D-Link COVR 12xx

CWE-78 OS Command Injection

0x1 OCI in SetNetworkTomographySettings()

Affected components

binary prog.cgi in firmware 101b01.

Attack vector

A user in the router's network can exploit the device by sending malicious http requests

Description

D-Link COVR 1200,1203,1210 v1.08 was discovered to contain a command injection vulnerability via the

tomography_ping_number parameter at function SetNetworkTomographySettings.

```
snprintf(v16, 64, "tomography_ping_number");
webGetVarN(a1, v16, (int)v18, 32);
v5 = 3908;
if ( !v18[0] )
  goto LABEL_2;
snprintf(v16, 64, "tomography_ping_size");
webGetVarN(a1, v16, (int)v19, 32);
v5 = 3913;
if ( !v19[0] )
  goto LABEL_2;
snprintf(v16, 64, "tomography_ping_timeout");
webGetVarN(a1, v16, (int)v14, 64);
v5 = 3918;
if ( v14[0] && (snprintf(v16, 64, "tomography_ping_ttl"), webGetVarN(a1, v16, (int)v15, 64), v5 = 3923, v15[0]) )
{
  snprintf(v13, 128, "tomography.ping.address=%s", v17);
```

```
    snprintf(v13, 128, "tomography.ping.ttl=%s", v15);
    if ( !sub_41D208((int)v13) )
    {
      v6 = 1;
      sub_41D974((int)"tomography", 0);
      snprintf(v13, 127, "ping -c %s -t %s -W %s -s %s %s > /var/ping_result 2>&1", v18, v15, v14, v19, v17);
      system(v13);
      v7 = a1;
      goto LABEL_3;
    }
```

0x2 OCI in SetNTPServerSettings()

Affected components

binary prog.cgi in firmware

Attack vector

A user in the router's network can exploit the device by sending malicious http requests

Description

D-Link COVR 1200,1202,1203 v1.08 was discovered to contain a command injection vulnerability via the system_time_timezone parameter at function SetNTPServerSettings.

```
webGetVar(a1, v9, v11, 32);
if ( v11[0] )
{
  snprintf(v10, 64, "system.@system[0].timezone=%s", v11);
  if ( sub_41D208((int)v10) )
  {
    v5 = 19;
    syslog(
      135,
      "%s:%s:%d:return ret error : SAVE_CONFIG_ERROR\n",
      "modules/management.c",
      "SetNTPServerSettings",
      3701);
  }
  else
  {
    v5 = 1;
    sub_41D974((int)"system", 0);
    sprintf(v8, "echo %s > /etc/TZ", v11);
    system(v8);
    system("/etc/init.d/sysntpd restart");
    system("date -k");
  }
```

0x3 OCI in SetTriggerWPS()

Affected components

binary prog.cgi in firmware

Attack vector

A user in the router's network can exploit the device by sending malicious http requests

Description

D-Link COVR 1200,1202,1203 v1.08 was discovered to contain a command injection vulnerability via the /SetTriggerWPS/PIN parameter at function SetTriggerWPS.

```
40        v9 = webGetVarString(a1, (int)"/SetTriggerWPS/PIN");
41        if ( v9 )
42        {
43          v10 = v12;
44          for ( i = 0; i != 2; ++i )
45          {
46            snprintf((int)v14, 128, "wireless.@wifi-iface[%d].wps_pin=%s", *v10, v9);
47            sub_41D208((int)v14);
48            snprintf((int)v14, 128, "wireless.wifi%d.disabled", i);
49            sub_41D560((int)v14, (int)v16, 16);
50            if ( !strcmp(v16, "0") )
51            {
52              snprintf(
53                (int)v15,
54                128,
55                "/usr/sbin/hostapd_cli -i ath%d -p /var/run/hostapd-wifi%d wps_pin any %s 120",
56                i,
57                i,
58                v9);
59              system(v15);
60            }
61            ++v10;
```

