INFORMATION SECURITY

LAB 4

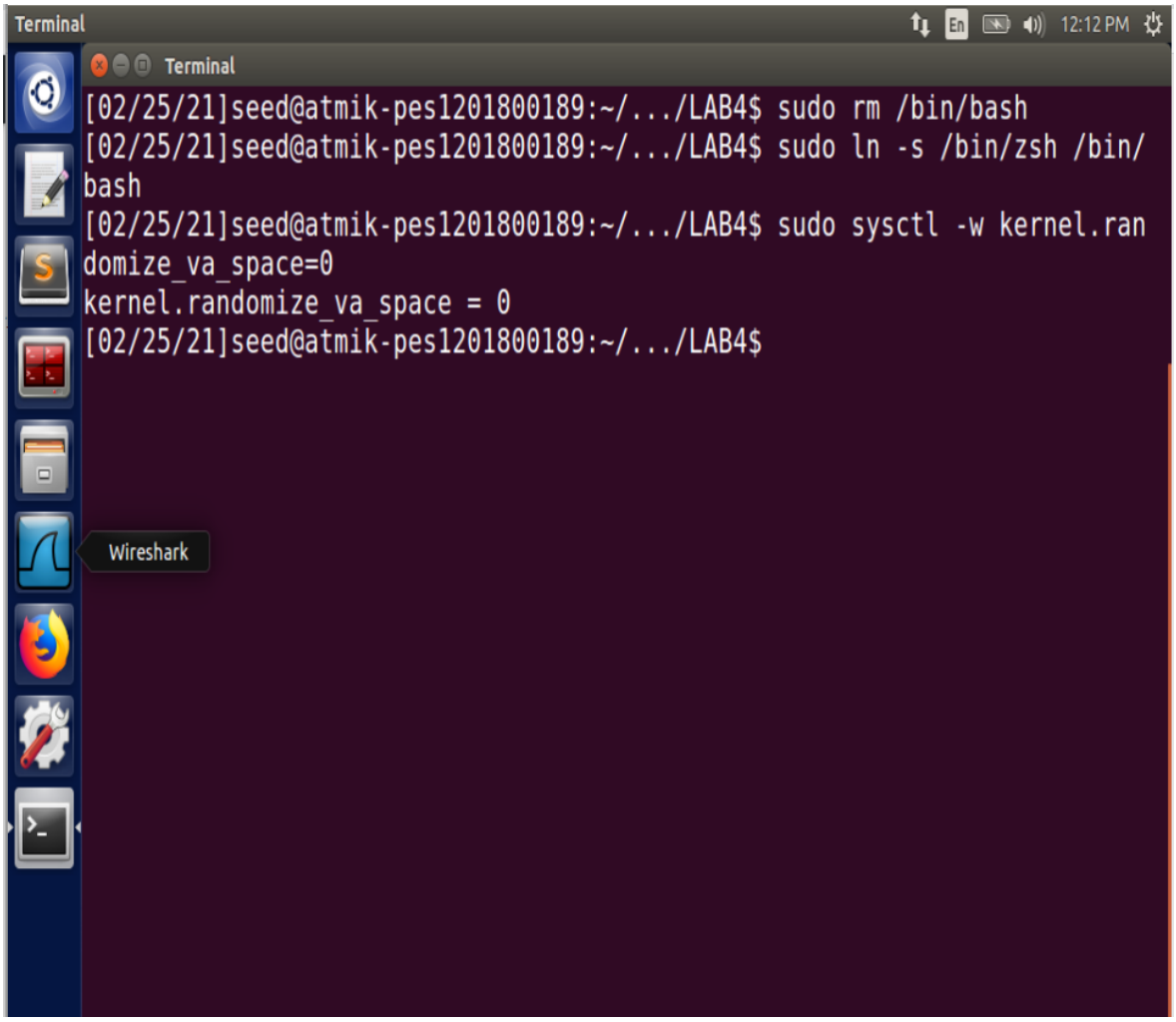
Name: Atmik Ajoy

SRN: PES1201800189

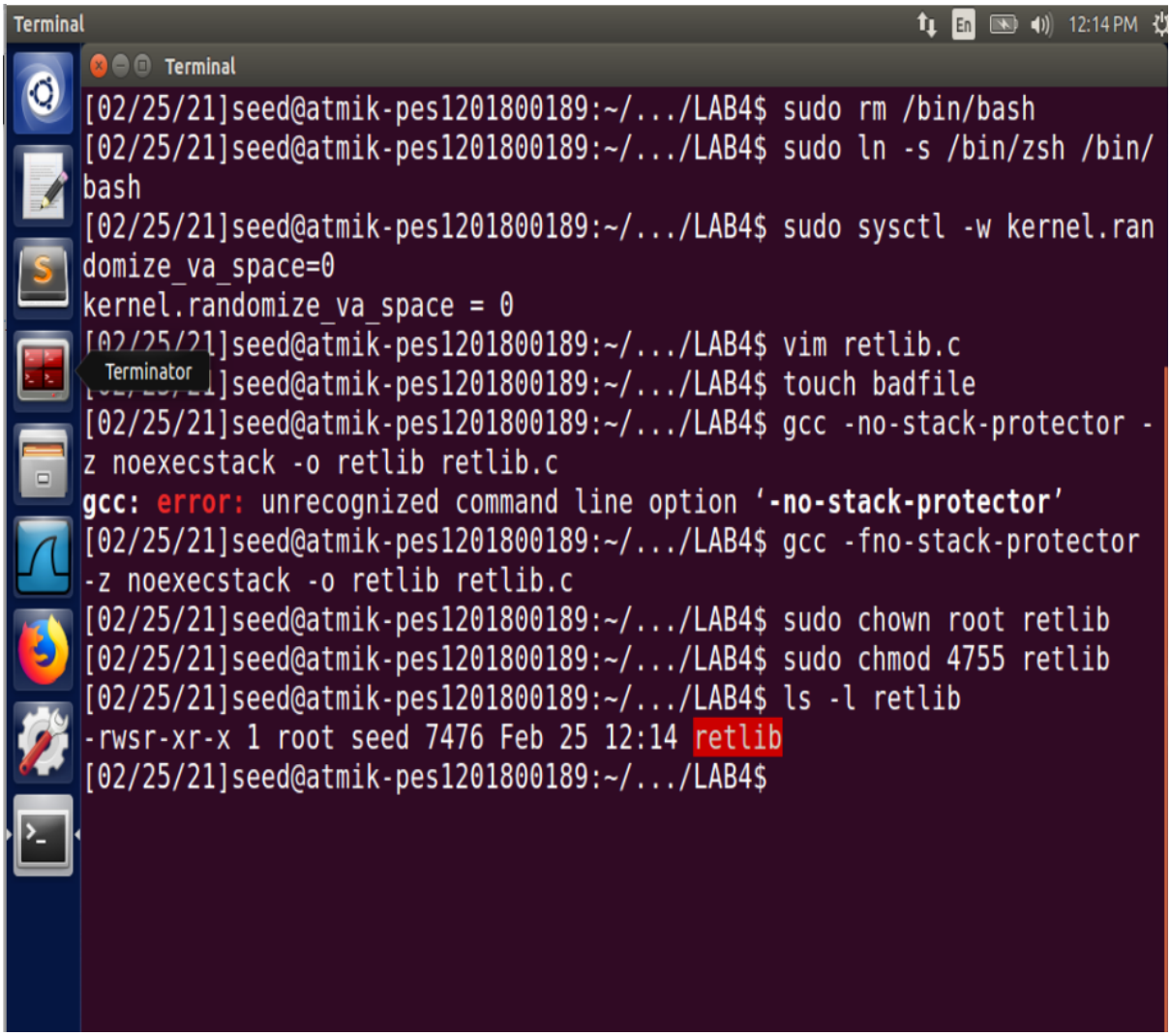
Section: A

Task 1:

