**CSE 484 In-class Worksheet #3**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ UW Student #: \_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Email address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Partner names for this activity: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

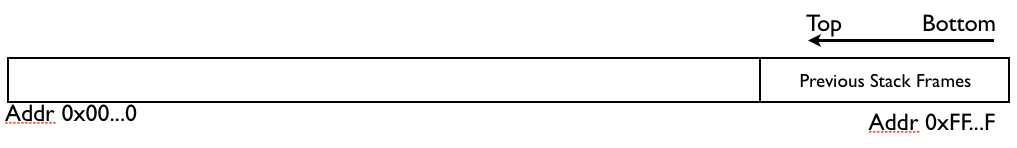
**Q1:** In the figure below, draw what happens on the stack (x86) when this function is called. What might get overwritten if str is longer than 126 bytes?

void func(char \*str) {

char buf[126];

strcpy(buf,str);

}



**Q2:** Apache 1.3 had the following code:

strcpy(record,user);

strcat(record,”:”);

strcat(record,cpw);

The published fix:

strncpy(record,user,MAX\_STRING\_LEN-1);

strcat(record,”:”);

strncat(record,cpw,MAX\_STRING\_LEN-1);

Is this fix good? If so, why? If not, why not?

Q3: Consider this code:

void mycopy(char \*input) {

char buffer[512]; int i;

for (i=0; i<=512; i++)

buffer[i] = input[i];

}

void main(int argc, char \*argv[]) {

if (argc==2)

mycopy(argv[1]);

}

Is this code exploitable? If not, why not? If so, why? You may use the diagram below to help answer this question, if you wish.

