Ayrx's Blog

ProLink PRC2402M V1.0.18 Multiple Vulnerabilities

2021-07-11 - (5 min read)

All vulnerabilities mentioned in this post were tested against firmware version V1.0.18, older versions might be affected as well. Affected devices should be updated to V1.0.23 to resolve the issues.

Proof-of-concept exploits for all vulnerabilities can be found here: https://github.com/Ayrx/PRC2402M

Background

The admin console for the PRC2402M router consists of various CGI binaries in the `/cgi-bin` directory that can be accessed through HTTP POST requests.

All requests contain `page` form parameter that the CGI binary uses to route to a specific function within the binary. The routing is done in the `main` function of the CGI binary as a giant chunk of if-else conditionals.

```
00400db4    int32_t page_param = web_get@GOT(0x416de8, params, 0)  {"page"}
<snip>
00400dfc    if (strcmp@GOT(page_param, 0x416e08) == 0)  {"sysAdm"}
004012c4        set_sys_adm(params: params)
00400e18    else if (strcmp@GOT(page_param, 0x416e10) == 0)
00401138        set_ntp(params)
00400e34    else if (strcmp@GOT(page_param, 0x416e14) == 0)  {"loaddefault"}
00401360        load_default()
00400e50    else if (strcmp@GOT(page_param, 0x416e20) == 0)  {"sysCMD"}
00401374        set_sys_cmd(params: params)
00400e6c    else if (strcmp@GOT(page_param, 0x416e28) == 0)  {"repeatLastCMD"}
00401388        set_last_cmd()
00400e88    else if (strcmp@GOT(page_param, 0x416e38) == 0)  {"wzdap"}
0040134c        set_wzdap(params: params)
00400ea4    else if (strcmp@GOT(page_param, 0x416e40) == 0)  {"wzdrepeater"}
0040139c        set_wzdrepeater(params: params)
```

Post-Auth Command Injections

The admin console contained 3 command injection vulnerabilities that could be accessed after authentication. Exploiting the vulnerabilities follow the general pattern of generating a reverse shell with `msfvenom`, calling `curl` with the command injection to fetch the payload onto the router, and executing the payload to obtain a reverse shell. The reverse shell has the privilege of the `root` user account.

Each directory on GitHub contains scripts that can be executed as follows:

Run `generate_payload.sh` to generate a reverse shell payload. The following parameters are configurable:

1. `LHOST`
2. `LPORT`
3. `PAYLOAD_NAME`

Run a HTTP server serving the generated payload:

```
python3 -m http.server 38888
```

Upload the generated payload to the router with `upload_payload.py`. The following parameters are configurable:

1. `TARGET_URL`
2. `PAYLOAD_URL`

Run a netcat listener to await for a reverse shell callback:

```
nc -nlvp 4242
```

sysCMD command injection (CVE-2021-36706)

The `set_sys_cmd` function in the `adm.cgi` binary, accessible with a `page` parameter value of `sysCMD` contains a trivial command injection where the value of the `command` parameter is passed directly to `system`.

```
int32_t set_sys_cmd(char* params)
00401f40  char* command_param = web_get(0x4154f4, params, 0)  {"command"}
00401f5c  char cmd[0x100]
00401f5c  memset(&cmd, 0, 0x100)
<snip>
00401fcc      system(&cmd)
00401fe8      sprintf(&cmd, 0x415554, command_param)  {"echo "%s" > /var/last_system_
00402000      system(&cmd)
00402030      fd = web_redirect(getenv(0x41557c))  {"HTTP_REFERER"}
00402044  return fd
```

ledonoff command injection (CVE-2021-36707)

The `set_ledonoff` function in the `adm.cgi` binary, accessible with a `page` parameter value of `ledonoff` contains a trivial command injection where the value of the `led_cmd` parameter is passed directly to `do_system`.

```
int32_t set_ledonoff(char* params)
00410ad0  char* led_cmd_param = strdup(web_get(0x416b64, params, 0))  {"led_cmd"}
<snip>
00410b1c  return do_system(0x415528, led_cmd_param) __tailcall  {"%s"}
```

`do_system` is a wrapper function that essentially calls out to `system`.

