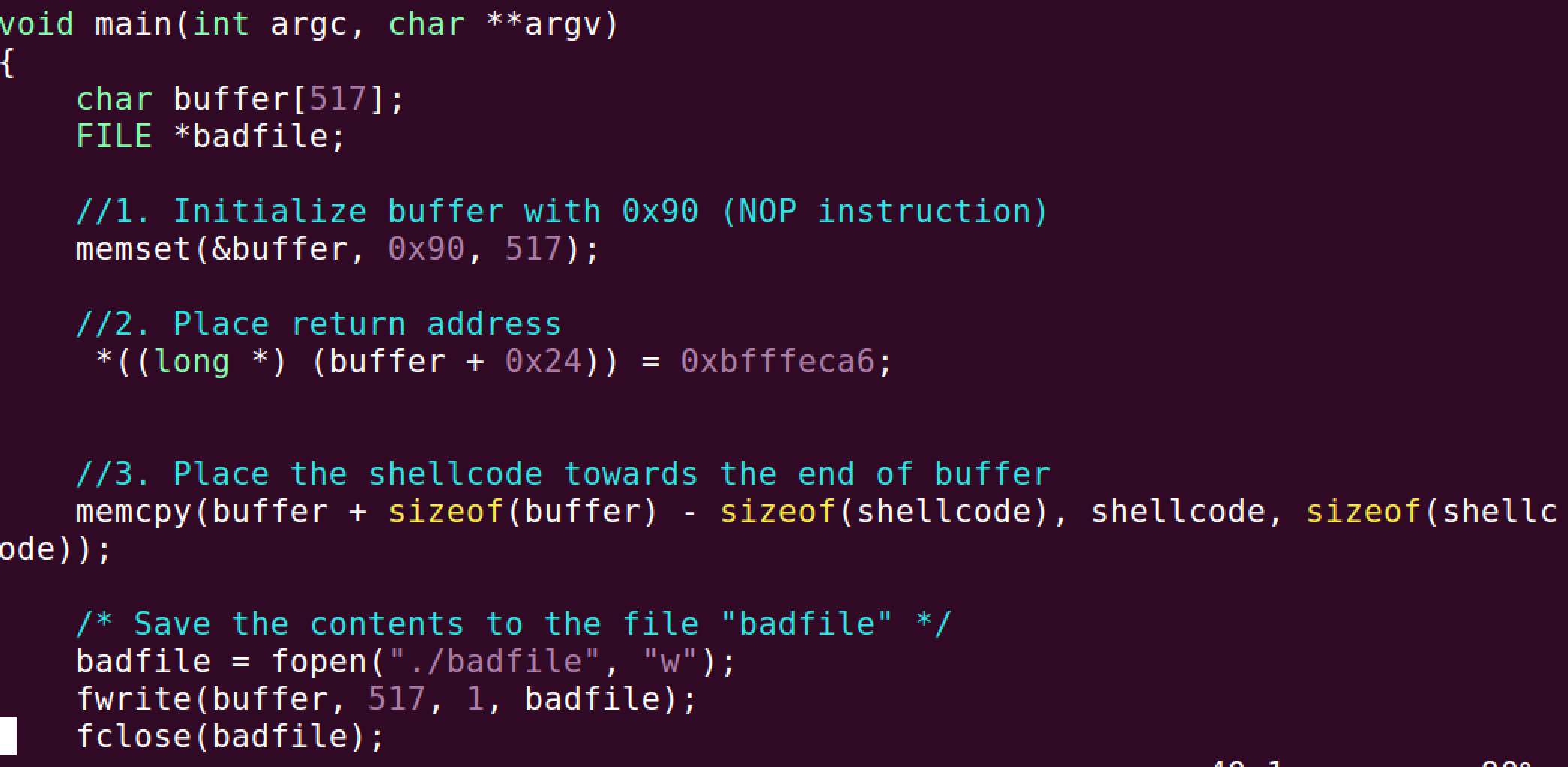
By Marco.

