Vendor of the products: D-Link

Affected Device: D-Link DI-7300G+、DI-7200G+V2、DI-8200G

Version: DI-7300G+ V19.12.25A1、DI_7200G+V2-24.04.18D1、DI_8200G-17.12.20A1

Firmware Download:http://www.dlink.com.cn/techsupport/ProductInfo.aspx?m=DI-7300G%2B

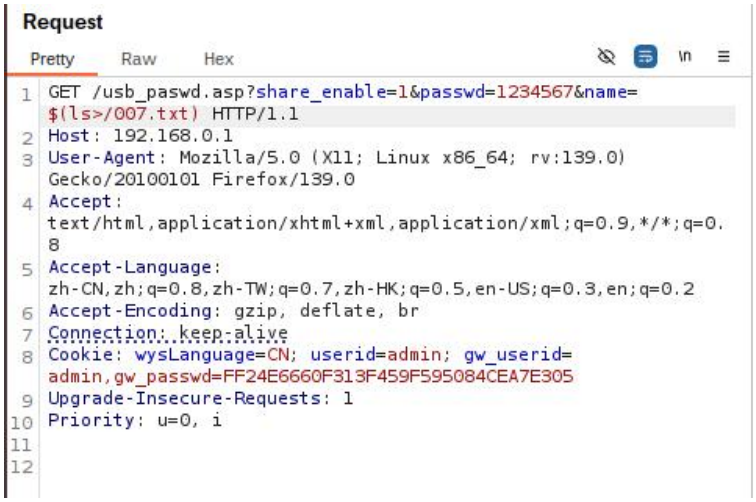
http://www.dlink.com.cn/techsupport/ProductInfo.aspx?m=DI-7000G%20V2%E7%B3%BB%E5%88%97

http://www.dlink.com.cn/techsupport/ProductInfo.aspx?m=DI-8200G

Vulnerability Description: A command injection vulnerability was discovered in D-Link DI-7300G+ V19.12.25A1, DI_7200G+V2-24.04.18D1, and DI_8200G-17.12.20A1, triggered by the name and hname parameters in usb_paswd.asp. Attackers can exploit this vulnerability by crafting malicious packets to execute arbitrary commands, thereby gaining full control of the target device.

POC:

