详细的路由器漏洞分析环境搭建教程

2016-8-24 by 伐秦

声明：此教程内容完全是在**Debian 8.0**稳定版中进行的。添加更新源，系统更新，添加**sudo**，读者自己搞定**!**读者当然可以使用任意其他**Linux**发行版。至于为什么用**Debian**，最后有笔者的吐槽。

0x0 安装Git

因为Debian 8.0默认并没有安装Git，所以用以下命令 安装Git

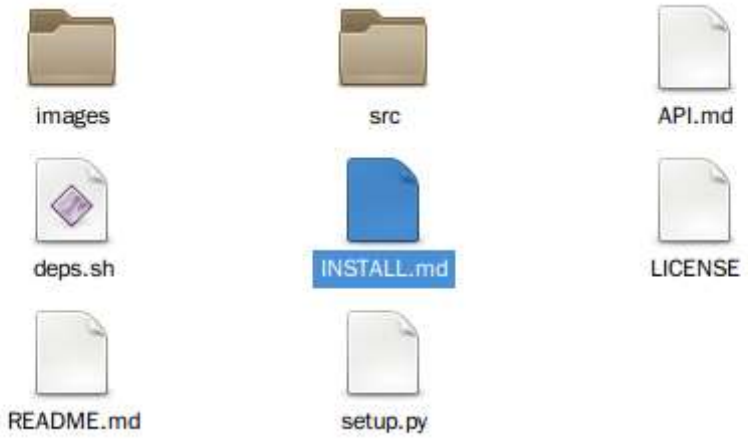
```
woody@debian:~$ sudo apt-get install git
```

0x1 安装Binwalk

首先在Github上面把Binwalk clone 下来。

```
woody@debian:~$ git clone https://github.com/devttys0/binwalk.git
正克隆到 'binwalk'...
remote: Counting objects: 6533, done.
接收对象中:  15% (998/6533), 404.01 KiB | 58.00 KiB/s
```

打开clone下来的文件夹，里面有个install.md这货就是安装binwalk之前要安装的其他一些依赖或者第三方工具。



用gedit打开install.md。按里面说的一个一个安装，但是呢会提示缺少其他的一些东西，所以这里先告诉大家，先装下面这几个东西

```
woody@debian:~$ sudo apt-get install build-essential autoconf
```

再按照install.md里面的内容来搞。你愿意装python2.7.x和3.x都可以，但是我比较喜欢2.7.x，就是喜欢，无理由

装完以后来试一下，就用论坛里面有前辈分析过的DIR-100*fw*reva*113ALLen*20110915.zip，先解压zip，然后得到的文件夹里面有个DIR100*v5.0.0EUb3*patch02.bix，用下面这条指令解包。

```
binwalk -Me DIR100_v5.0.0EUb3_patch02.bix
```

可以看到，多出来一个文件夹

打开它

这个文件夹里面就是我们要的固件。

如果你想试试解包出来的能不能运行，就按下面的这么做：
先安装这两个东东：

```
sudo apt-get install binfmt-support qemu-user-static
```

然后在Binwalk解包的固件目录下，执行这条指令

```
cp $(which qemu-mips-static) ./
```

接着，就可以执行下面这条指令来运行下固件里面的ifconfig看看

```
sudo chroot . ./qemu-mips-static ./bin/ifconfig
```

```
woody@debian:~/fuck/DIR-100_fw_reva_113_ALL_en_20110915/_DIR100_v5.0.0EUb3_pa
tch02.bix.extracted/squashfs-root$ sudo chroot . ./qemu-mips-static ./bin/ifc
onfig
ifconfig: Warning: cannot open /proc/net/dev. Limited output.: No such file o
r directory
br0       Link encap:Ethernet  HWaddr 00:0C:29:1C:A9:09
          inet addr:192.168.186.129  Bcast:192.168.186.255  Mask:255.255.255.
0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
```

0x2 安装Buildroot

Binwalk装完了，接下来转buildroot,到buildroot.org去下载，我下的是buildroot-2016.05.tar.bz2
然后安装这几个包

```
woody@debian:~$ sudo apt-get install  libncurses5-dev patch
[sudo] password for woody:
```