- Setting kernel.randomize_va_space =0 will make sure that the memory segments are not randomized in order for our attack to be carried out successfully, as we know the address .
- fread(buffer,sizeof(char),40,badfile) will read from the badfile eventually in further experiments facilitating access to root shell (will be explained later)

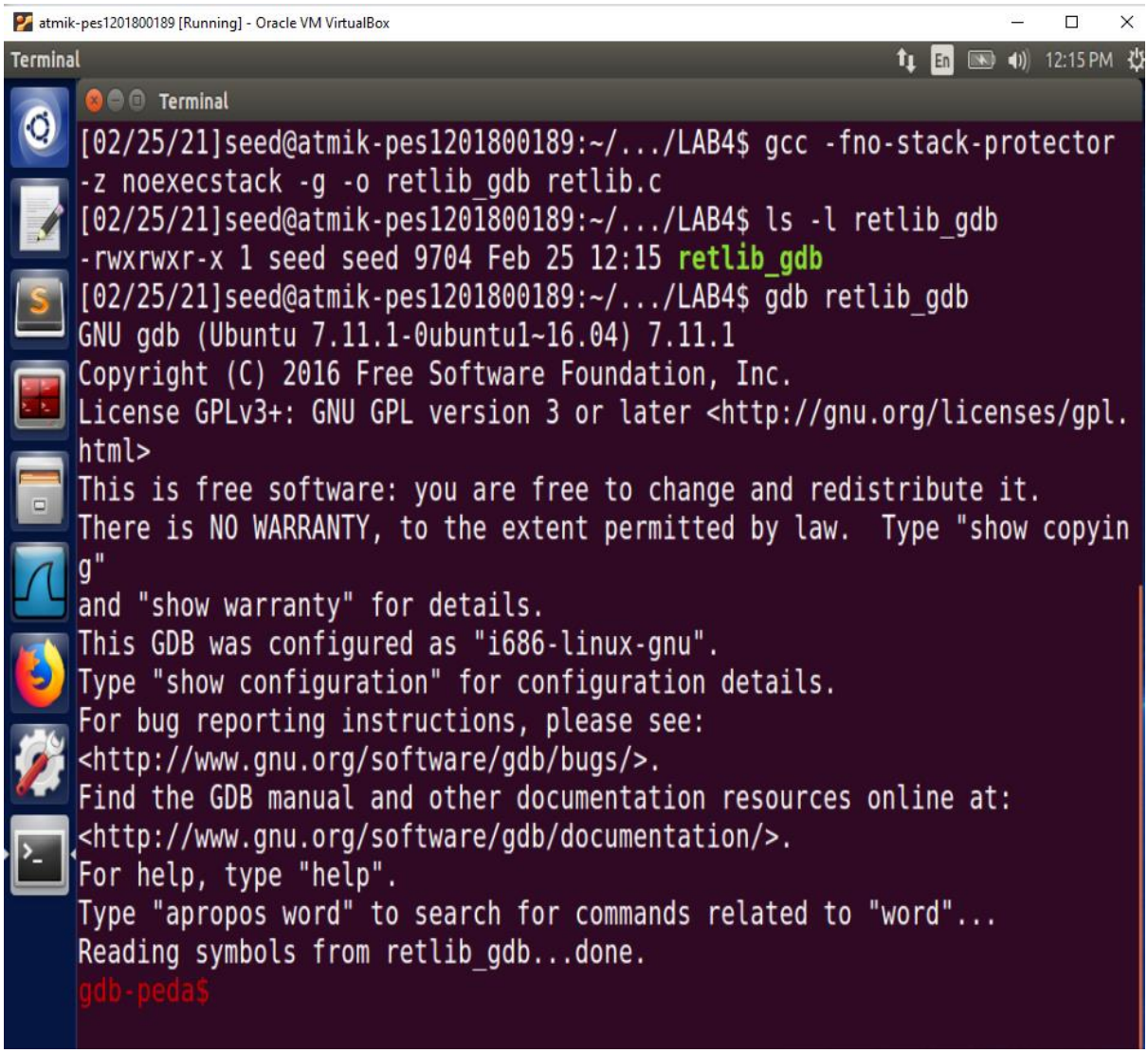


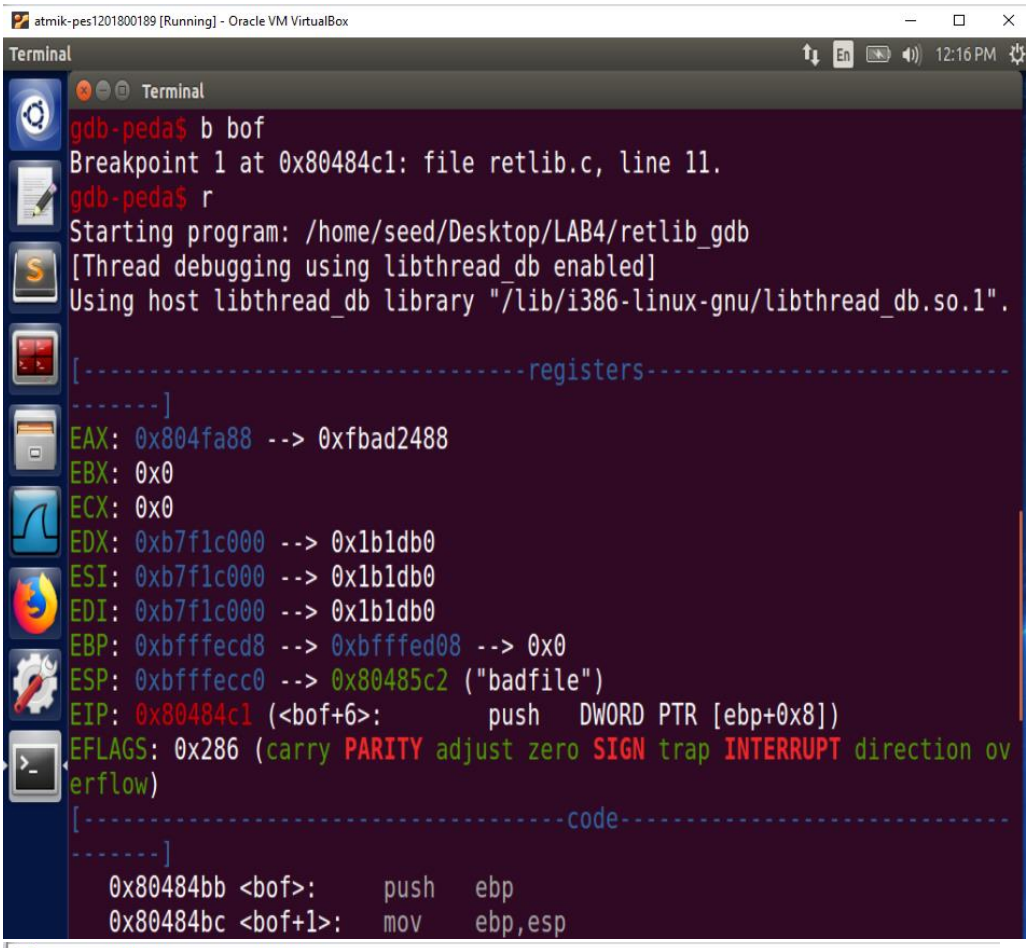
- The options for executing gcc command have been elaborated in previous labs