CWE-337 Predictable Seed in Pseudo-Random Number

0x1 PS in GetCAPTCHAsetting

Affected components

binary prog.cgi in firmware

Attack vector

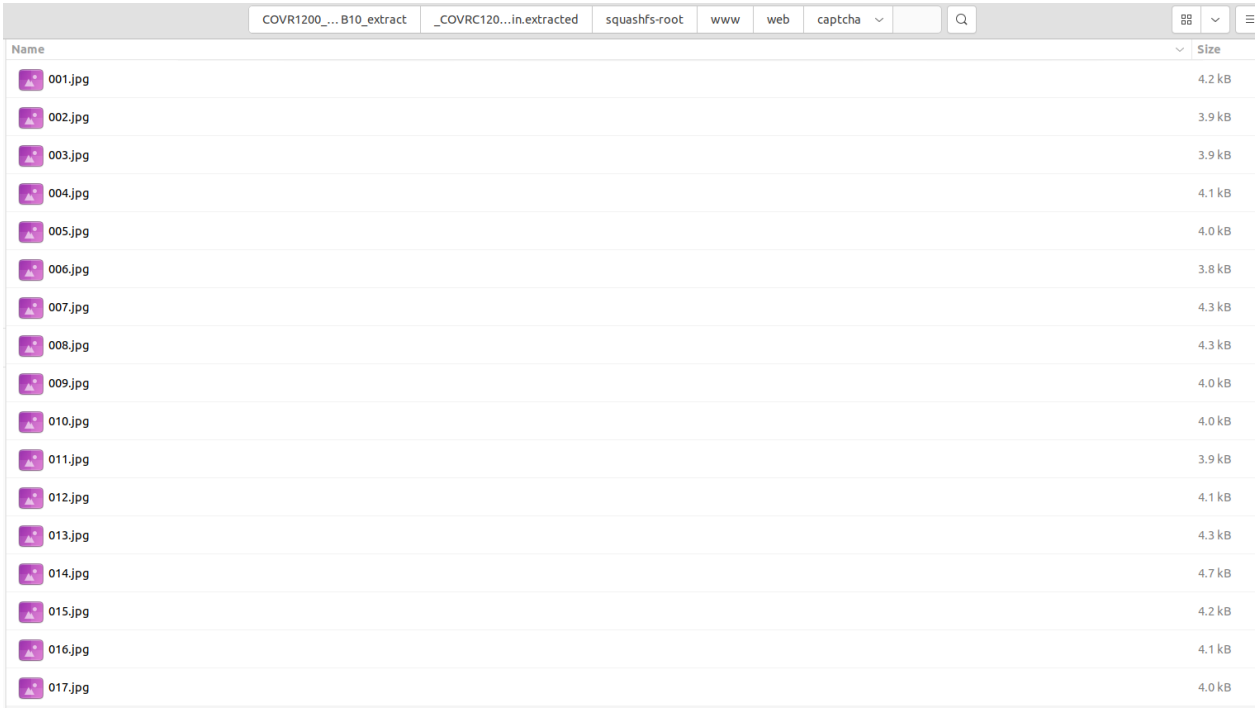
A user in the router's network can predict the CAPTCHA graph

Description

D-Link COVR 1200,1202,1203 v1.08 was discovered to have a predictable seed in a Pseudo-Random Number Generator in function GetCAPTCHAsetting.

In function GetCAPTCHAsetting(), the device uses the srand(time(0)) to random choose a captcha jpg in the path /www/web/captcha.

```
v2 = time(0);
srand(v2);
v4 = rand() % captcha_value;                    // predict the number of captcha graph
v3 = malloc(10);
*(_DWORD *)v3 = 0;
*(_DWORD *)(v3 + 4) = 0;
*(_WORD *)(v3 + 8) = 0;
tmp_captcha_name = (const char *)v3;
sub_437B28(10, v3);
sprintf((int)temp_captcha_jpg, "%s%s%s", "/www/web/docs/", tmp_captcha_name, ".jpg");
offset = 8 * v4;
sprintf((int)captcha_jpg, "%s%s%s", "/www/web/captcha/", *(const char **)(captcha_size + offset), ".jpg");
system("mkdir /www/web/docs");
sprintf((int)v34, "%s %s %s", "ln -s", captcha_jpg, temp_captcha_jpg);// make link
system("rm -f /www/web/docs/*.jpg");
system(v34);
sprintf((int)v30, "%s%s%s", "/docs/", tmp_captcha_name, ".jpg");
v7 = (int)tmp_captcha_name;
v8 = 0;
free(v7);
```



