Talos Vulnerability Report

TALOS-2022-1578

Robustel R1510 web_server /action/import_authorized_keys/ OS command injection vulnerability

OCTOBER 14, 2022

CVE NUMBER

CVE-2022-34850

SUMMARY

An OS command injection vulnerability exists in the web_server /action/import_authorized_keys/ functionality of Robustel R1510 3.1.16 and 3.3.0. A specially-crafted network request can lead to arbitrary command execution. An attacker can send a sequence of requests to trigger this vulnerability.

CONFIRMED VULNERABLE VERSIONS

The versions below were either tested or verified to be vulnerable by Talos or confirmed to be vulnerable by the vendor.

Robustel R1510 3.1.16
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 features like an innovative use of Open VPN, Cloud management, data over-use guard, smart reboot and others.

The R1510's web_server offers an API to add, for the logged users, the SSH authorization keys.

Here is the `/action/import_authorized_keys/` API that manages the imports of the SSH authorization keys:

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

{
  [...]
      memset(admin_ssh_folder,0,0x40);
      user_username = uci_get("user_management.username");
      snprintf(admin_ssh_folder,0x40,"/home/%s/.ssh",user_username);
      iVar1 = dir_exists(admin_ssh_folder);
      if (iVar1 == 0) {
        sysprintf("mkdir -p %s",admin_ssh_folder);
        sysprintf("chmod 700 %s",admin_ssh_folder);
      }
      iVar1 = scaselessmatch(webs->method,"POST");
      if (iVar1 != 0) {
        for (current_file = (WebsKey *)hashFirst(webs->files); current_file !=
(WebsKey *)0x0;
            current_file = (WebsKey *)hashNext(webs->files,current_file)) {
          sysprintf("cp -rf \"%s\" %s/authorized_keys",
                    ((current_file->content).WebsUpload)-
>temp_filename,admin_ssh_folder);              [1]
        }
      }
     [...]
}
```

This function will fetch the admin-chosen username, create the `/home/<ADMIN_USERNAME>/.ssh` folder and set the right permissions for it. Then it will iterate over the request's files and import those using the instruction at `[1]`. This instruction uses the `sysprintf` function that will first execute the `vsnprintf` function with the provided arguments, then use the output as argument for the `system` function. So, the instruction will copy the uploaded file from the temporary upload location to the `/home/<ADMIN_USERNAME>/.ssh/authorized_keys` folder, executing `cp -rf \"<TEMP_LOCATION>\" /home/<ADMIN_USERNAME>/.ssh/authorized_keys`.

The web_server offers the possibility to change the admin username. The API responsible to do so is `/ajax/webs_uci_set_super_user/`:

```
undefined4 /ajax/webs_uci_set_super_user/(Webs *webs)

{
  [...]

      new_username = websGetVar(webs,"new_username",0);
      is_not_empty = string_is_not_empty(new_username);
      if ((is_not_empty == 0) ||
         (does_not_match = string_reg_verify(new_username,
                                    "^[a-zA-Z0-9@\\#\\.\\$\\*\\!][a-zA-Z0-
9@\\#\\.\\$\\*\\!\\-]{4,32 }$"
                                   ), does_not_match == 0)) {
[2]
        old_password = websGetVar(webs,"old_password",0);
        is_error = string_reg_verify(old_password,
                                  "^[a-zA-Z0-9@\\#\\.\\$\\*\\!][a-zA-Z0-
9@\\#\\.\\$\\*\\!\\-]{4,32}$ "
                                 );
        if (is_error == 0) {
          new_password = websGetVar(webs,"new_password",0);
          is_error = string_reg_verify(new_password,
                                    "^[a-zA-Z0-9@\\#\\.\\$\\*\\!][a-zA-Z0-
9@\\#\\.\\$\\*\\!\\-]{4,32 }$"
                                   );
          if (is_error == 0) {
            current_password = uci_get("user_management.password");
            is_error = string_matched(old_password,current_password);
            if (is_error == 0) {
              websWrite(webs,"{name: \"%s\", reason: \"%s\"}","old_password","not
match");
            }
            else {
              is_error = string_is_not_empty(new_username);
              if (((is_error == 0) ||
                  (is_error = uci_set("user_management.username",new_username),
is_error == 0)) &&      [3]
                 (is_error = uci_set("user_management.password",new_password), iVar1
== 0)) {
                res = "OK";
              [...]
}
```

This function will take the request's `new_username` parameter and checks at `[2]`, if it is in a valid format. The same goes for the password parameters. Then, at `[3]`, the `new_username` is committed as the new one.

Because the `new_username` valid format is `^[a-zA-Z0-9@\\#\\.\\$\\*\\!][a-zA-Z0-9@\\#\\.\\$\\*\\!\\-]{4,32 }$`, it is allowed to use special characters that can be interpreted in the unix shell. So, crafting a specific admin username and then using the `/action/import_authorized_keys/` API would lead to a command injection at `[1]`.

TIMELINE

2022-07-13 - Vendor Disclosure

2022-10-14 - Public Release

CREDIT

Discovered by Francesco Benvenuto of Cisco Talos.

VULNERABILITY REPORTS | PREVIOUS REPORT | NEXT REPORT

TALOS-2022-1577 | TALOS-2022-1580