Talos Vulnerability Report

TALOS-2022-1572

Robustel R1510 web_server action endpoints OS command injection vulnerabilities

JUNE 30, 2022

CVE NUMBER

CVE-2022-33312,CVE-2022-33313,CVE-2022-33314

Summary

Multiple command injection vulnerabilities exist in the web_server action endpoints functionalities of Robustel R1510 3.3.0. A specially-crafted network request can lead to arbitrary command execution. An attacker can send a sequence of requests to trigger these vulnerabilities.

Tested Versions

Robustel R1510 3.3.0

Product URLs

R1510 - https://www.robustel.com/en/product/r1510-industrial-cellular-vpn-router/

CVSSv3 Score

9.1 - CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H

CWE

CWE-78 - Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’)

Details

The R1510 is an industrial cellular router. It offers several advanced software like an innovative use of Open VPN, Cloud management, data over-use guard, smart reboot and others.

The R1510 has a web server that manages several endpoints. One group of endpoints have the following form `/action/<API_endpoint>/`. Several of those endpoints use unsafe functions with user provided parameters, like the standard `system` function, and a custom one called `sysprintf`.

Here it is `sysprintf`:

```
void sysprintf(char *format_string,char *param_2,char *char*,char *param_4)

{
  [...]

  va_list_ptr = va_list;
  va_list[0] = param_2;
  va_list[1] = char*;
  va_list[2] = param_4;
  vsnprintf(shell_command,0x200,format_string,va_list_ptr);
[1]
  system(shell_command);
[2]
  return;
}
```

At `[1]` a string is formatted, using the first argument of the function as format string and the others parameters as format string arguments. If one of the argument is controllable by an attacker a command injection would occur at `[2]`.

CVE-2022-33312 - /action/import_cert_file/ command injection

This command injection is in the `/action/import_cert_file/` API.

The function that handles that endpoint is:

```
void /action/import_cert_file/(Webs *webs)

{
 [...]

  [...]
      path = (char *)websGetVar(webs,"path",0);
[3]
      if ((path != (char *)0x0) &&
         (target_file = websGetVar(webs,"target_file",0), target_file != 0)) {
        hash = scaselessmatch(webs->method,"POST");
        iVar7 = 0;
        if (hash != 0) {
          pWVar1 = hashFirst((char)webs->files);
          while (pWVar1 != (WebsKey *)0x0) {
            ppcVar9 = *(char ***)&(pWVar1->content).value;
            hash = dir_exists(path);
[4]
            if (hash == 0) {
              sysprintf("mkdir -p %s",path);
[5]
            }
            [...]
 }            At `[3]` the variable `path` is fetched, then at `[4]` it is checked
if its value correspond to an existing directory. If the directory does not exist
the value will be used, at `[5]`, as argument of the `sysprintf` function. This can
lead to a command injection.
```

CVE-2022-33313 - /action/import_https_cert_file/ command injection

This command injection is in the `/action/import_https_cert_file/` API.

The function that handles that endpoint is:

```
void /action/import_https_cert_file/(Webs *webs)

{
  [...]

  [...]
      type_var = websGetVar(webs,"type",0);
      path_var = websGetVar(webs,"path",0);
[6]
      if ((type_var != 0) && (path_var != 0)) {
        iVar1 = scaselessmatch(webs->method,"POST");
        if (iVar1 != 0) {
          pWVar2 = hashFirst((char)webs->files);
          while (pWVar2 != (WebsKey *)0x0) {
            uploaded_location = *(undefined4 **)&(pWVar2->content).value;
            iVar1 = string_matched(type_var,"ca");
            if (iVar1 == 0) {
              iVar1 = string_matched(type_var,"private_key");
              if (iVar1 != 0) {
                path_formatted_value = "%s/server.key";
                goto LAB_00481458;
              }
            }
            else {
              path_formatted_value = "%s/server.crt";
LAB_00481458:
              path_formatted_value = (char *)sfmt(path_formatted_value,path_var);
[7]
            }
            if (path_formatted_value != (char *)0x0) {
              sysprintf("mv %s %s -f",*uploaded_location,path_formatted_value);
[8]
              [...]
}
```

At `[6]` the variable `path` is fetched, then at `[7]` it is used as argument to format a string based on another provided variable. The formatted string is then used at `[8]` as argument for the `sysprintf` function. This can lead to a command injection.

CVE-2022-33314 - /action/import_sdk_file/ command injection

This command injection is in the `/action/import_sdk_file/` API.

The function that handles that endpoint is:

```
void /action/import_sdk_file/(Webs *webs)

{
  [...]

  [...]
      path_param = websGetVar(webs,"path",0);
[9]
      if (path_param != 0) {
        websSetStatus(webs,200);
        websWriteHeaders(webs,0xffffffff,0);
        websWriteHeader(webs,"Content-Type","text/html");
        websWriteEndHeaders(webs);
        iVar1 = scaselessmatch(webs->method,"POST");
        if (iVar1 != 0) {
          pWVar4 = hashFirst((char)webs->files);
          while (pWVar4 != (WebsKey *)0x0) {
            ppcVar8 = *(char ***)&(pWVar4->content).value;
            iVar1 = dir_exists(path_param);
[10]
            if (iVar1 == 0) {
              sysprintf("mkdir -p %s",path_param);
[11]
            }
            [...]
}
```

At `[9]` the variable `path` is fetched, then at `[10]` it is checked if its value correspond to an existing directory. If the directory does not exist the value will be used, at `[11]` as argument of the `sysprintf` function. This can lead to a command injection.

Timeline

2022-06-27 - Initial vendor contact

2022-06-28 - Vendor Disclosure

2022-06-30 - Public Release

CREDIT

Discovered by Francesco Benvenuto of Cisco Talos.

TALOS-2022-1573 TALOS-2022-1571