It makes a symbol link to the graph and finally response to the user to check at a default url

```
132   v13 = sub_41B4AC((int)v29, 0x2000);
133   if ( v13 < 0x2000 )
134     v13 += snprintf(
135               (int)&v29[v13],
136               0x2000 - v13,
137               "<GetCAPTCHAsettingResponse xmlns=\"http://purenetworks.com/HNAP1/\">\n"
138               "\t<GetCAPTCHAsettingResult>OK</GetCAPTCHAsettingResult>\n"
139               "\t<CaptchaUrl>%s</CaptchaUrl>\n"
140               "\t</GetCAPTCHAsettingResponse>",
141               v30);                          // Eventually, response a CaptchaUrl that point to the Captcha graph for the user
142   sub_41B534((int)&v29[v13], 0x2000 - v13);
143   sub_40A5C8(a1, (int)v29);
144   return 1;
145 }
```

However, the attack can predcit the captcha by knowing the time of requesting ..../GetCAPTCHAsetting function

CWE-1188 Insecure Default Initialization of Resource

0x1 Telnet enable in SetTelnetSettings()

Affected components

binary prog.cgi in firmware

Attack vector

A user in the router's network could exploit the device by sending malicious http request

Description

D-Link COVR 1200,1202,1203 v1.08 was discovered to enable a telnet and use a default TELNET account to get unauthorized access

```
sub_40A340((int)"SetTelnetSettings", (int)sub_42BEFC);
```

SBO

0x1 SBO in Main function(1)

Affected components

binary prog.cgi in firmware

Attack vector

A unauthentication user in the router's network can exploit the device by sending malicious http requests

Description

D-Link COVR 1200,1203,1210 v1.08 was discovered to contain a stack buffer overflow vulnerability that does not require authentication via the REQUEST_URI header.

If the value of QUERY_STING is not None, it reads the value REQUEST_URI

```
146     v7 = getenv("REQUEST_METHOD");
147     v8 = (const char *)v7;
148     if ( v7 )
149     {
150       v5[49] = sub_40D7F0((char *)v7, 4655600, 213);
151       v9 = getenv("HTTP_SOAPACTION");
152       v5[52] = sub_40D7F0((char *)v9, 4655600, 214);
153       v10 = getenv("HTTP_REFERER");
154       v5[50] = sub_40D7F0((char *)v10, 4655600, 215);
155       v11 = (_BYTE *)getenv("QUERY_STRING");
156       v5[48] = v11;
157       if ( v11 && !*v11 )
158         v5[48] = getenv("REQUEST_URI");
159       syslog(135, "%s:%s:%d:wp->path:%s,pmethod:%s\n\n", "fastcgi.c", "main", 221, (const char *)v5[48], v8);
```

Once the value of REQUEST_URI equals "method=xxxxxxx.....x&", the length is not check and is copied to v64, which is a 254 length buffer on the stack

```
168         v12 = getenv("HTTP_CONTENT_LENGTH");
169         v0 = strtol(v12, 0, 10) + 1;
170         if ( v0 < 2 )
171           syslog(135, "%s:%s:%d:get no data:%d!\n", "fastcgi.c", "main", 227, v0);
172         REQUEST_URI = (char *)v5[48];
173         if ( REQUEST_URI || v5[52] )
174         {
175           v14 = strstr(REQUEST_URI, "method=");
176           v15 = (char *)(v14 + 7);
177           if ( v14 )
178           {
179             v16 = strchr(v14 + 7, '&');
180             if ( v16 )
181             {
182               memset((int)v64, 0, 254);
183               strncpy(v64, v15, v16 - (_DWORD)v15);// method=aaaaa......aa& the length of string between method and & not check
184               v17 = v64;
185               v18 = 240;
186             }
187             else
188             {
```