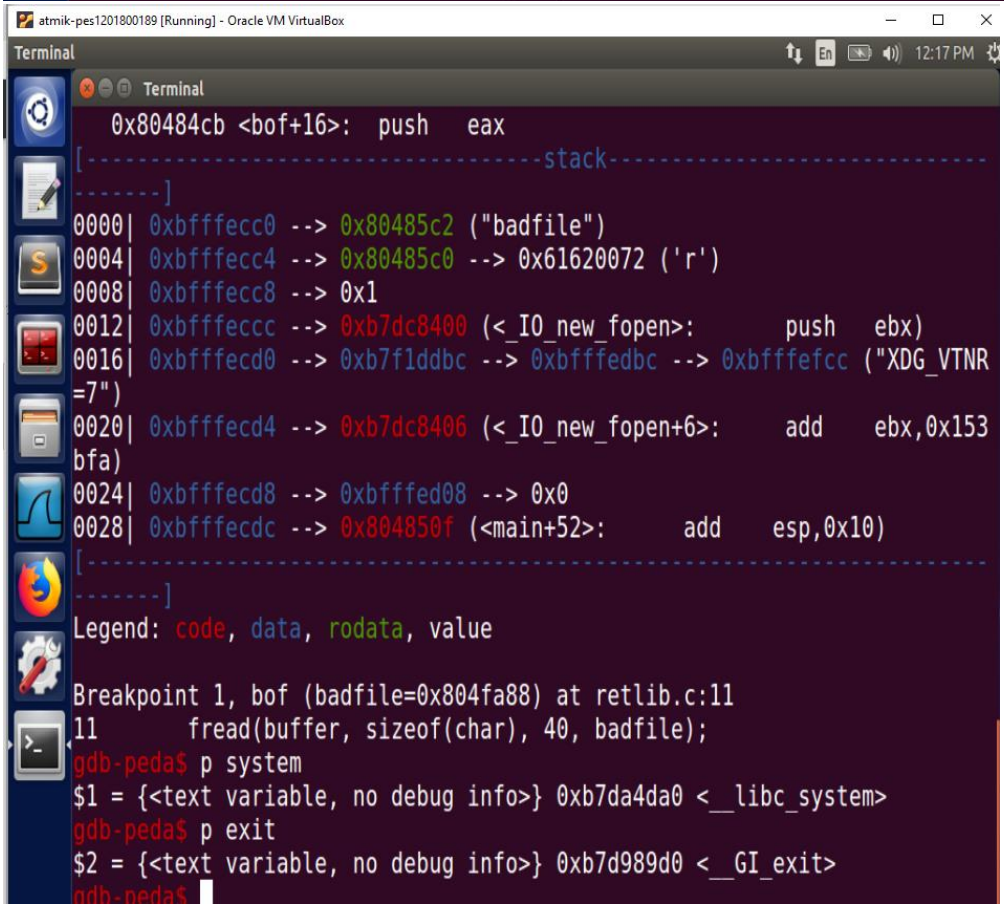


Task 2:

- We now want to find the address of the **system(), exit()** function and to do so we run **gdb**.
- We set a breakpoint at **bof** function and get the address data from registers by running r , to get system() address we run **p system** and same for exit()
- Spread over 3 screenshots to get the full output