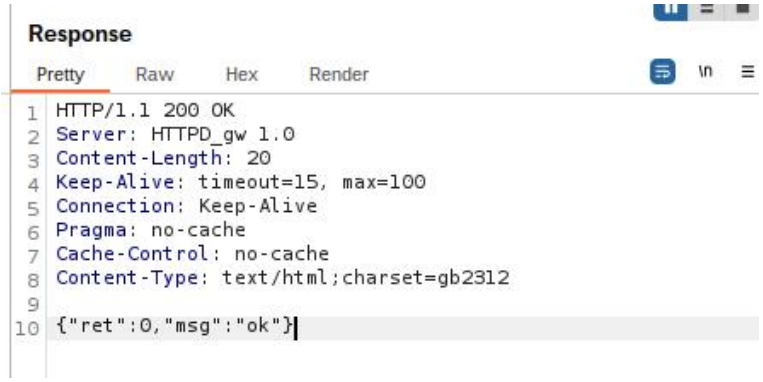
request1：

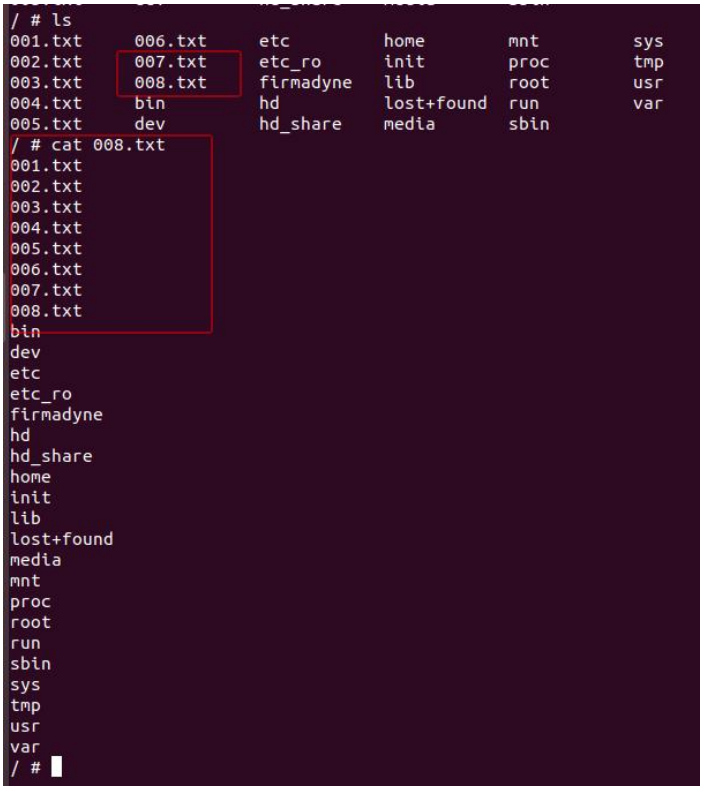


request2:

Vulnerability Effect:

It can be observed that the router receives the request and successfully executes the command.





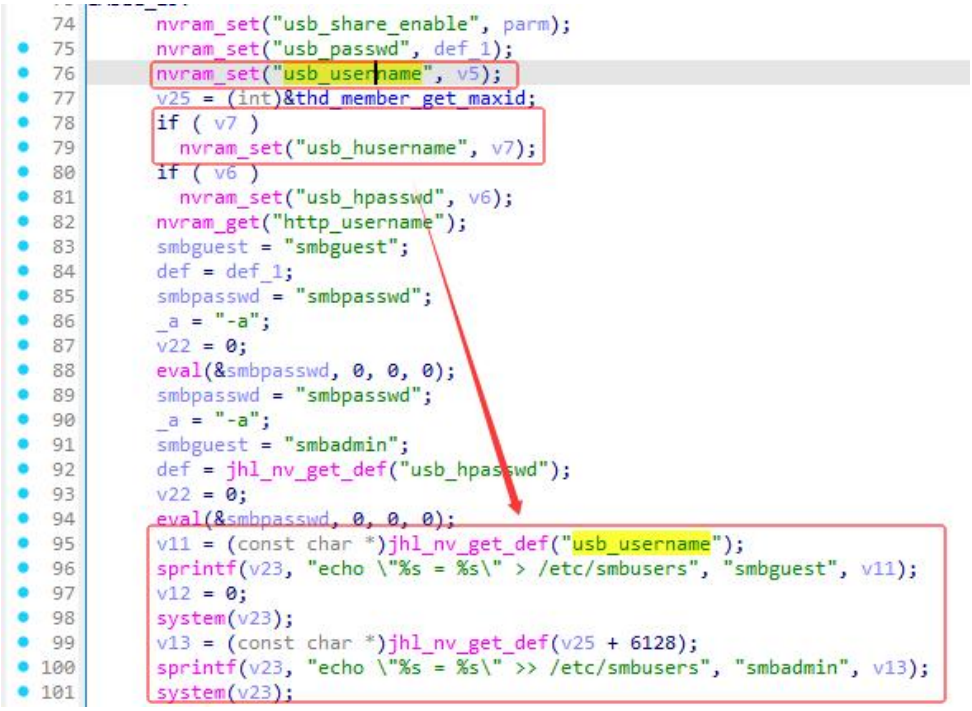
Vulnerability Cause:

