Project Name
libcurl

Source Code Version
7.30.0 (192c4f7)
POC Not Downloadable

CVE ID
CVE-2013-2174

Program Crash Procedure
1. Install Valgrind. (apt-get install valgrind)
2. Download libcurl v7.30.0
3. Run `./configure --enable-debug`
4. Run `make`
5. Run `make install`
6. Run `gcc -g input.c -o input -lcurl`
7. Run `valgrind ./input`

Crash Details
Program Location of Crash
`Curl_urldecode (escape.c:161)`

Program Location of Root Cause
`Curl_urldecode (escape.c:162)`

Curl_urldecode is a function that decodes a given URL encoded input string to a "plain string" and returns that in an allocated memory area. In URL, special characters are encoded into %XX where XX is the corresponding hexadecimal number. These characters are converted to their binary versions.

Curl_urldecode takes in a URL string and URL string length, among other things. In this POC, the parameters that are relevant are only the URL string, and the URL string length. If the user does not specify any URL string length (i.e. 0), the function calculates the strlen() of the URL string.

The vulnerability occurs when the user supplies a wrong length, resulting in an integer overflow of the loop counter and reading beyond the allocated buffer. In this case, we crafted a string “%FF” as the URL string and input 1 as the length. The length is stored as a size_t, which is an unsigned integer.

The function loops through each character in the URL string by decrementing the length as a counter, and checks if the character is ‘%’. If so, it determines if the next two characters in the buffer is a hexadecimal digit. Here, no check is performed to see if the next two characters are still within the buffer size. As a result, the program reads beyond the allowed buffer.

Furthermore, the length counter is decremented by 2 after the check above is executed. As the length counter is an unsigned integer, the counter wraps to a very large number. This causes the loop to continue even after length counter is exhausted. Also, the string pointer is incremented to read the character after %XX, which is beyond the original string provided. This causes a crash since we do not know what these characters are; they may point to an invalid or inaccessible memory segment, leading to a segmentation fault.

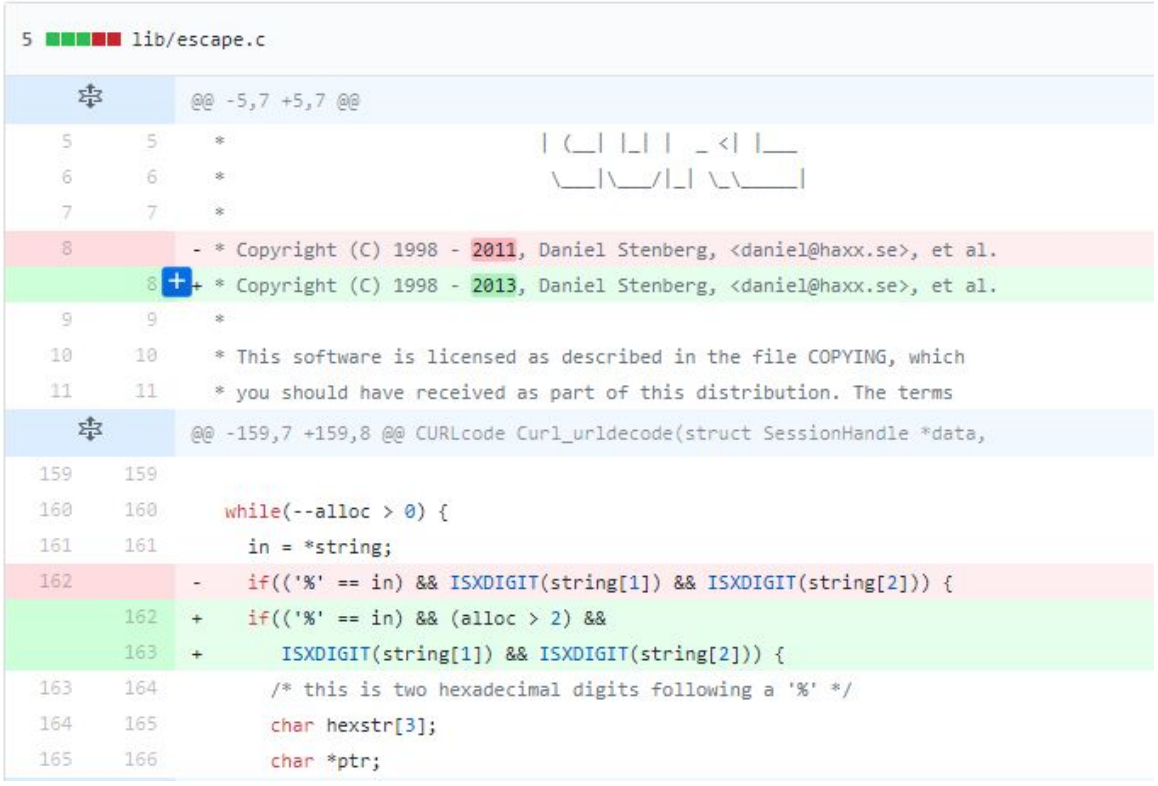
In the function, a buffer of size length+1 is allocated to write the decoded string. As the loop continues because of the integer overflow, more characters are written into the variable that stores the decoded string beyond the allocated buffer.

A malicious attacker can exploit this vulnerability by manipulating the string input. Since we have already established that the string can be read beyond its original memory space, the attacker can put a crafted string which points to any part of the user's memory. This is very dangerous since important user's information will be compromised, especially since libcurl is a very popular library.

Bug Fixes

The fix for this bug is to perform a check of the remaining length before reading the digits after '%' character. The check ensures that the length will never be reduced that an overflow happens and trigger the invalid read and write.

Commit that fixes the bug:



5 lib/escape.c

```
@@ -5,7 +5,7 @@
 *                                  | (__| |_| |  _ <| |___
 *                                   \___|\___/|_| \_\_____|
 *
- * Copyright (C) 1998 - 2011, Daniel Stenberg, <daniel@haxx.se>, et al.
+ * Copyright (C) 1998 - 2013, Daniel Stenberg, <daniel@haxx.se>, et al.
 *
 * This software is licensed as described in the file COPYING, which
 * you should have received as part of this distribution. The terms
@@ -159,7 +159,8 @@ CURLcode Curl_urldecode(struct SessionHandle *data,

   while(--alloc > 0) {
     in = *string;
-    if(('%' == in) && ISXDIGIT(string[1]) && ISXDIGIT(string[2])) {
+    if(('%' == in) && (alloc > 2) &&
+       ISXDIGIT(string[1]) && ISXDIGIT(string[2])) {
       /* this is two hexadecimal digits following a '%' */
       char hexstr[3];
       char *ptr;
```

Summary

Integer overflow often happens because no checking was performed on the user input, so a malicious input can trigger a crash, or even exploit the vulnerability. Developers need to take extra caution in writing operations that relies on an integers supplied by user input to prevent this problem.

References

http://www.cvedetails.com/cve-details.php?t=1&cve_id=CVE-2013-2174
https://curl.haxx.se/docs/adv_20130622.html
https://github.com/bagder/curl/commit/192c4f788d48f82c03e9cef40013f34370e90737