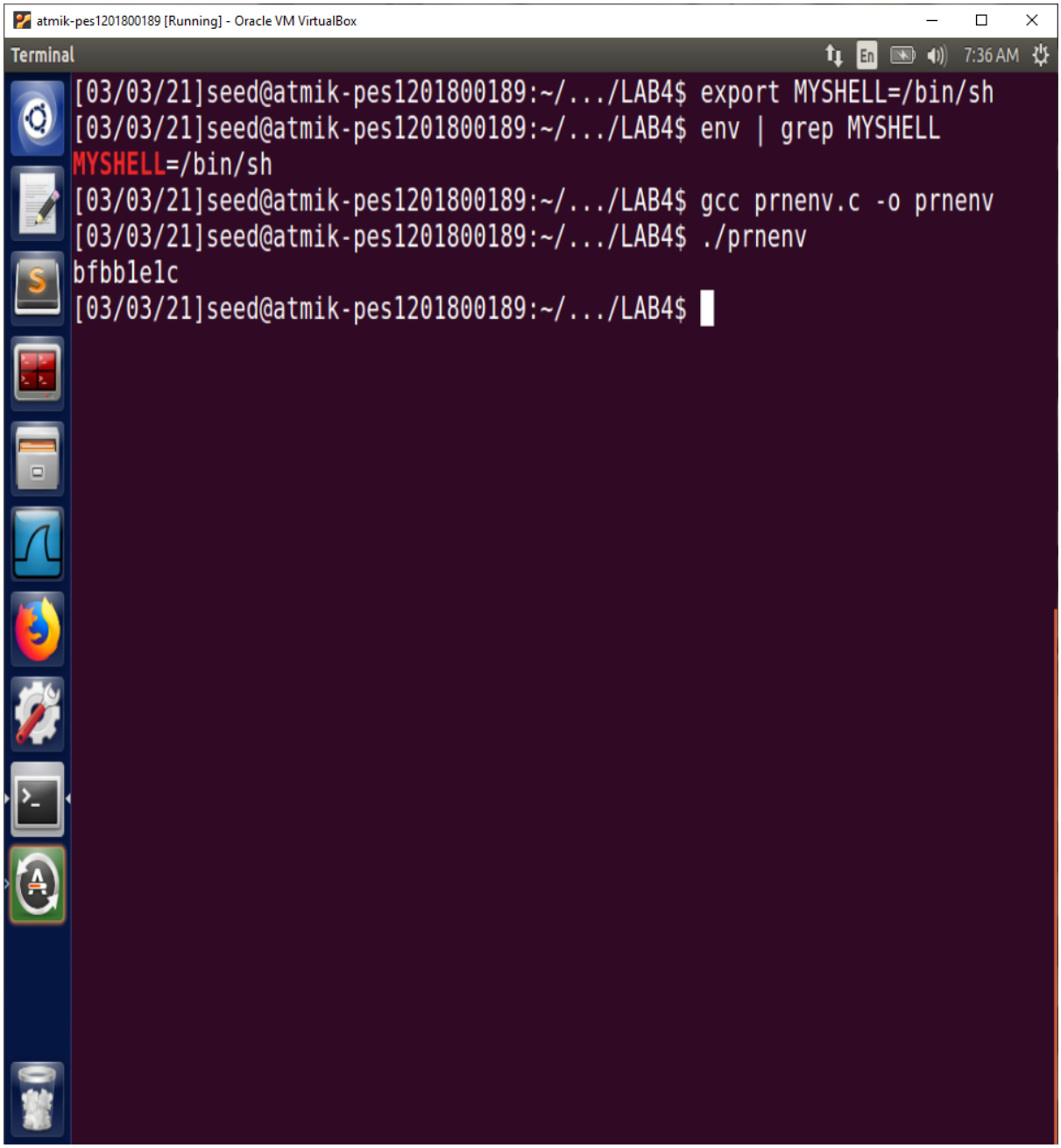



atmik-pes1201800189 [Running] - Oracle VM VirtualBox
Terminal
12:16 PM
Terminal
gdb-peda$ b bof
Breakpoint 1 at 0x80484c1: file retlib.c, line 11.
gdb-peda$ r
Starting program: /home/seed/Desktop/LAB4/retlib_gdb
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".
[----------registers----------]
EAX: 0x804fa88 --> 0xfbad2488
EBX: 0x0
ECX: 0x0
EDX: 0xb7f1c000 --> 0x1b1db0
ESI: 0xb7f1c000 --> 0x1b1db0
EDI: 0xb7f1c000 --> 0x1b1db0
EBP: 0xbfffecd8 --> 0xbfffed08 --> 0x0
ESP: 0xbfffecc0 --> 0x80485c2 ("badfile")
EIP: 0x80484c1 (<bof+6>: push DWORD PTR [ebp+0x8])
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[----------code----------]
0x80484bb <bof>: push ebp
0x80484bc <bof+1>: mov ebp,esp

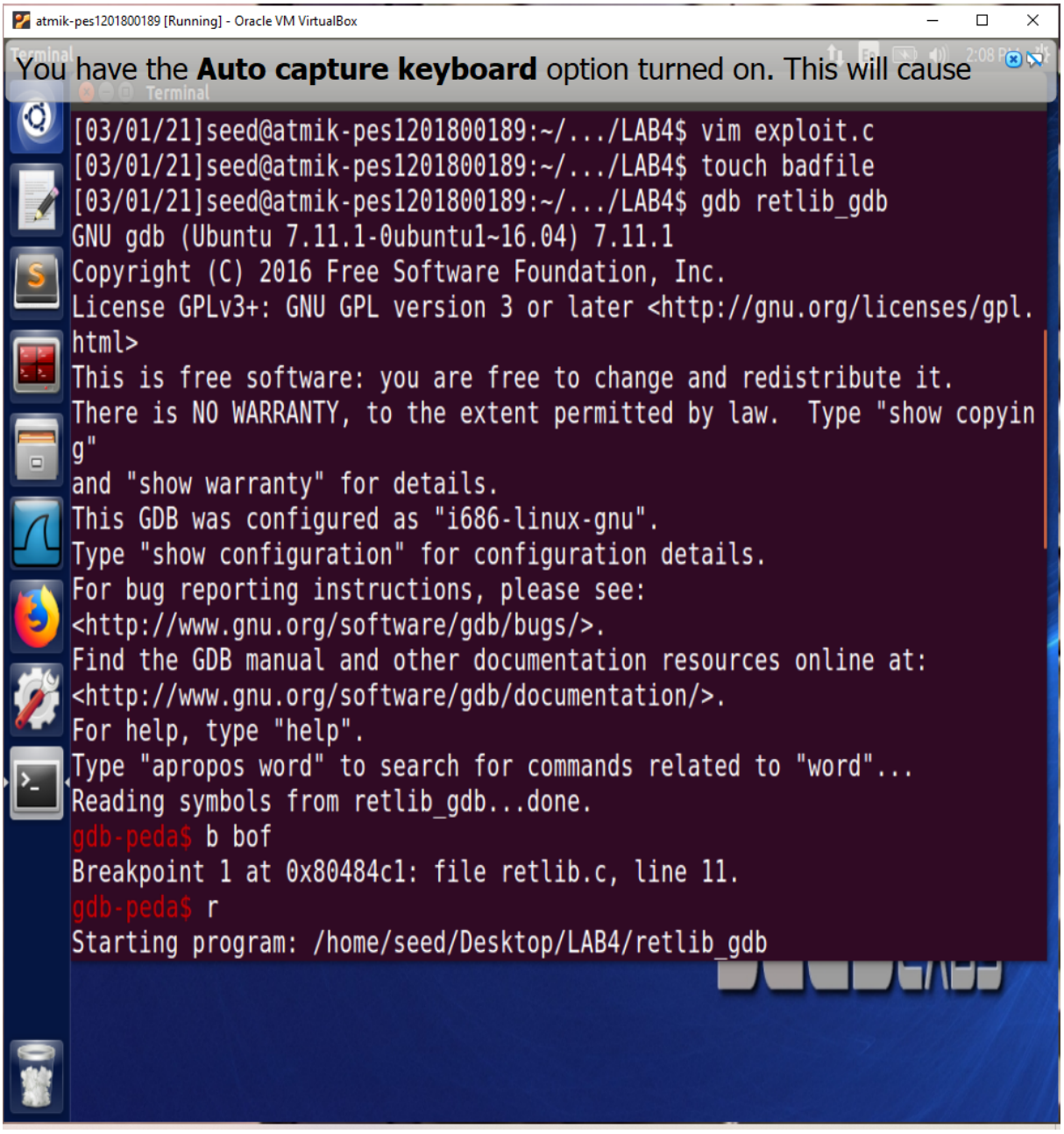

atmik-pes1201800189 [Running] - Oracle VM VirtualBox
Terminal
12:17 PM
Terminal
0x80484cb <bof+16>: push eax
[----------stack----------]
0000| 0xbfffecc0 --> 0x80485c2 ("badfile")
0004| 0xbfffecc4 --> 0x80485c0 --> 0x61620072 ('r')
0008| 0xbfffecc8 --> 0x1
0012| 0xbfffeccc --> 0xb7dc8400 (<_IO_new_fopen>: push ebx)
0016| 0xbfffecd0 --> 0xb7f1ddbc --> 0xbfffedbc --> 0xbfffefcc ("XDG_VTNR=7")
0020| 0xbfffecd4 --> 0xb7dc8406 (<_IO_new_fopen+6>: add ebx,0x153bfa)
0024| 0xbfffecd8 --> 0xbfffed08 --> 0x0
0028| 0xbfffecdc --> 0x804850f (<main+52>: add esp,0x10)
[----------]
Legend: code, data, rodata, value
Breakpoint 1, bof (badfile=0x804fa88) at retlib.c:11
11 fread(buffer, sizeof(char), 40, badfile);
gdb-peda$ p system
$1 = {<text variable, no debug info>} 0xb7da4da0 <__libc_system>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0xb7d989d0 <__GI_exit>
gdb-peda$