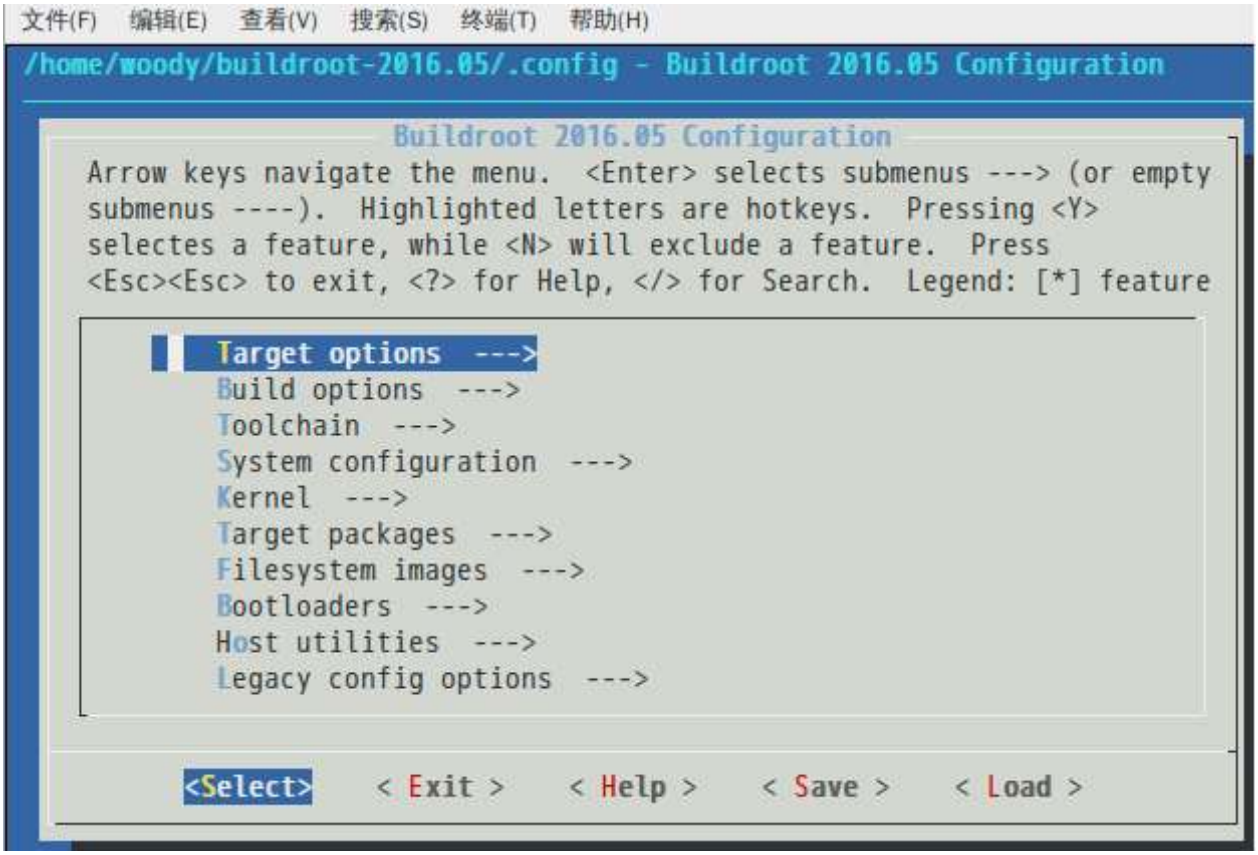
解压buildroot-2016.05.tar.bz2，切换到buildroot-2016.05目录，执行:

```
woody@debian:~/buildroot-2016.05$ make clean
```

接着执行:

```
woody@debian:~/buildroot-2016.05$ make menuconfig
```

弹出一个丑陋的对话框(嗯，我这么称呼它)



