Derrick Lee
Homework 1
CSCI 180
October 19, 2019

1

This function is vulnerable to buffer overflow. Since there is a char buffer of 30 bytes and gets(str) is used without checking for overflow. This results in possibly attempting to store more data than the buffer. The stack frame is as follows:

str[30]
SFP
ret

If a user has an input that is over 30 bytes, it will overflow the buffer and consequently overwrite the previous frame pointer along with the return address. This will result in various possible ways if the return address is modified:

1. The return address points to a valid instruction elsewhere in the memory, potentially in the given buffer, thus allowing external code execution.
2. The return address points to a protected or invalid location causing a segmentation fault.

2

The main issue is the lack of input validation associated with the memory allocation resulting in an attack with integer overflow. With the given len from the network, any number can be provided. The numbers that would cause issues are between UINT_MAX − 9 and UINT_MAX (UINT_MAX being 4294967295 on a 32 bit system or 18446744073709551615 on a 64 bit system).

Assuming UINT_MAX is provided and assigned to len, the malloc(len + 10) will result in an allocation of UINT_MAX + 10 which is equivalent to 9. With 9 bytes allocated to buf and info is read into buf with a length of UINT_MAX, the data copied is larger than the buffer size causing a buffer overflow.

3

1. **Off-by-One error**

   The for loop iterates from 0 to <= n which would result in an index value that is one past the end of the input shoplist array.

2. **Buffer Overflow**

When using snprintf(), the input food name is copied to the next position in the expensive buffer. Because each food name is 1024 bytes and the expensive buffer is also 1024 bytes, having at least two elements in the shoplist can produce a buffer overflow as the sum of the shoplist name lengths (for items priced over 70) can be greater than the expensive buffer size.

Another area that could produce a buffer overflow is adding len to size_exp. Since len is a size_t and size_exp is an int, adding enough to size_exp can cause an integer overflow (possibly negative) and thus allow snprintf write to a memory location that isn't intended.

3. **OS Command Execution**

If the total is greater than 1200, a command will be executed on the system that contains a list of the expensive items from the input food names. Since the list of expensive food names are listed without input sanitation, any command can be executed on the system by the user. If the program is run as root, then this allows a malicious user to execute any command as root.

For example, if the total is over 1200 and an expensive food item is given the name "; cat /etc/passwd" or "&& sudo rm −rf /" (a "#" can be added at the end if there are to be items after the food name to be commented out and thus ignored. e.g. "&& cat /etc /passwd #").