Exception Handling

Exception handling can help ameliorate the problem of indexing arrays outside their bounds. It allows the programmer to control what happens when an input string is too long. Exception handling is a complex subject, though, and you have to be very careful about what kinds of side effects might have occurred if an exception is thrown 10 subroutine calls deep from where it is caught. These examples were written in Java but the idea is the same for C++.

Your program may just want to give an error message saying the input was too long and then go about its business as if the user had never input anything in the first place. This is simple to implement:

```
/*** Give an error message ***/
char s[] = new char[10];
char input[] = GetSomeInput();

try
{
        for (int i = 0; i < input.length; i++)
        {
                s[i] = input[i];
        }
        GiveSomeOutput(s);
}
catch (ArrayIndexOutOfBoundsException e)
{
        GiveSomeOutput("The input was too long.");
}
```

This is not always an option, though. Suppose a certain kind of GPS packet is always 72 characters followed by “;*#” where ‘#’ is the number of satellites the GPS unit detects on its frequency. You’re always expecting something like “;*4” or “;*5”, so you make your buffer long enough to hold 75 characters. The attackers may not know the MAC code to give your GPS unit false data, but they transmit on the frequency with 10 simultaneous random bit streams. Your GPS unit sends valid packets but they’re 76 characters long because they end with “;*14”. You keep throwing these packets out to prevent a buffer overflow when you should be truncating them. Now you get no new GPS data and you don’t know where you are or what altitude you’re at so you end up crashing into a mountain.

Another thing you may want to do is just truncate the input to fit in your buffer. This can be done like so:

```
/*** Truncate the input to the size of your buffer ***/
char s[] = new char[10];
char input[] = GetSomeInput();

try
{
        for (int i = 0; i < input.length; i++)
        {
                s[i] = input[i];
        }
}
catch (ArrayIndexOutOfBoundsException e)
{
}

GiveSomeOutput(s);
```

In Java the **char** array s will hold a string of 10 characters, and the NULL character is not needed to terminate the string because the end of the array terminates the string. This is not true for C++, which always terminates strings with a NULL. Before you use special buffer classes in C++ that throw exceptions, be sure you understand how they work. If you declare a string in C++ as:

CMySafeStringClass s = new CMySafeStringClass(100);

…will it hold 99 characters or 100?

You may want to keep asking the user for input until they give you some input that is not too long for the buffer. This can be accomplished with a **while** loop:

```
/*** Loop until the input is an okay length ***/
char s[] = new char[10];
char input[];
boolean GoodInput = false;

while (!GoodInput)
{
        input = GetSomeInput();
        try
        {
                for (int i = 0; i < input.length; i++)
                {
                        s[i] = input[i];
                }
                GoodInput = true;
        }
        catch (ArrayIndexOutOfBoundsException e)
        {
                for (int i = 0; i < 10; i++)
                        s[i] = '\0';
        }
}

GiveSomeOutput(s);
```

You might limit the number of times they can try, though, so that they don't tie up the program and cause a denial-of-service.

Another valid thing to do is to allocate the buffer to be big enough to hold the input like this:

```
/*** Growing Buffer ***/
char s[];
char input[] = GetSomeInput();

s = new char[input.length];
for (int i = 0; i < input.length; i++)
{
        s[i] = input[i];
}

GiveSomeOutput(s);
```

The problem with this is that if you're running on a system with limited memory, like an embedded microprocessor, an exception may be thrown saying that there is not enough space on the heap if the attacker enters a very long string. You have to handle this exception, too, or else you'll be open to a denial-of-service attack.

This Document was Funded by the National Science Foundation Federal Cyber Service Scholarship For Service Program: Grant No. 0113627

For more information, go to: http://nsfsecurity.pr.erau.edu
or contact Susan Gerhart, gerharts@erau.edu
Embry-Riddle Aeronautical University • Prescott, Arizona • USA

Authored by Jedidiah R. Crandall