SANTA CLARA UNIVERSITY
Computer Engineering Department

COEN 225: Lab 5

Source Code Analysis

(15 points)

Find the software defects resulting from string manipulation, heap management, integer, and formatted output operations in each of the following programs. *Rewrite each program correctly*. For example, you may have to replace unsafe functions with safe functions, correct the syntax of unsafe functions, add range checks, validate inputs, allocate and free memory correctly, use pre-conditions or post-conditions to detect integer overflows, add appropriate error messages, and abort program execution if necessary. Test your solution by running the program with various inputs that try to "break" the program. The program should not terminate with a segmentation fault.

1. In this program, the method `readFile` is used to read the contents of a binary file. The first element in the file is the number of bytes of data stored in the remainder of the file, and this value is read into the variable `len` using the first call to the function `fread`. A buffer called `buf` is created next, and the remaining data in the file is read into this buffer (using the second call to `fread`). (Hint: Validate `len`, check whether data was read correctly from the file, etc. Note that SIZE_MAX is the maximum size of a size_t.)

```
#include <stdlib.h>
#include <stdio.h>
char* readFile(FILE* fp);

int main(void) {
   FILE *fp;
   char *buf;
   fp = fopen("file1.txt", "r+");
   buf = readFile(fp);
   printf("%s\n", buf);
   fclose(fp);
}

char* readFile(FILE* fp) {
   char *buf;
   size_t len;
   int bytes_read;

   /* read data from a binary file */
   bytes_read = fread(&len, sizeof(len), 1, fp);
   printf("len=%u, bytes_read=%d\n", len, bytes_read);
   buf = malloc(len+1);
   bytes_read = fread(buf, sizeof(char), len, fp);
   return buf;
}
```

2. In this program, a buffer of size `argv[1]` is created using `calloc`. Using a `for` loop, some data is stored in the buffer and displayed. (Hint: check that `calloc` was successful, etc.)

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
   size_t len;
   int  *buf, i;

   len = atoi(argv[1]);
   printf("len=%d\n", len);
   buf  = (int *) calloc(len,  sizeof(int));

   for (i = 0; i < len; i++) {
        buf[i] = i * 10;
        printf("%d\t", buf[i]);
   }

    return 0;
}
```

3. In this program, a file is opened using "`w+`" mode. The command line argument `argv[1]` gives the number of characters (including null terminator) in the string stored in `argv[2]`. The argument `argv[2]` is copied into `str`, which is then displayed on the terminal as well as written into the file.

```
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <stdlib.h>

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

   char *str;
   FILE *fp;
   fp = fopen("textfile", "w+");
   if (fp == NULL) {
      printf("Error reading file");
  }
   str = (char *) malloc(atoi(argv[1]));
   strcpy(str, argv[2]);
   printf(str);
   fprintf(fp, str);
   free(str);
   return 1;
}
```