选择“Target options中的Target Architecture”，改成MIPS（分别是大端或小端），我这里就选小端了

然后选择Tollchain中的Kernel Headers 最好选择和你的内核版本一样的，我不知道为啥，有些是这么说，可能是万恶的兼容性吧，这里没有和我一一样的，所以我选择低于我内核版本的，3.14.x

```
/home/woody/buildroot-2016.05/.config - Buildroot 2016.05 Configuration
> Toolchain
                                  Toolchain
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  selectes a feature, while <N> will exclude a feature.  Press
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] feature

      Toolchain type (Buildroot toolchain)  --->
      (buildroot) custom toolchain vendor name (NEW)
          *** Kernel Header Options ***
          Kernel Headers (Linux 3.14.x kernel headers)  --->
          C library (uClibc)  --->
          *** uClibc Options ***
      (package/uclibc/uClibc-ng.config) uClibc configuration file to us
      ()  Additional uClibc configuration fragment files (NEW)
      [ ] Enable RPC support (NEW)
      [ ] Enable WCHAR support (NEW)
      (+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

save，退出:

执行 make 命令，接下来是漫长的等待

你说要多长时间，这说不好，反正我是编译时期又去睡了一觉

编译成功的shell内容大致是这样的:

```
ody/buildroot-2016.05/output/build/_users_table.txt /home/woody/buildroot-2016.0
5/output/target >> /home/woody/buildroot-2016.05/output/build/_fakeroot.fs
cat system/device_table.txt > /home/woody/buildroot-2016.05/output/build/_device
_table.txt
printf '        /bin/busybox                    f 4755 0  0 - - - - -\n /dev/co
nsole c 622 0 0 5 1 - - -\n' >> /home/woody/buildroot-2016.05/output/build/_devi
ce_table.txt
echo "/home/woody/buildroot-2016.05/output/host/usr/bin/makedevs -d /home/woody/
buildroot-2016.05/output/build/_device_table.txt /home/woody/buildroot-2016.05/o
utput/target" >> /home/woody/buildroot-2016.05/output/build/_fakeroot.fs
echo "  tar -cf /home/woody/buildroot-2016.05/output/images/rootfs.tar --numeric
-owner -C /home/woody/buildroot-2016.05/output/target ." >> /home/woody/buildroo
t-2016.05/output/build/_fakeroot.fs
chmod a+x /home/woody/buildroot-2016.05/output/build/_fakeroot.fs
PATH="/home/woody/buildroot-2016.05/output/host/bin:/home/woody/buildroot-2016.0
5/output/host/sbin:/home/woody/buildroot-2016.05/output/host/usr/bin:/home/woody
/buildroot-2016.05/output/host/usr/sbin:/usr/local/bin:/usr/bin:/bin:/usr/local/
games:/usr/games" /home/woody/buildroot-2016.05/output/host/usr/bin/fakeroot --
/home/woody/buildroot-2016.05/output/build/_fakeroot.fs
rootdir=/home/woody/buildroot-2016.05/output/target
table='/home/woody/buildroot-2016.05/output/build/_device_table.txt'
/usr/bin/install -m 0644 support/misc/target-dir-warning.txt /home/woody/buildro
ot-2016.05/output/target/THIS_IS_NOT_YOUR_ROOT_FILESYSTEM
woody@debian:~/buildroot-2016.05$
```

我们可以来使用编译出来的交叉编译工具来编译一个文件试验一下(说起来好拗口)，就用经典的helloworld

```
woody@debian:~/test$ /home/woody/buildroot-2016.05/output/host/usr/bin/mipsel-li
nux-gcc helloworld.c  -o helloworld.out -static
```

画红线的部分就是交叉编译工具与buildroot目录的相对路径

编译得到目标文件运行之，会发现这个错误提示

```
woody@debian:~/test$ ./helloworld.out
bash: ./helloworld.out: cannot execute binary file: 可执行文件格式错误
woody@debian:~/test$
```

当然了，因为我们编译出来的是MIPSEL机器的可执行文件呀~这时候就要安装QEMU了。

```
woody@debian:~/test$ sudo apt-get install qemu qemu-system
```

然后可以在MIPSEL环境下执行helloworld.out

