Dylan Paulus
CSCD 437 – Secure Coding
Assignment 4

a)

- strcpy is being used without any checks if the incoming string is the same length as the outgoing buffer. This can cause buffer overflow.
    - This can be avoided by checking the arg string length, and see if it matches the out string length in the foo() function.

b)

- 'len' in the foo() function bounds the length to 136 characters where the char buff[128] is only set up for 128 characters. This can cause buffer overflow.
    - Changing the 'if' statement to only allow 127 characters would fix this problem.
- '\0' null termination is missing at the end of buf in foo().
    - After the initialization of char buff[128], memsetting all the bits can first, get rid of any garbage, and second, make sure the ending of the string ends with null termination.

```
char buf[128];
memset(buff, '\0', 128);
```

c)

- If the length of arv[1] is greater than the passed in ltarg value, then the length of arg is set to ltarg. In this case ltarg is set to 140, where buff is only set to be length 128. This can be a problem if too big of an argv is passed in.
    - Change the function bar() to only accept a length equal to the length of buf.
    - Or, just remove ltarg in general –since we are already looking at the strlength of arg, just use the strlen of targ also.
- Again, no null termination at the end of the new string.
    - Memset the new buf[] char array to be all nulls, then add the value of the string.

d)

- A short is being used when an int might be better. If the input length surpasses the length of the short, it will underflow.
    - Convert short to int
- Null terminator needs to be applied to the buf[] char array to terminate the string
    - Memset(buf, '\0', arglen);
- Non-necessary strlen being used in the for loop. Arglen already exists with the length of the arg input

- Just use arglen in the for loop instead of strlen(arg)