**Reminders for gdb in general**

x86 is little endian, so entering command x/1wx 0x202dfd40 displays 0x74206968 (“hi t“)

0x202dfd40 = 0x68

0x202dfd41 = 0x69

0x202dfd42 = 0x20

0x202dfd43 = 0x74

can easily show this by displaying by byte (x/b)

**sploit1.c**

/\* Notes:

\* The overflow string is divided into three sections:

\* NOPs, the shellcode, and guessed address

\* AlephOne's shell code is 46 bytes (with NULL byte at end!)

\* x86 is little-endian

\* lab has the stack at a fixed address!!!

\* On ECF machine, the return address of main is 0x202dfe88

\* buf is located from 0x202dfe10 to 0x202dfe70

\* distance between buf and return address to overwrite is 0x18 (24) bytes

\* add to the end of the attack string the return address,

\* (only need lower 4 bytes) so repeat it 7 times (6 to get to start of

\* return address and 1 more to overwrite it)

\* Pad the beginning of the string with NOPs; need 96 - 46 + 1 = 51 bytes

\* Realize that calculation is such that if you need to pad for one address

\* you need one byte

\*/

**sploit2.c**

/\* Notes:

\* Unlike target1, the return address to overwrite is foo(), not lab\_main()

\* On ECF machine, the ret addr of foo() in target2.c is 0x202dfe58

\* The local variable buf[] in foo() starts at 0x202dfd40

\* As it 256 chars long, it ends at 0x202dfe40 (0x100)

\* Distance between \*end\* of buf and \*start\* of ret addr is 0x18

\* Pad beginning of string with enough NOPs: 256 - 46 + 1 = 211

\*/

**sploit3.c**

**sploit4.c**