```
woody@debian:~/test$ qemu-mipsel  ./helloworld.out
hello world
woody@debian:~/test$
```

成功了，这说明我们的交叉编译环境和QEMU都是安装成功滴。

0x3 选择QEMU-MIPS虚拟机映像

访问 https://people.debian.org/~aurel32/qemu/，下载MIPSEL的系统映像

mips/	2014-06-22 09:56	-
mipsel/	2014-06-22 09:55	-

当然选择mipsel，

Parent Directory		-
README.txt	2014-06-22 09:55	3.4K
debian_squeeze_mipsel_standard.qcow2	2013-12-09 00:56	270M
debian_wheezy_mipsel_standard.qcow2	2013-12-18 14:20	287M
vmlinux-2.6.32-5-4kc-malta	2013-09-24 13:00	6.6M
vmlinux-2.6.32-5-5kc-malta	2013-09-24 13:07	7.5M
vmlinux-3.2.0-4-4kc-malta	2013-09-21 01:39	7.7M
vmlinux-3.2.0-4-5kc-malta	2013-09-21 01:48	8.8M

至于这些文件下载哪个，下面的说明已经很清楚了。另外squeeze和wheezy是说的版本号。

```
To use this image, you need to install QEMU 1.1.0 (or later). Start QEMU
with the following arguments for a 32-bit machine:
  - qemu-system-mipsel -M malta -kernel vmlinux-2.6.32-5-4kc-malta -hda debian_squeeze_mipsel_standard.qcow2 -append "root=/dev/sda1 console=ttyO"
  - qemu-system-mipsel -M malta -kernel vmlinux-3.2.0-4-4kc-malta -hda debian_wheezy_mipsel_standard.qcow2 -append "root=/dev/sda1 console=ttyO"

Start QEMU with the following arguments for a 64-bit machine:
  - qemu-system-mips64el -M malta -kernel vmlinux-2.6.32-5-5kc-malta -hda debian_squeeze_mipsel_standard.qcow2 -append "root=/dev/sda1 console=ttyO"
  - qemu-system-mips64el -M malta -kernel vmlinux-3.2.0-4-5kc-malta -hda debian_wheezy_mipsel_standard.qcow2 -append "root=/dev/sda1 console=ttyO"
```

我就下vmlinux-2.6.32-5-4kc-malta和 debian*squeeze*mipsel_standard.qcow2.1

0x4 配置桥接网络

为了能够让QEMU虚拟机和宿主机都上网，并且互通，为动态调试做准备，必须要配置桥接网络
先装两个东西： bridge-utils和apt-get install uml-utilities 修改你的/etc/network/interfaces内容为：

```
auto lo
iface lo inet loopback
auto eth0
```

```
iface eth0 inet manual
up ifconfig eth0 0.0.0.0 up
auto br0
iface br0 inet dhcp
bridge_ports eth0
bridge_stp off
bridge_maxwait 1
```

编辑/etc/qemu-ifup， 换成下面的这个

```
#!/bin/sh
echo "Executing /etc/qemu-ifup"
echo "Bringing up $1 for bridged mode..."
sudo /sbin/ifconfig $1 0.0.0.0 promisc up
echo "Adding $1 to br0..."
sudo /sbin/brctl addif br0 $1
sleep 3
```

保存，重启。
重启之后，你的主机ifconfig是这样的

```
br0       Link encap:Ethernet  HWaddr 00:0c:29:1c:a9:09
          inet addr:192.168.186.129  Bcast:192.168.186.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe1c:a909/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:140 errors:0 dropped:0 overruns:0 frame:0
          TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:17239 (16.8 KiB)  TX bytes:11457 (11.1 KiB)

eth0      Link encap:Ethernet  HWaddr 00:0c:29:1c:a9:09
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:133 errors:0 dropped:0 overruns:0 frame:0
          TX packets:105 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:18773 (18.3 KiB)  TX bytes:12916 (12.6 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:29 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3221 (3.1 KiB)  TX bytes:3221 (3.1 KiB)
```

只要没有明显的保存就算是成功了
接下来呢就要用QEMU启动系统映像啦：

```
sudo qemu-system-mips -M malta -kernel vmlinux-2.6.32-5-4kc-malta
-hda debian_squeeze_mipsel_standard.qcow2.1
-append "root=/dev/sda1 console=tty0" -net nic, -net tap
```

执行以后，会弹出一个窗口，过一会儿系统的登陆界面就出来了