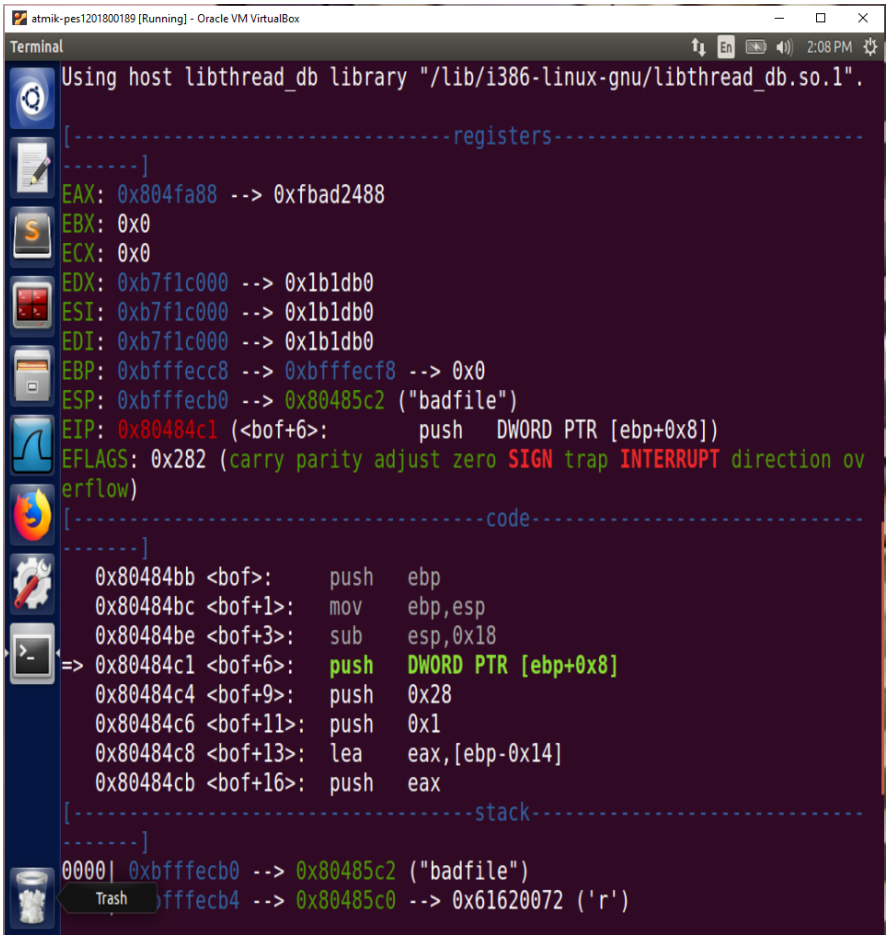
Task 3:

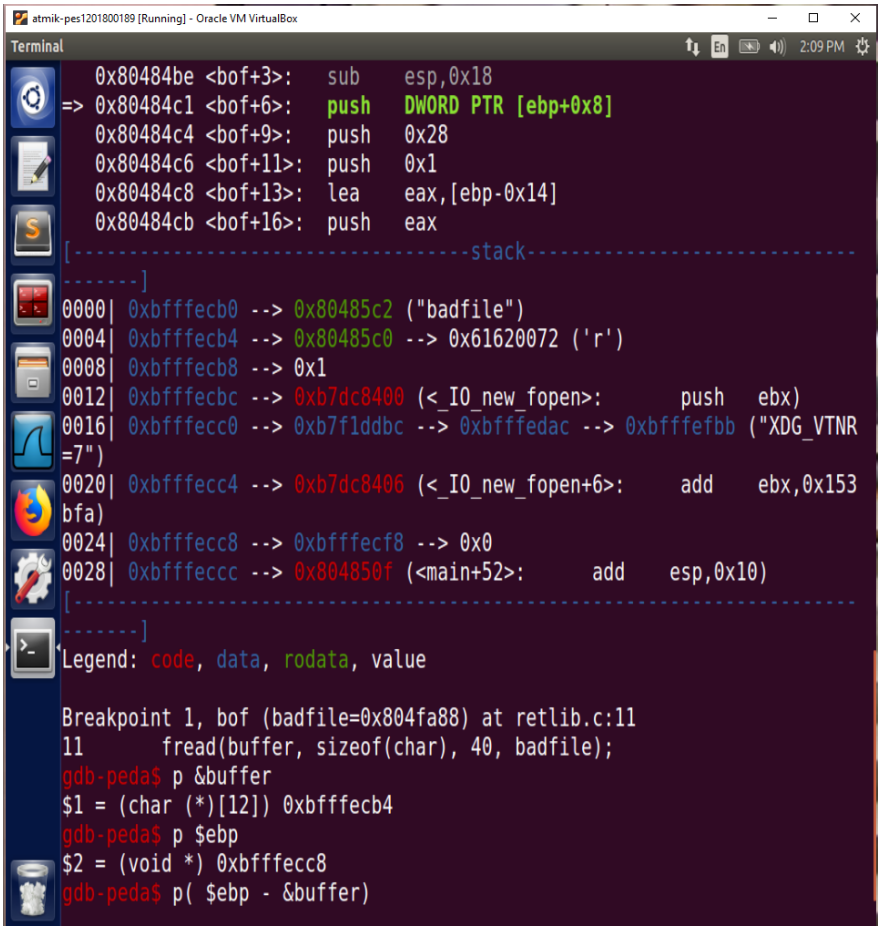
- We now want to put the string **/bin/sh** in memory, to do that the easiest way is to put it as an environment variable . We do this using **export MYSHELL=/bin/sh**
- **getenv("MYSHELL")** gets where MYSHELL env variable is stored and we print out the address after casting it to unsigned int to get the correct address.

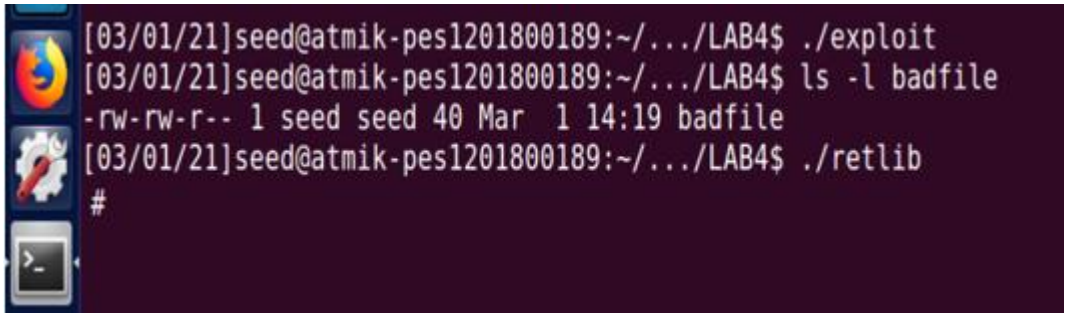


- With the address of /bin/bash that we have from above, and from task 2, we have gotten the address of system() and exit(). We can replace those addresses in the respective buffer values on the C program.
- Value of the sizes can be calculated using the formula in the manual on running **gdb and p &buffer, p $ebp**



