Project Name
libxml2

Source Code Version
2.7.2
POC Not Downloadable

CVE ID
CVE-2008-4225

Program Crash Procedure
1. Download libxml v2.7.2
2. Run `./configure`
3. Run `make`
4. Run `make install`
5. Move to directory with `input.c`
6. Run `gcc -o input `xml2-config --cflags` input.c `xml2-config --libs``
7. Because the input file is very large additional python script to generate the xml file is included in `generate_xml.py`.
8. [WARNING] This step will generate 4GB xml file. Generate large xml file by executing `python generate_xml.py`
9. Run `./input large.xml`

Crash Details
Program Location of Error
`xmlBufferResize (tree.c:6999)`

Program Location of Root Cause
`xmlBufferResize (tree.c:6999)`

The part of the program that caused the crash is responsible for allocating more buffer to its caller. The function takes in the current buffer and the desired new size. It checks if the current buffer's size is bigger than the new size; if the new size is bigger than the current size, the buffer size is incremented to match the demand.

One of the schemes to increase the buffer size is by doubling its current size. However, if the new size is very large, which is close to the unsigned integer limit, the current size cannot reach a number larger than the new size by simply doubling. The size will eventually overflow, and this continues indefinitely, causing an infinite loop. To illustrate this, let's assume that the maximum unsigned integer size is 0xffffffff. If the new size is also 0xffffffff, there is no way that the current

buffer size can be larger than the new size. When the current buffer is doubled, it will wrap back to a small value.

The POC that we created is a large XML file with a large number of spaces between the nodes. To optimize the parsing the XML, libxml2 attempts to remove the unnecessary spaces. This is done by loading the text into memory, and sufficient amount of buffer is needed and allocated by calling xmlBufferResize(). The rest happens as described above, the new size being inputted is very large and triggered the infinite loop.

Bug Fixes

https://bugzilla.redhat.com/attachment.cgi?id=322846&action=diff

(-)tree.c.orig (-2 / +8 lines)

Lines 14-20 Link Here

```
14 #include "libxml.h"
15
16 #include <string.h> /* for memset() only ! */
17
18 #ifdef HAVE_CTYPE_H
19 #include <ctype.h>
20 #endif
```

Lines 6996-7002 Link Here

```
6996         case XML_BUFFER_ALLOC_DOUBLEIT:
6997             /*take care of empty case*/
6998             newSize = (buf->size ? buf->size*2 : size + 10);
6999             while (size > newSize) newSize *= 2;
7000             break;
7001         case XML_BUFFER_ALLOC_EXACT:
7002             newSize = size+10;
```

```
14 #include "libxml.h"
15
16 #include <string.h> /* for memset() only ! */
17 #include <limits.h>
18 #ifdef HAVE_CTYPE_H
19 #include <ctype.h>
20 #endif
```

```
6996         case XML_BUFFER_ALLOC_DOUBLEIT:
6997             /*take care of empty case*/
6998             newSize = (buf->size ? buf->size*2 : size + 10);
6999             while (size > newSize) {
7000                 if (newSize > UINT_MAX / 2) {
7001                     xmlTreeErrMemory("growing buffer");
7002                     return 0;
7003                 }
7004                 newSize *= 2;
7005             }
7006             break;
7007         case XML_BUFFER_ALLOC_EXACT:
7008             newSize = size+10;
```

The fix for this bug is to add a check of the current allocated buffer size before doubling the amount if more is needed. A limit is set, which is half of the computer's maximum unsigned integer size. If this limit is reached, then the program will throw an error. No actual buffer is allocated inside this loop. The fix ensures that the buffer will not grow indefinitely and block the computer's resources with an infinite loop.

Summary

In conclusion, this vulnerability can be prevented by carefully setting a hard limit to a loop to ensure the loop exits every time. In this case, the program did not allocate any buffer in the loop so the memory is not affected, only the CPU. Nevertheless, developers should always be cautious in allowing a loop which uses user input as a stopping condition.

References

http://www.cvedetails.com/cve/CVE-2008-4225/
https://bugzilla.redhat.com/attachment.cgi?id=322846&action=diff
https://bugzilla.redhat.com/show_bug.cgi?id=470480