```
Debian GNU/Linux 7 debian-mipsel tty1

debian-mipsel login: _
```

用户名和密码全是root,登陆以后是这样的

```
Debian GNU/Linux 7 debian-mipsel tty1

debian-mipsel login: root
Password:
Linux debian-mipsel 3.2.0-4-4kc-malta #1 Debian 3.2.51-1 mips

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@debian-mipsel:~# pwd
/root
root@debian-mipsel:~#
```

查看虚拟机IP，在ping一下虚拟机

```
ifc     root@debian-mipsel:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:34:56
          inet addr:192.168.186.130  Bcast:192.168.186.255  Mask:2
```

woody@debian: ~

文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
woody@debian:~$ ping 192.168.186.130
PING 192.168.186.130 (192.168.186.130) 56(84) bytes of data.
64 bytes from 192.168.186.130: icmp_seq=1 ttl=64 time=0.363 ms
64 bytes from 192.168.186.130: icmp_seq=2 ttl=64 time=0.582 ms
64 bytes from 192.168.186.130: icmp_seq=3 ttl=64 time=0.282 ms
64 bytes from 192.168.186.130: icmp_seq=4 ttl=64 time=0.334 ms
64 bytes from 192.168.186.130: icmp_seq=5 ttl=64 time=0.525 ms
^C
--- 192.168.186.130 ping statistics ---
```

0x5 配置IDA动态调试

然后安装IDA Pro 我用的是论坛提供的6.8 在安装之前要先安装wine，wine的安装就不说了。很简单 然后：

```
sudo wine <ida安装程序路径>
```

会提示你装Mono装了吧，反正我装了。
像0D一样，可以直接调试，也可以附加调试，我们以之前binwalk解包的bin目录下的busybox为例
需要用到远程调试，所以我们需要建立一个脚本

```
#! /bin/bash

INPUT="$1"
LEN=$(echo -n "$INPUT" | wc -c)
PORT="1234"

if [ "$LEN" == "0" ] || [ "$INPUT" == "-h" ] || [ "$UID" != "0" ]
    then
    echo -e "\nUsage: sudo $0 \n"
exit 1
fi
cp $(which qemu-mips-static) ./qemu
echo "$INPUT" | chroot . /qemu -E CONTENT_LENGTH=$LEN -g $PORT /bin/busybox 2>/dev/null
```

这个脚本如果稍微学一下linux shell就可以看懂，所以我不在赘述。 运行之，

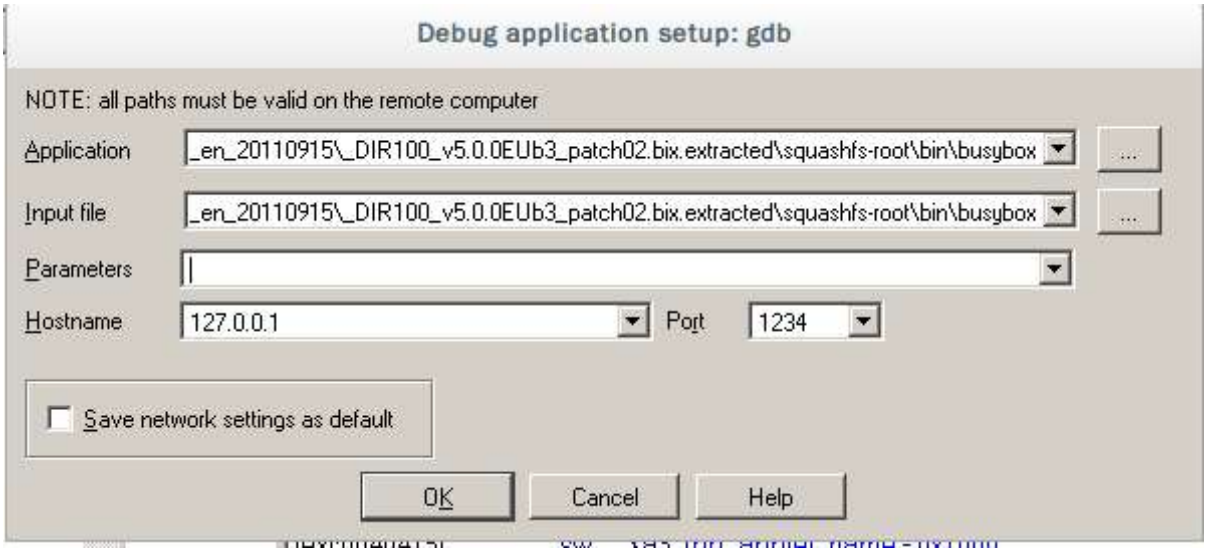
```
woody@debian:~/fuck/DIR-100_fw_reva_113_ALL_en_20110915/_DIR100_v5.0.0EUb3_patch
02.bix.extracted/squashfs-root$ sudo bash debug_busybox.sh "fuckyou"
```