**Task1.**

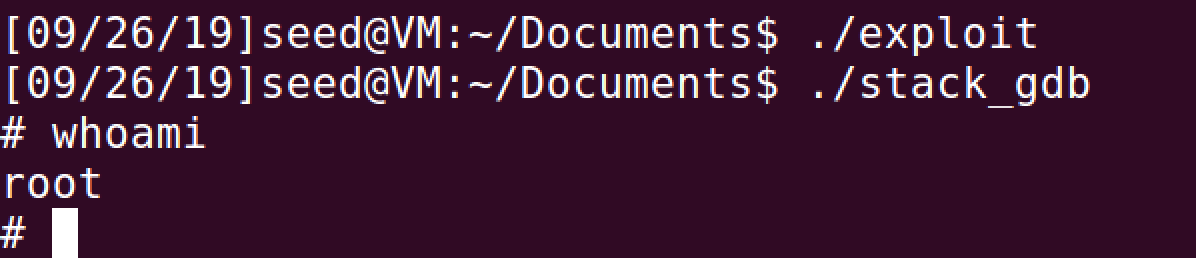
Check point: The original input can have a maximum length of 517 bytes, but the buffer has only 12 bytes long. Because strcpy() does not check boundaries, buffer overflow will occur.

Compile stack.c file using the command gcc stack.c -o stack\_gbd -g -z execstack -fno-stack-protector. Then, use GDB tool to get address



From the GDB, we flowed the command to get the address of buffer and $edp, then modified the exploit.c to give the return address.

Executed the exploit to get badfile, and the run stack\_gdb to get the root shell



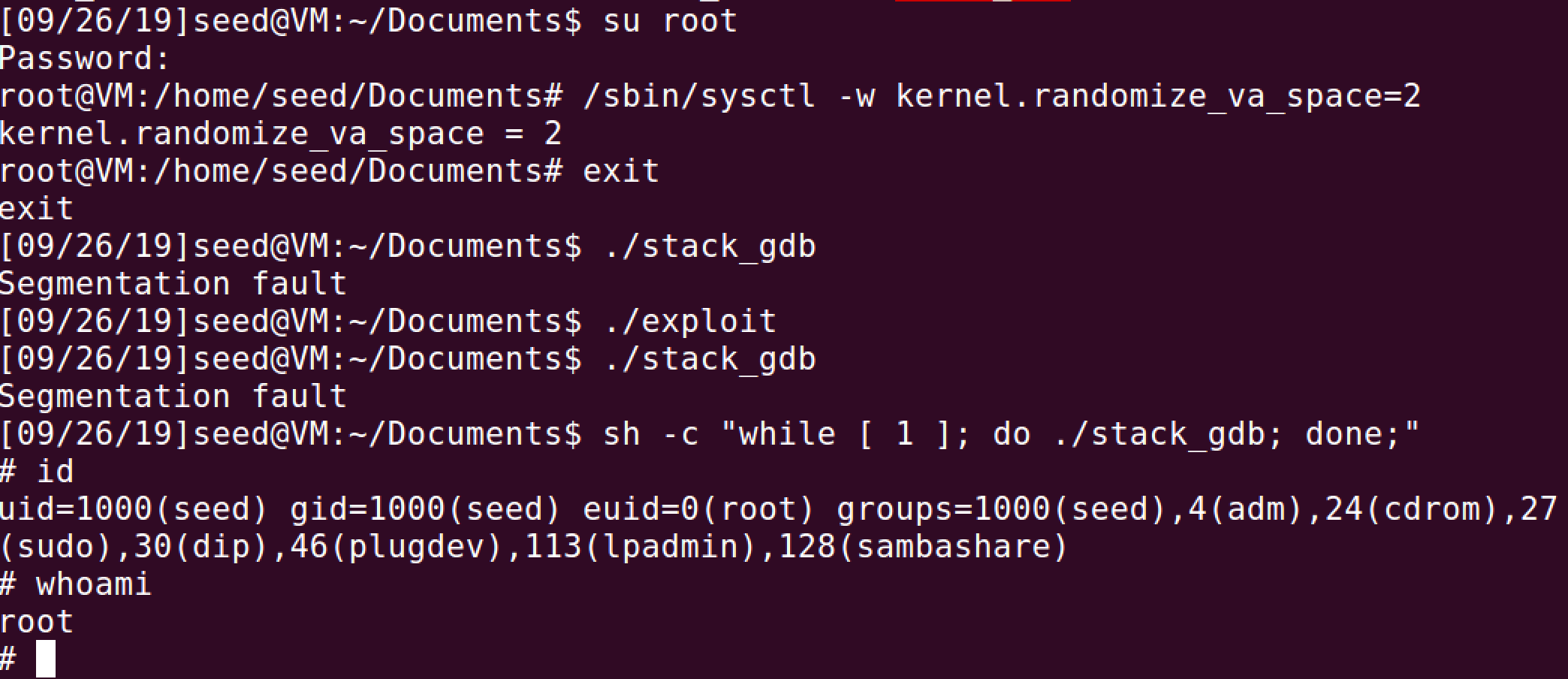
The badfile looks like this:



After we finish the above program, compile and run it. This will generate the contents for "badfile". Then run the vulnerable program stack. If our exploit is implemented correctly, we should be able to get a root shell.

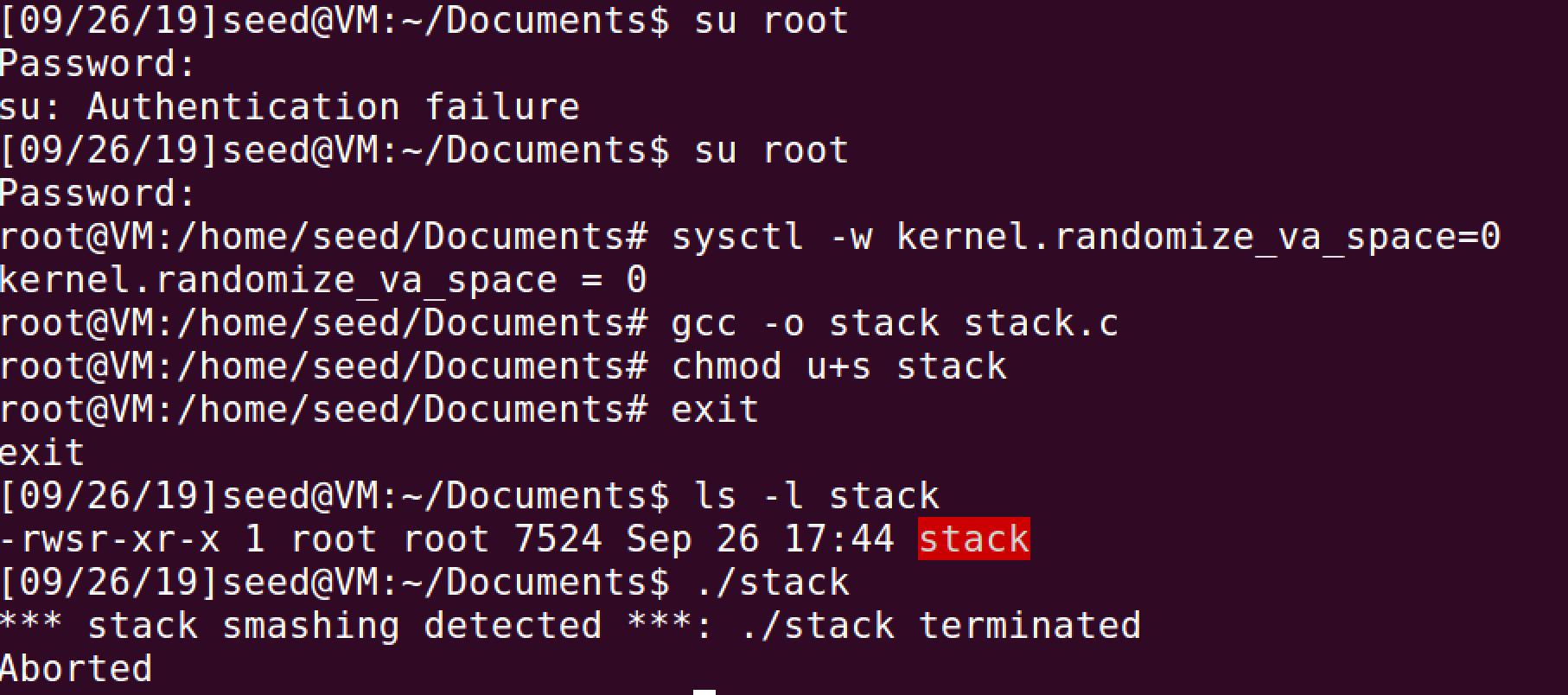
**Task2.**

Set /sbin/sysctl -w kernel.randomize\_va\_space=2, and then run stack\_gdb file for many time, we can get a root shell