0x2 SBO in websGetAuthCodeTime

Affected components

binary prog.cgi in firmware

Attack vector

A unauthentication user in the router's network can exploit the device by sending malicious http requests

Description

D-Link COVR 1200,1203,1210 v1.08 was discovered to contain a stack buffer overflow vulnerability that does not require authentication via the HNAP_AUTH header.

```
13  memset((int)v9, 0, sizeof(v9));
14  memset((int)v10, 0, sizeof(v10));
15  HNAP_AUTH = *(_DWORD *)(a1 + 204);
16  if ( HNAP_AUTH )
17  {
18    HNAP_AUTH_END = strchr(*(_DWORD *)(a1 + 204), ' ');// // a1 + 204 points to HNAP_AUTH
19    if ( HNAP_AUTH_END )
20    {
21      v5 = HNAP_AUTH_END + 1;
22      strncpy(v9, HNAP_AUTH, HNAP_AUTH_END - HNAP_AUTH);
23      v6 = strlen(v5);
24      strncpy(v10, v5, v6);
25      HIDWORD(v7) = strtoull(v10, 0, 10);
26      v2 = v7;
27    }
28  }
29  syslog(135, "%s:%s:%d:websGetAuthCodeTime webstime : %lld\n", "security.c", "websGetAuthCodeTime", 718, v2);
30  return v2;
31 }
```

The function calling chain is:

main()->websUrlHandlerRequest()->websSecurityHandler()->sub_416148()

(1) main

```
if ( strstr(v35, "FirmwareUpload") || strstr(v35, "ConfigFileUpload") || sub_46C5C8(v5) == 1 )
  websUrlHandlerRequest(v5);
```

(2) websUrlHandlerRequest

The way the request distribution works internally is to compare the http request path with the pre-defined path, and execute the corresponding function if they are equal

```
40    v6 = ((int (__fastcall *)(_DWORD *, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD))*v5)(
41          a1,
42          v5[2],
43          v5[1],
44          v5[4],
45          a1[56],
46          a1[48],
47          a1[61]) != 0;
48    result = 1;
49    if ( v6 )
```

(3) sub_419540 websSecurityHandler

Any request executes the callback function websSecurityHandler, and then hands it to other websFormHandler for processing.

```
v2 = getpid();
FCGI_fprintf(v1, "%d", v2);
FCGI_fclose(v1);
openlog("prog-cgi", 9, 168);
sub_419D94(1);
strcpy((int)v63, (int)"/www/web");
sub_409F38(v63);
sub_41300C(v62);
sub_41A474();
sub_411CA0(80, 5);
sub_411850(0, "websOpenServer \n");
sub_40D964((int)"/", 0, 0, (int)sub_419540, 1);
sub_40D964((int)"/HNAP1/", 0, 0, (int)sub_409F90, 0);
sub_40D964((int)"/cgi-bin", 0, 0, (int)sub_40905C, 0);
sub_40D964((int)"", 0, 0, (int)sub_409AC4, 2);
sub_411850(0, "websUrlHandlerDefine cgi-bin\n");
sub_41CB2C();
sub_4380C0();
sub_423460();
sub_45D918();
```

websSecurityHandler is used to process various login requests, when the /Login/Action request is matched,it goes to Auth_RequestHandler

```
1  int __fastcall sub_4193C4(int a1)
2  {
3    char *VarString; // $s1
4    int v3; // $s1
5
6    sub_419000();
7    if ( sub_414740(a1) )
8    {
9      VarString = webGetVarString(a1, (int)"/Login/Action");
10     syslog(135, "%s:%s:%d:action:%s \n", "security.c", "websAuthHandler", 3103, VarString);
11     if ( !VarString )
12       goto LABEL_6;
13     if ( !strncmp(VarString, "request", 7) )
14     {
15       AUTH_RequestHandle(a1);
16       return 1;
17     }
```

