SI485H Stack Based Binary Exploits and Defenses
FALL 2015

NAME:_____

HW3

COLLABORATOR(S):_____

15/13/10/5/0

1. Match the gdb shortcut command to the full command name
(*note: some of these you might have try yourself in gdb)*

b ___	(a) info args
r ___	(b) info frame
i f ___	(c) info registers
n ___	(d) list
s ___	(e) run
ni ___	(f) step
l ___	(g) step instruction
ds ___	(h) next
bt ___	(i) next instruction
si ___	(j) examine
fin __	(k) print
i r __	(l) finish
p ___	(m) disassemble
x ___	(n) break
i ar ___	(o) backtrace

10/8/5/0

2. Given the following **info frame** and **backtrace** output from gdb, apply the appropriate label from the ones provided:

```
(gdb) info frame
Stack level 0, frame at 0xbffff680:
 eip = 0x8048423 in foo (frame.c:5); saved eip = 0x804846b
 called by frame at 0xbffff6a0
 source language c.
 Arglist at 0xbffff678, args: a=1, b=2, c=3
 Locals at 0xbffff678, Previous frame's sp is 0xbffff680
 Saved registers:
  ebp at 0xbffff678, eip at 0xbffff67c
(gdb) backtrace
#0  foo (a=1, b=2, c=3) at frame.c:5
#1  0x0804846b in main () at frame.c:10
```

(a) the address of the function foo

(b) the return address of the function foo

(c) the address arguments to the function foo

(d) the address of the current instruction in the function foo

__/25

NAME: _____

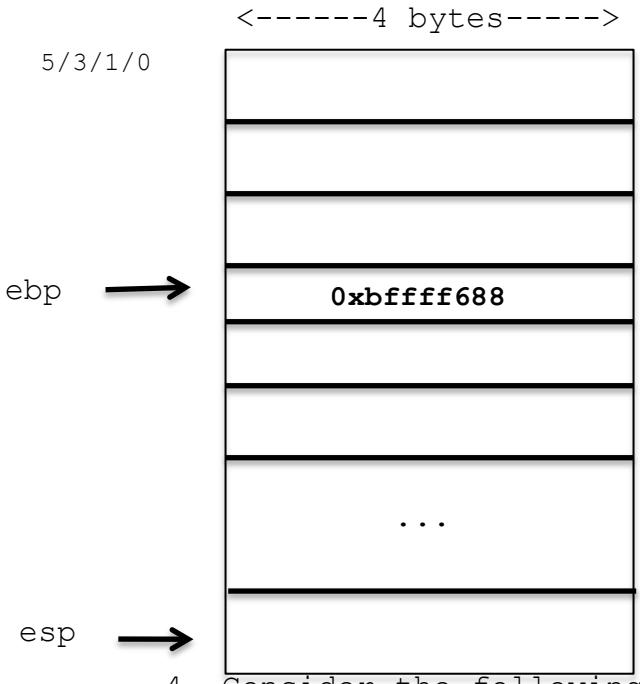
3. Clone the following repository

```
git clone git@saddleback.academy.usna.edu:aviv/HW-3.git
```

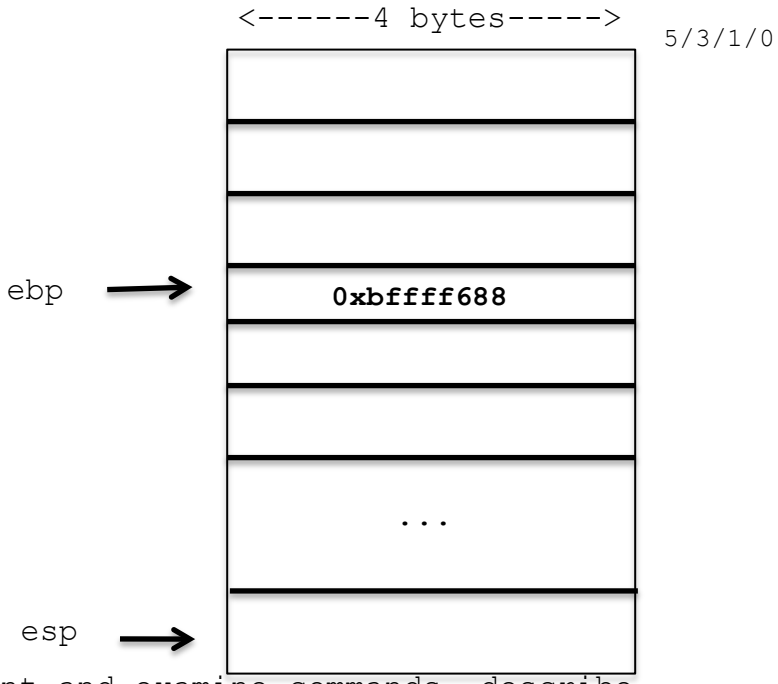
And run the program **main** under the debugger. Place a break point at the end of the function at the start of **foo** and then run the program with the command line arguments:

```
(gdb) run 5 1
```

a) Fill in the memory diagram for the function rame after instruction at address x0804845a executes:



b) Step through the loop two times: After the instruciton 0x08048472 executes the second time, fill in the memory



4. Consider the following print and examine commands, describe what their output would be in a gdb terminal:

3/2/0 a) x/20cx 0x0808486

3/2/0 b) p/x $ebp

3/2/0 c) x/xh $esp

3/2/0 d) x/i $eip

3/2/0 e) x/5gd $ebp-0x4

__/25

NAME: _____

10/8/5/0

5. In the HW-3 project you cloned before, you will find a program name **stacked**. Execute the program in gdb and put a break point at the function baz.
How many functions are called before baz is called?

and how did you determine this?

20/18/15/10/0

6. In the HW-3 project you cloned before, you will find a program named **trace-me**. Determine what the right input is for this program (using gdb) to have it print the secret message. Describe the secret message below

and how you obtained it:

(Hint 1: Does it take arguemnts?)
*(Hint 2: Trace main using **ni** [next instruction])*
(Hint 3: What is being compared?)

20/18/15/10/0

7. In the HW-3 project you cloned before, you will find a program named **crack-me**. Use gdb to inspec the various: elements of the main() program to determine the secret message
Describe the secrement message below

and how you obtained it:

*(Hint 1: use examine [i.e., **x**] check things out)*
(Hint 2: use print to perform operations and see results)
(Hint 3: use print again to reinterpret those results)

__/50