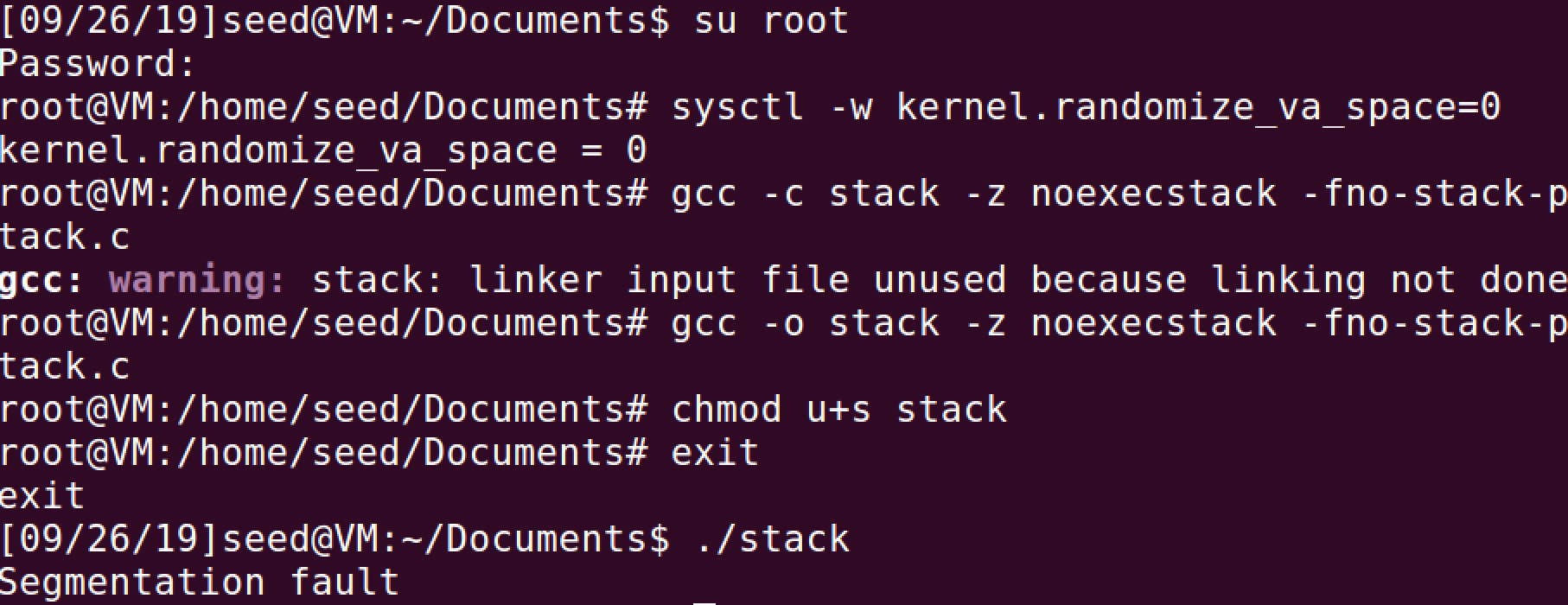
**Task3.**

Set the kernel.randomize\_va\_space=0 first, compile the file stack again without -fno-stack-protector, it showed that the file smashing detected.

****

**Task4**

We used noexecstack command to compile stack and try to run it:



We can’t execute stack because non-executable stack makes it impossible to run shellcode on the stack.