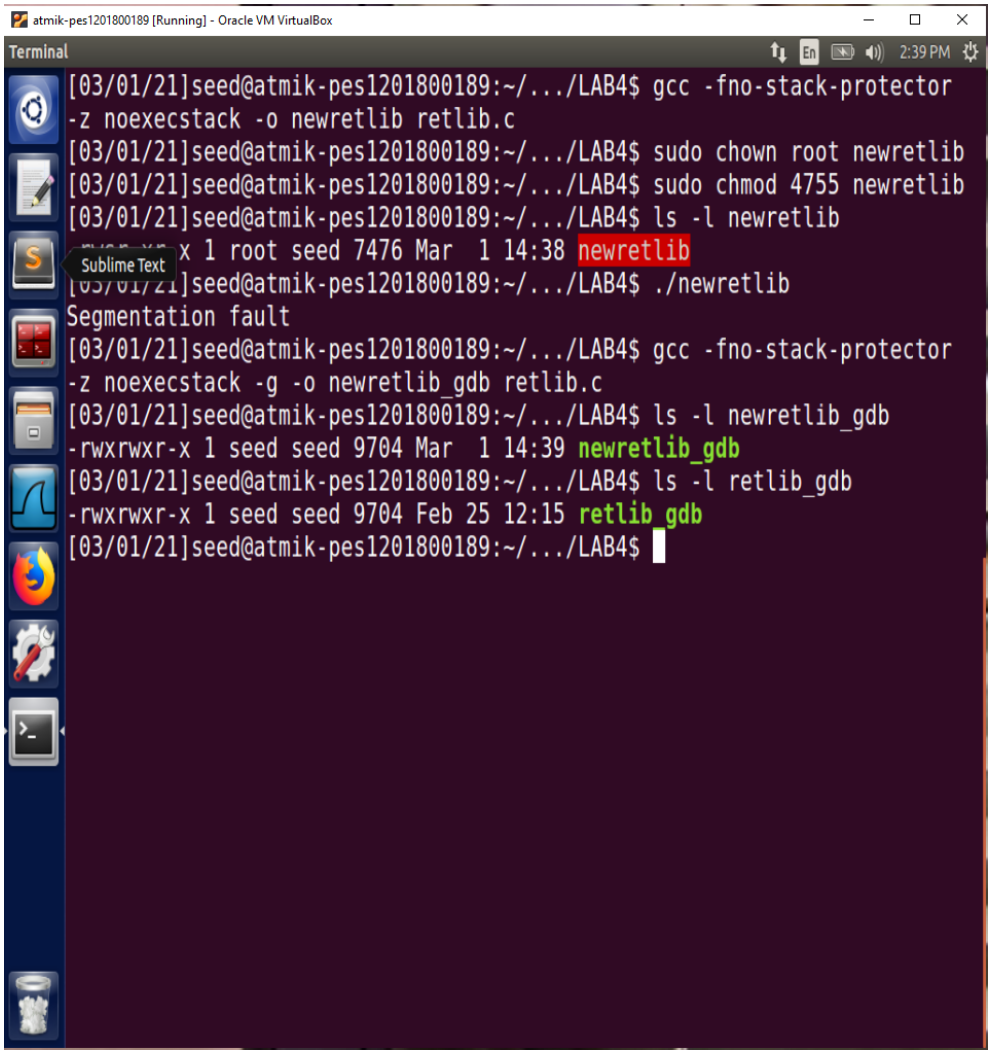

atmik-pes1201800189 [Running] - Oracle VM VirtualBox
Terminal
2:08 PM
Using host libthread_db library "/lib/i386-linux-gnu/libthread_db.so.1".
[----------registers----------]
EAX: 0x804fa88 --> 0xfbad2488
EBX: 0x0
ECX: 0x0
EDX: 0xb7f1c000 --> 0x1b1db0
ESI: 0xb7f1c000 --> 0x1b1db0
EDI: 0xb7f1c000 --> 0x1b1db0
EBP: 0xbfffecc8 --> 0xbfffecf8 --> 0x0
ESP: 0xbfffecb0 --> 0x80485c2 ("badfile")
EIP: 0x80484c1 (<bof+6>: push DWORD PTR [ebp+0x8])
EFLAGS: 0x282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[----------code----------]
0x80484bb <bof>: push ebp
0x80484bc <bof+1>: mov ebp,esp
0x80484be <bof+3>: sub esp,0x18
=> 0x80484c1 <bof+6>: push DWORD PTR [ebp+0x8]
0x80484c4 <bof+9>: push 0x28
0x80484c6 <bof+11>: push 0x1
0x80484c8 <bof+13>: lea eax,[ebp-0x14]
0x80484cb <bof+16>: push eax
[----------stack----------]
0000| 0xbfffecb0 --> 0x80485c2 ("badfile")
Trash
fffecb4 --> 0x80485c0 --> 0x61620072 ('r')


atmik-pes1201800189 [Running] - Oracle VM VirtualBox
Terminal
2:09 PM
0x80484be <bof+3>: sub esp,0x18
=> 0x80484c1 <bof+6>: push DWORD PTR [ebp+0x8]
0x80484c4 <bof+9>: push 0x28
0x80484c6 <bof+11>: push 0x1
0x80484c8 <bof+13>: lea eax,[ebp-0x14]
0x80484cb <bof+16>: push eax
[----------stack----------]
0000| 0xbfffecb0 --> 0x80485c2 ("badfile")
0004| 0xbfffecb4 --> 0x80485c0 --> 0x61620072 ('r')
0008| 0xbfffecb8 --> 0x1
0012| 0xbfffecbc --> 0xb7dc8400 (<_IO_new_fopen>: push ebx)
0016| 0xbfffecc0 --> 0xb7f1ddbc --> 0xbfffedac --> 0xbfffefbb ("XDG_VTNR=7")
0020| 0xbfffecc4 --> 0xb7dc8406 (<_IO_new_fopen+6>: add ebx,0x153bfa)
0024| 0xbfffecc8 --> 0xbfffecf8 --> 0x0
0028| 0xbfffeccc --> 0x804850f (<main+52>: add esp,0x10)
[----------]
Legend: code, data, rodata, value
Breakpoint 1, bof (badfile=0x804fa88) at retlib.c:11
11 fread(buffer, sizeof(char), 40, badfile);
gdb-peda$ p &buffer
$1 = (char (*)[12]) 0xbfffecb4
gdb-peda$ p $ebp
$2 = (void *) 0xbfffecc8
gdb-peda$ p($ebp - &buffer)