直接调试：

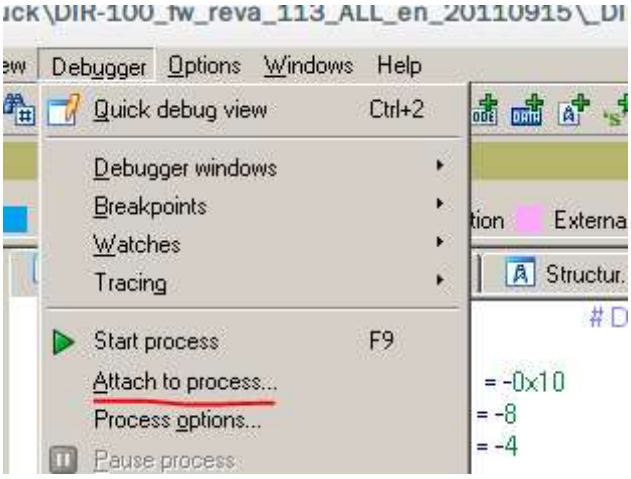
先用ida加载busybox,反汇编完成后，在main函数处插入断点

```
text:00404130                    # DATA XREF:
text:00404130
text:00404130 var_10      = -0x10
text:00404130 var_8       = -8
text:00404130 var_4       = -4
text:00404130
text:00404130          li    $gp, 0xFC04A20
text:00404138          addu  $gp, $t9
text:0040413C          addiu $sp, -0x20
text:00404140          sw    $gp, 0x20+var_10($sp)
text:00404144          sw    $ra, 0x20+var_4($sp)
text:00404148          sw    $gp, 0x20+var_8($sp)
text:0040414C          lw    $a3, 0($a1)
text:00404150          li    $v0, 0x2D
text:00404154          la    $at, bb_applet_name
text:00404158          nop
text:0040415C          sw    $a3, (bb_applet_name - 0x1000
text:00404160          lb    $v1, 0($a3)
```

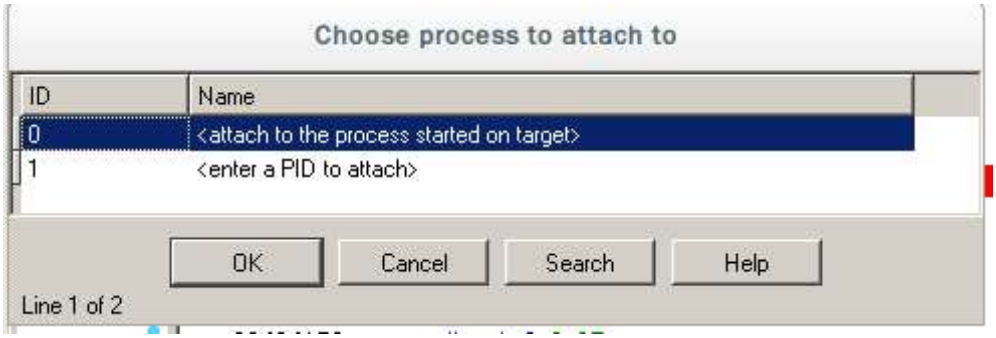
然后在Debuger中选择 Process Options,填上IP和端口，其他不变



然后再Debuger中选择附加进程



弹出一个对话框，然我们选择是附加到进程，还是选择，当然选第一个。



弹出一个对话框提示成功

点击OK，你发现没有断在我们之前设置的断点

```
MEMORY:767B9A7E .byte   0
MEMORY:767B9A7F .byte   0
MEMORY:767B9A80 # ---------------------------
MEMORY:767B9A80 bltzal  $zero, loc_767B9A88
MEMORY:767B9A84 nop
MEMORY:767B9A88
MEMORY:767B9A88 loc_767B9A88:                   # COD
MEMORY:767B9A88 la      $gp, unk_4CDD8
```