Auth_RequestHandler calls AddTempSessionList

```
62       sub_413F18(i);
63     }
64     sub_414624((int)v11, 20);
65     sub_414624((int)v12, 10);
66     sub_414624((int)v13, 20);
67     if ( !v3 || strncmp(v3, "Username", 8) )
68       sub_4156D0(v14, 64);
69     else
70       strncpy(v14, VarString, 64);
71     sub_414DC8((int)v11, (int)v14, (int)v13, (int)v10, 128);
72     v7 = AddTempSessionList(a1, v10, v11, v12, v13);
73     sub_4147A0(a1, v12);
74     sub_415B20(a1, 0);
75   }
```

AddTempSessionList calls websGetAuthCodeTime

```
LABEL_8:
    syslog(135, "%s:%s:%d:sizeof(ST_AUTH_TEMP)=%d \n", "security.c", "AddTempSessionList", 748, 96)
    v12 = malloc(96);
    v13 = v12;
    if ( v12 )
    {
      memset(v12, 0, 96);
      *(_DWORD *)(v13 + 24) = sub_40D7F0(*(char **)(a1 + 228), (int)"security.c", 760);
      *(_DWORD *)(v13 + 28) = sub_40D7F0(a3, (int)"security.c", 761);
      *(_DWORD *)(v13 + 32) = sub_40D7F0(a4, (int)"security.c", 762);
      *(_DWORD *)(v13 + 36) = sub_40D7F0(a5, (int)"security.c", 763);
      *(_DWORD *)(v13 + 40) = sub_40D7F0(a2, (int)"security.c", 764);
      *(_QWORD *)(v13 + 72) = websGetAuthCodeTime(a1);
      v14 = sub_4146F4();
      v15 = *(_DWORD *)(v13 + 76);
      v16 = *(_DWORD *)(v13 + 72);
      *(_QWORD *)(v13 + 64) = v14;
      sub_413E9C(a1, v17, v16, v15);
      v18 = dword_4AA554;
      if ( dword_4AA554 )
      {
        for ( i = *(_DWORD *)(dword_4AA554 + 4); i; i = *(_DWORD *)(i + 4) )
```

In websGetAuthCodeTime, there is stack buffer overflow at line 22, the copy length of strcnpy is not checked

```
1  __int64 __fastcall websGetAuthCodeTime(int a1)
2  {
3    __int64 v2; // $s3
4    int HNAP_AUTH; // $s1
5    int HNAP_AUTH_END; // $v0
6    int v5; // $s0
7    int v6; // $v0
8    __int64 v7; // $v1
9    char v9[512]; // [sp+2Ch] [-404h] BYREF
10   char v10[512]; // [sp+22Ch] [-204h] BYREF
11
12   v2 = 0LL;
13   memset((int)v9, 0, sizeof(v9));
14   memset((int)v10, 0, sizeof(v10));
15   HNAP_AUTH = *(_DWORD *)(a1 + 204);
16   if ( HNAP_AUTH )
17   {
18     HNAP_AUTH_END = strchr(*(_DWORD *)(a1 + 204), ' ');// // a1 + 204 points to HNAP_AUTH
19     if ( HNAP_AUTH_END )
20     {
21       v5 = HNAP_AUTH_END + 1;
22       strncpy(v9, HNAP_AUTH, HNAP_AUTH_END - HNAP_AUTH);
23       v6 = strlen(v5);
24       strncpy(v10, v5, v6);
25       HIDWORD(v7) = strtoull(v10, 0, 10);
UNKNOWN websGetAuthCodeTime:16 (414FB4)
```