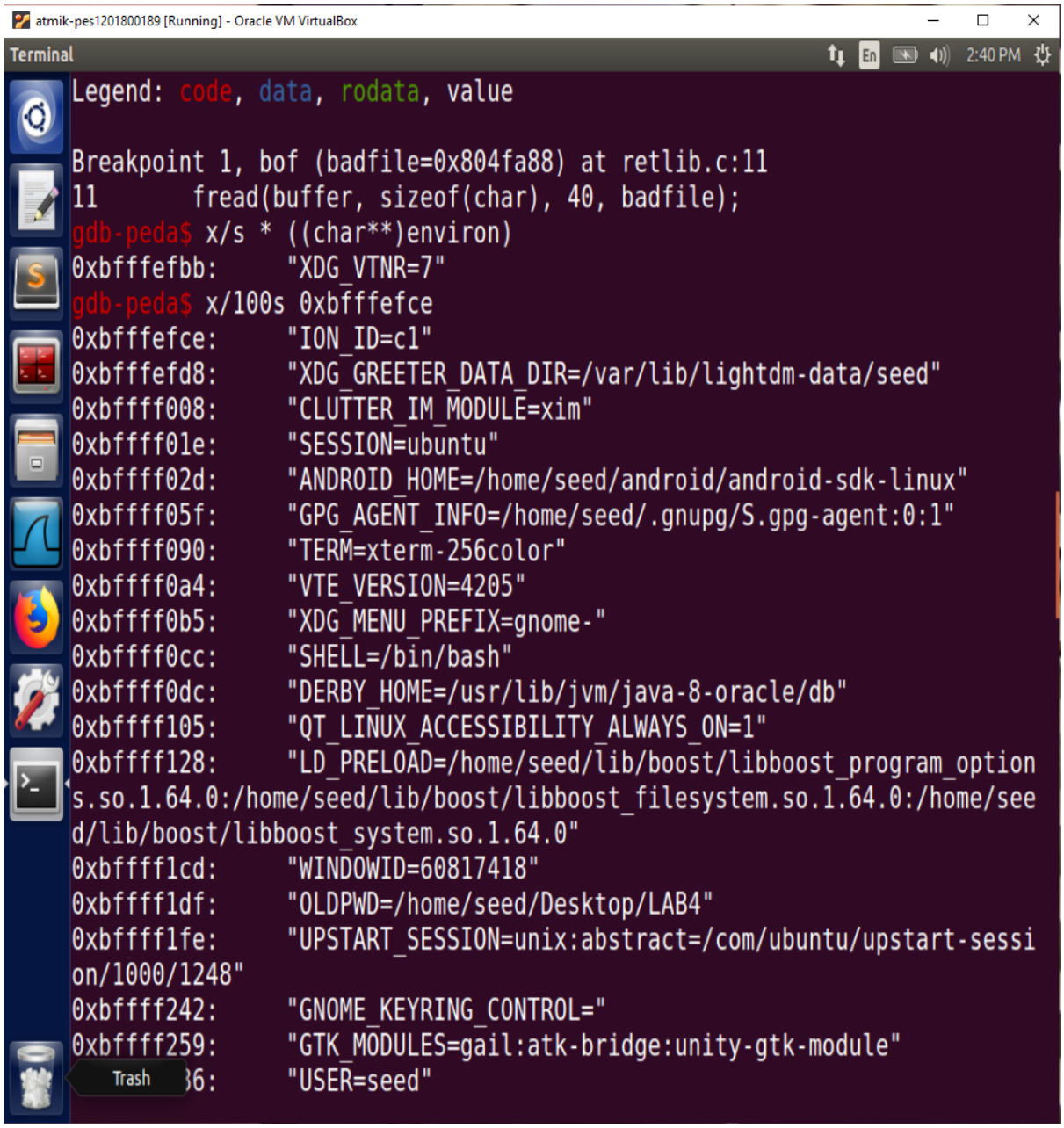


Task 4:

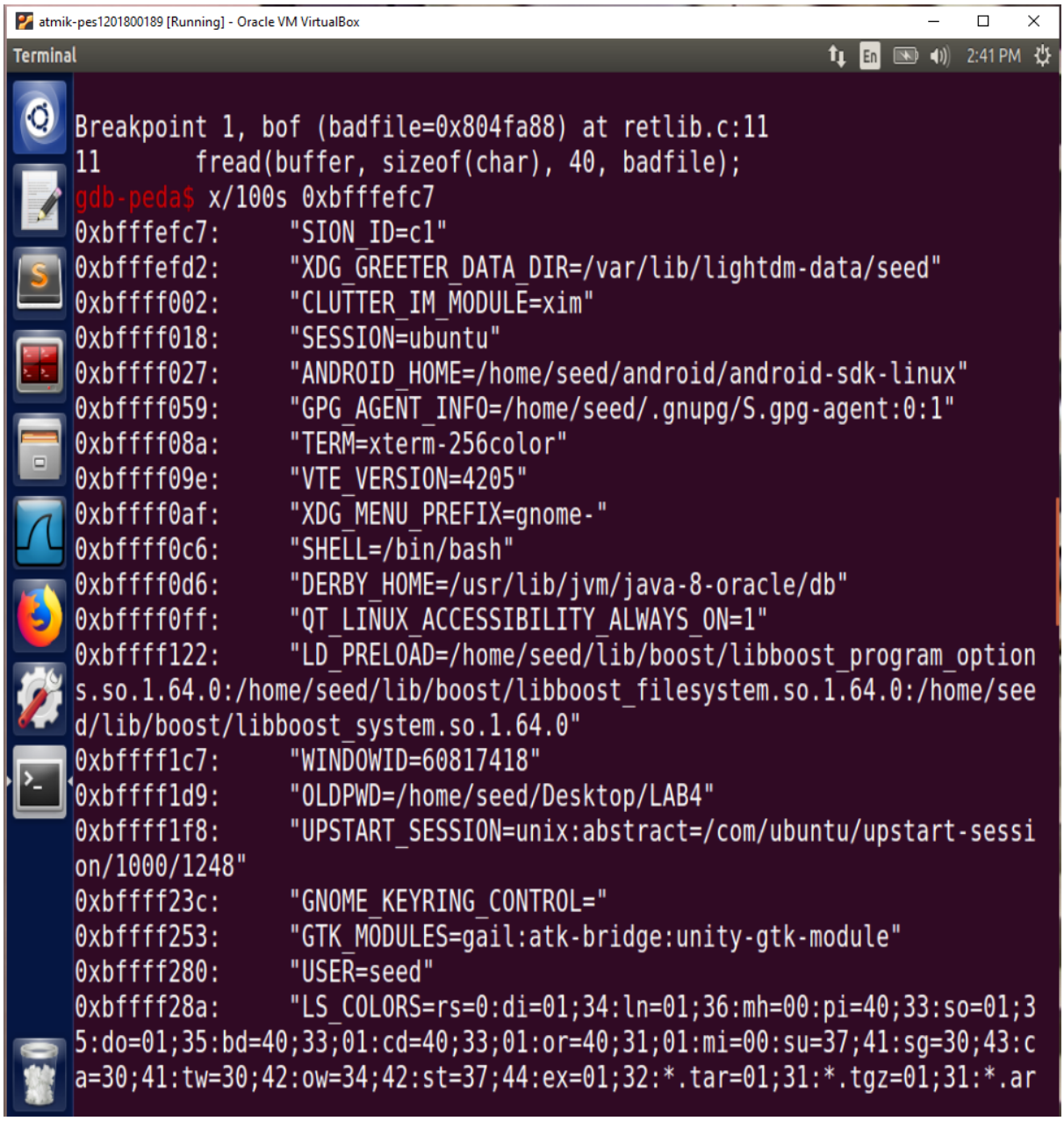
- Attack fails and gives segmentation fault, segmentation fault comes when it tried to access memory not available or allowed to it. Hence segmentation fault is the correct output we are meant to get



- Environ is a pointer to pointer and has the type char** environ , hence when we want to access the address we need to execute **x/s * ((char**)environ)**