TR069 command injection (CVE-2021-36705)

The `set_TR069` function in the `adm.cgi` binary, accessible with a `page` parameter value of `TR069` contains a trivial command injection where the value of the `TR069_local_port` parameter is passed directly to `system`.

```
int32_t set_TR069(char* params)
0040883c  char cmd[0x100]
0040883c  memset(&cmd, 0, 0x100)
00408874  char* local_enable_param = strdup(web_get(0x4160c8, params, 0))  {"TR069_l
004088ac  char* acs_url_param = strdup(web_get(0x4160dc, params, 0))  {"TR069_acs_ur
004088e0  char* acs_username_param = strdup(web_get(0x4160ec, params, 0))  {"TR069_a
00408914  char* acs_password_param = strdup(web_get(0x416100, params, 0))  {"TR069_a
00408948  char* local_port_param = strdup(web_get(0x416114, params, 0))  {"TR069_loc
<snip>
00408b5c  sprintf(&cmd, 0x416190, local_port_param)  {"iptables -t filter -I INPUT -
00408b74  system(&cmd)
00408b90  sprintf(&cmd, 0x4161c8, local_port_param)  {"iptables -t filter -I INPUT -
00408ba8  system(&cmd)
00408bf0  return do_system(0x416200)  {"easycwmpd &"}
```

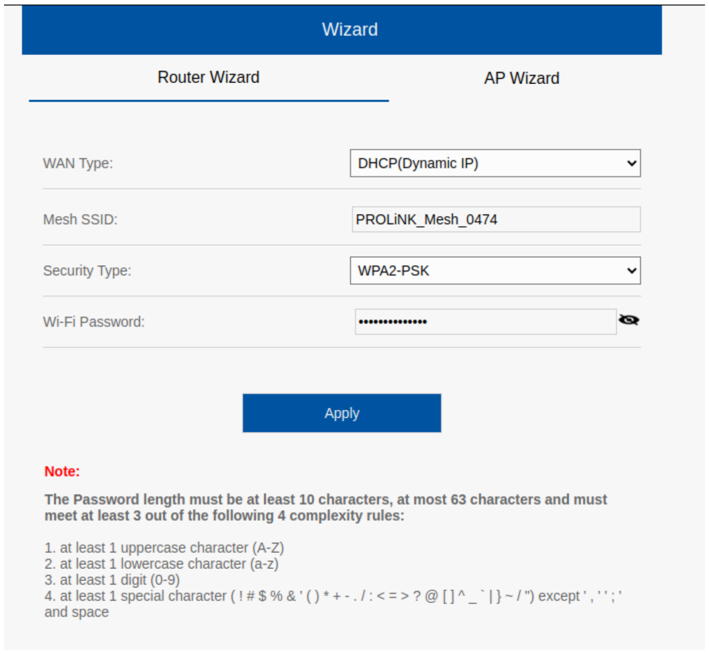
sysinit password reset (CVE-2021-36708)

The `set_sys_init` function in the `login.cgi` binary, accessible with a `page` parameter value of `sysinit` is used during the router setup process to configure the password of the admin account.

```
int32_t set_sys_init(char* params)
<snip>
00404328  char* newpass_param = strdup(web_get(0x407144, params, 0))  {"newpass"}
<snip>
00404728  nvram_bufset(0, 0x407240, 0x40747c)  {"ntpInit"}
00404744  nvram_bufset(0, 0x407248, time_zone_param)
00404760  nvram_bufset(0, 0x406ce4, newpass_param)  {"Password"}
00404780  nvram_bufset(0, 0x406cf0, 0x406fe8)  {"UserInit"}
```

The function is *still* accessible as an unauthenticated user after the initial setup and could be used to change the password to any attacker-controlled value.

After the password is reset, the only configuration that needs to be re-done is WiFi SSID / password and WAN type in the wizard displayed after logging in. All other configurations made in the router remain untouched.



WAN Type:	DHCP(Dynamic IP)
Mesh SSID:	PROLiNK_Mesh_0474
Security Type:	WPA2-PSK
Wi-Fi Password:	••••••••••••••

Note:

The Password length must be at least 10 characters, at most 63 characters and must meet at least 3 out of the following 4 complexity rules:

1. at least 1 uppercase character (A-Z)
2. at least 1 lowercase character (a-z)
3. at least 1 digit (0-9)
4. at least 1 special character (! # $ % & ' ( ) * + - . / : < = > ? @ [ ] ^ _ ` | } ~ / ") except ' , ' ' ; ' and space

This vulnerability could be combined with any of the command injections mentioned above to gain a root shell on the router which would go unnoticed by regular users of the router until they try and log in to the admin console again.

Timeline

- 21 May 2021 - Vulnerability reported to the vendor.
- 10 June 2021 - Vulnerability fixed by vendor.
- 25 June 2021 - Updated firmware published at https://prolink2u.com/download/firmware-prc2402m/
- 11 July 2021 - Published write up.
- 6 August 2021 - MITRE assigned CVE IDs

‹ Binary Ninja Rust Hello World

Generating Binary Ninja Dash Docset ›