CSCI 180 Computer Security - Fall 2019
Homework 1 - Due 10/21/2019 in the beginning of class

1. (10 pts) Is the following function vulnerable to the buffer overflow? If yes, explain how by drawing the stack frame and explaining what could happen. If no, explain why not.

```
void readinput()
{
    char str[30];
    gets(str);
    printf("%s", str);
    return;
}
```

2. (10 pts) Identify the security vulnerability in this code. Explain.

```
void copyinfo(w) {
    size_t len;
    char *buf;

    len = read_int_from_network();
    buf = malloc(len+10);
    read(info, buf, len);
    …
}
```

3. (10 pts) For the following code, assume an attacker can control the value of shoplist passed into eval_list. The value of n is constrained to correctly reflect the number of elements in shoplist. The code includes several security vulnerabilities. **Show at least two such vulnerabilities** in the code and **explain each one**.

```
1  struct food {
2      char name[1024];
3      int price ;
4  };
5
6  /* Evaluate a shopping list with at most 16 food items.
7  Returns the number of inexpensive items , or -1 on a problem . */
8  int eval_list(struct food shoplist[], size_t n) {
9      struct food good[16];
10     char expensive[1024] , cmd[1024];
11     int i, total = 0, ngood = 0, size_exp = 0;
12
13     if (n > 16) return -1;
14
15     for (i = 0; i <= n; ++i) {
16         if (shoplist[i].price < 40)
17             good[ngood++] = shoplist[i];
18         else if (shoplist[i].price > 70) {
19             size_t len = strlen(shoplist[i].name);
20             snprintf(expensive + size_exp, len, "%s ", shoplist[i].name);
21             size_exp += len ;
22         }
23
24         total += shoplist[i].price;
25     }
26
27     if (total > 1200) {
28         const char *str_fmt = "total value %d --expensive-items %s";
29         fprintf(stderr , "too much!");
30         snprintf (cmd, sizeof cmd, str_fmt ,total ,expensive) ;
31         system(cmd);
32     }
33
34     return ngood;
35 }
```

Reminders:

- snprintf(buf, len, fmt, . . .) works like printf, but instead writes to buf, and won't write more than len - 1 characters. A terminating null character is automatically appended after the content written.
- system runs the shell command given by its first argument.