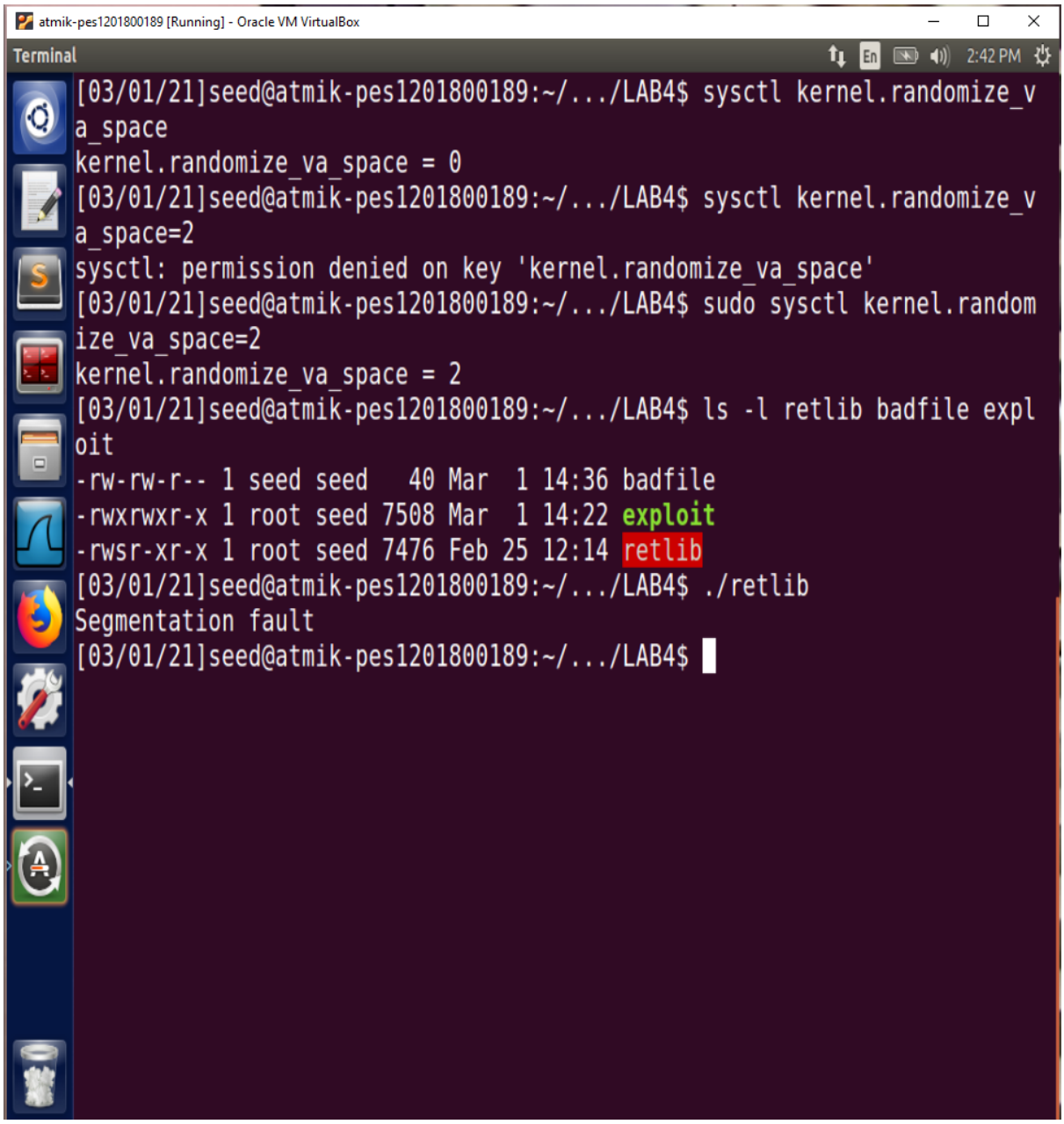


- after debugging the first program(retlib_gdb) we debug the new one(newretlib_gdb)

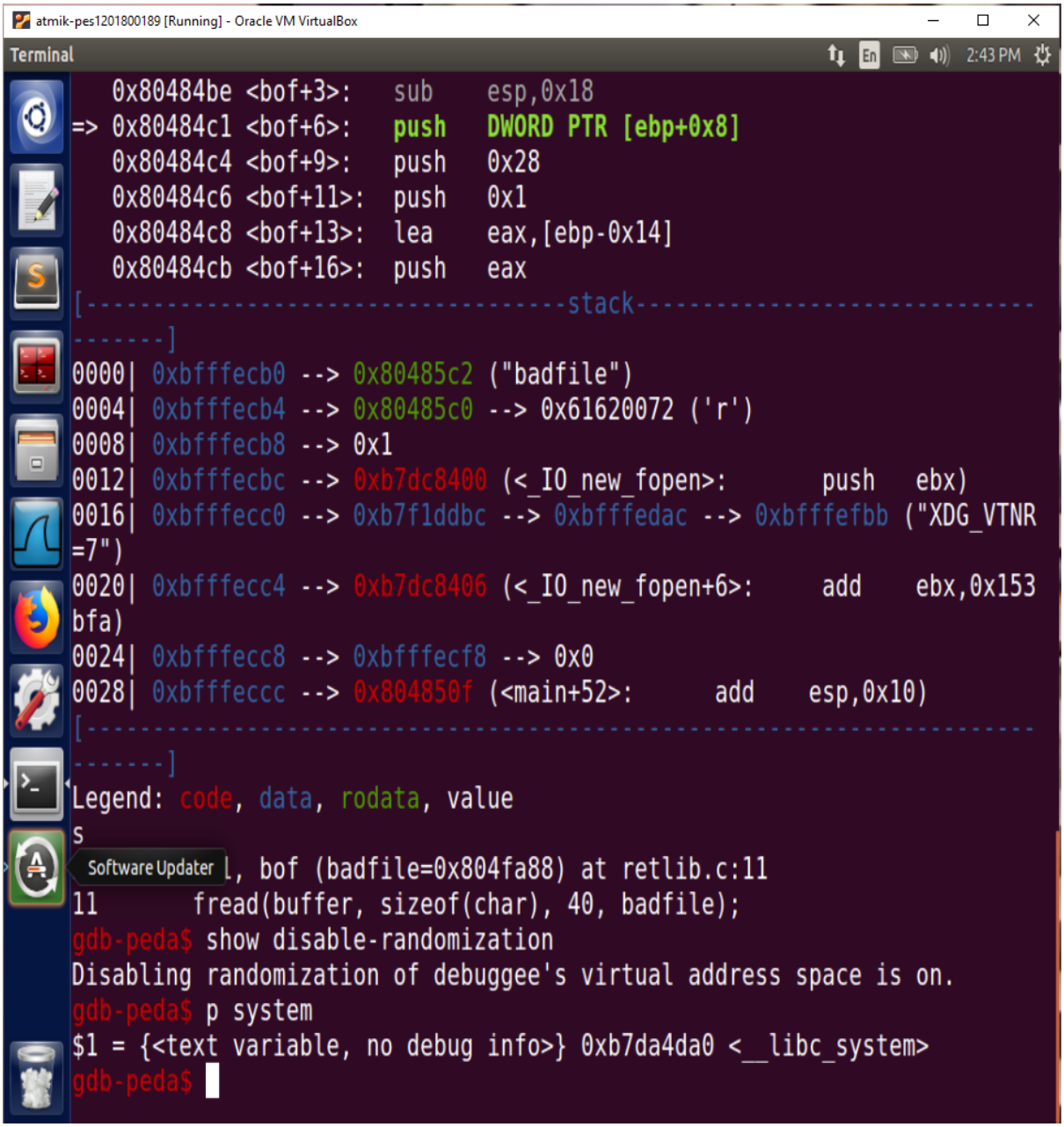


Task 5:

- Setting the value to 2 will enable the randomization od memory segments , it is the default setting of the system and most secure



- We disable address randomisation with show disable-randomization and get the address of system() call on putting breakpoint at bof function



- We disable the address randomisation using show disable-randomization and get the address of system on putting breakpoint at main function