The issue resides in the jhttpd component. In jhttpd, the program invokes the sub_4621DC function to handle requests related to usb_paswd.asp. The program first retrieves the user-input parameters name and hname via httpd_get_parm.

```
// usb_paswd.asp
int __fastcall sub_4621DC(int a1)
{
  _BYTE *parm; // $s3
  int def_1; // $s1
  char *v5; // $s6
  int v6; // $s4
  int v7; // $s2
  int v8; // $fp
  int v9; // $s7
  const char *_9100; // $s5
  const char *v11; // $v0
  int v12; // $s4
  const char *v13; // $v0
  int v14; // $v0
  int v15; // $v0
  int v16; // $a0
  int n20; // $a2
  const char *smbpasswd; // [sp+20h] [-538h] BYREF
  const char *_a; // [sp+24h] [-534h]
  const char *smbguest; // [sp+28h] [-530h]
  int def; // [sp+2Ch] [-52Ch]
  int v22; // [sp+30h] [-528h]
  _BYTE v23[256]; // [sp+34h] [-524h] BYREF
  char ___ret__:0___msg__:__ok___[1028]; // [sp+134h] [-424h] BYREF
  int v25; // [sp+538h] [-20h]
  int v26; // [sp+53Ch] [-1Ch]
  int v27; // [sp+540h] [-18h]
  int v28; // [sp+544h] [-14h]
  int v29; // [sp+548h] [-10h]
  int v30; // [sp+54Ch] [-Ch]
  int v31; // [sp+550h] [-8h]
  int v32; // [sp+554h] [-4h]

  if ( httpd_get_parm(a1, "opt") )
    return usb_email_asp(a1);
  parm = (_BYTE *)httpd_get_parm(a1, "share_enable");
  def_1 = httpd_get_parm(a1, "passwd");
  v5 = (char *)httpd_get_parm(a1, "name");
  v6 = httpd_get_parm(a1, "hpasswd");
  v7 = httpd_get_parm(a1, "hname");
  v26 = httpd_get_parm(a1, "acc_ip");
  v8 = httpd_get_parm(a1, "acc_mac");
  v27 = httpd_get_parm(a1, "acc_wan");
  v28 = httpd_get_parm(a1, "acc_auth");
  v29 = httpd_get_parm(a1, "device_name");
  v30 = httpd_get_parm(a1, "send_email_en");
  v31 = httpd_get_parm(a1, "send_email_name");
  v32 = httpd_get_parm(a1, "send_email_pwd");
  v9 = httpd_get_parm(a1, "printer_enable");
  _9100 = (const char *)httpd_get_parm(a1, "printer_port");
  killall_tk("smbd");
000621DC sub_4621DC:1 (4621DC)
```

Next, the program uses nvram_set to set the values of usb_username and usb_husername to the parameters v5 and v7 input by the user from the frontend. Subsequently, the program retrieves the values of the usb_username and usb_husername fields from NVRAM via the jhl_nv_get_def function. These values are concatenated using the sprintf function and directly passed to the system function for execution. Since no security checks are performed during this process, it leads to a command injection vulnerability. Attackers can execute arbitrary commands and fully control the device by constructing malicious parameters.


74 nvram_set("usb_share_enable", parm);
75 nvram_set("usb_passwd", def_1);
76 nvram_set("usb_username", v5);
77 v25 = (int)&thd_member_get_maxid;
78 if (v7)
79 nvram_set("usb_husername", v7);
80 if (v6)
81 nvram_set("usb_hpasswd", v6);
82 nvram_get("http_username");
83 smbguest = "smbguest";
84 def = def_1;
85 smbpasswd = "smbpasswd";
86 _a = "-a";
87 v22 = 0;
88 eval(&smbpasswd, 0, 0, 0);
89 smbpasswd = "smbpasswd";
90 _a = "-a";
91 smbguest = "smbadmin";
92 def = jhl_nv_get_def("usb_hpasswd");
93 v22 = 0;
94 eval(&smbpasswd, 0, 0, 0);
95 v11 = (const char *)jhl_nv_get_def("usb_username");
96 sprintf(v23, "echo \"%s = %s\" > /etc/smbusers", "smbguest", v11);
97 v12 = 0;
98 system(v23);
99 v13 = (const char *)jhl_nv_get_def(v25 + 6128);
100 sprintf(v23, "echo \"%s = %s\" >> /etc/smbusers", "smbadmin", v13);
101 system(v23);