Questions – Project 3

1. How would you prevent the buffer overread necessary to implement the disclosure attack? Be specific about the fix.

The problem comes from the upload_A function, strncpy doesn't check or add the null terminator at the end of the string. I would explicitly memset a null terminator at the last byte (the 16th). This way we are assured to terminate a string when the user input is not valid. However, this fix would truncate the string but that's doesn't seem like a big deal…

I would do something like this for every string in the structure:

```
        if ( strncmp( ptr, "string_f", 8 ) == 0) {
                funcs[0]( a->string_f, ptr+9, STRLEN );
                memset(a->string_f, '\0', sizeof(char)*STRLEN);
        }
```

2. How would you prevent the buffer overflow that exploits the disclosure attack? Be specific about the fix.

The buffer overflow comes from the line 1046 in cse543-proto-p3.c:

```
        memcpy(buf+totalBytes, plaintext, outbytes );
```

The fix would be to check id the size of the buffer is large enough for the content something like this would do the trick:

```
                if ( (totalBytes + outbytes) <= (2 * sizeof(struct A)) ) {
                        memcpy(buf+totalBytes, plaintext, outbytes );
                }
                else assert ( 0 );
```

3. What is the best way to ensure safe string processing in the C language?

The good practice is to always check if the destination buffer is big enough to hold the content with the null terminator at the end.

The programmer can either truncate the content or reallocate a buffer big enough to hold the entire content. The use of assert is also recommended. There exists function in classic libraries which can do the checks for the programmer. We can use safer the bound checking solutions like strlcpy or strlcat. Those functions still require some work and understanding from the programmer to be safely used.