没关系，直接点运行，顺利中断在我们设置的断点处

```
.text:00404130 var_4= -4
.text:00404130
.text:00404130 li      $gp, 0xFC04A20
.text:00404138 addu    $gp, $t9
.text:0040413C addiu   $sp, -0x20
.text:00404140 sw      $gp, 0x20+var_10($sp)
.text:00404144 sw      $ra, 0x20+var_4($sp)
.text:00404148 sw      $gp, 0x20+var_8($sp)
.text:0040414C lw      $a3, 0($a1)
```

附加调试

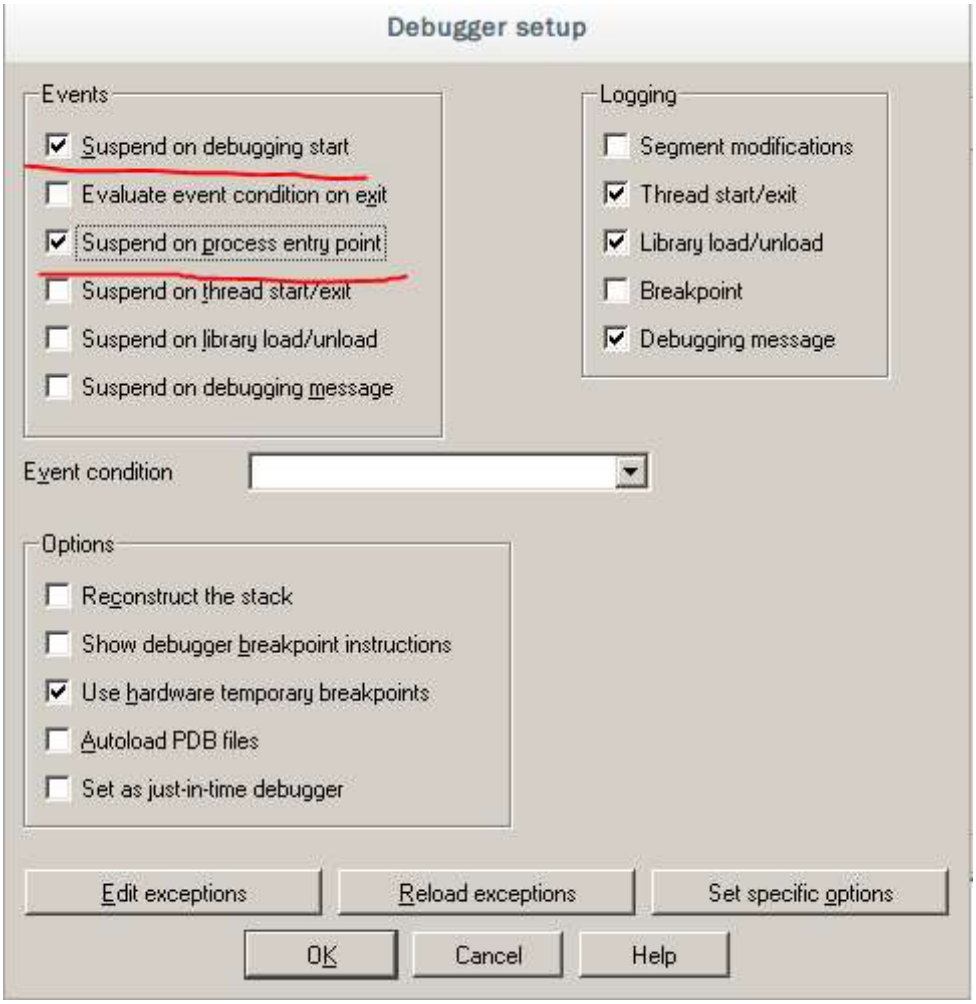
还是需要先运行这个脚本

```
woody@debian:~/fuck/DIR-100_fw_reva_113_ALL_en_20110915/_DIR100_v5.0.0EUb3_patch
02.bix.extracted/squashfs-root$ sudo bash debug_busybox.sh "fuckyou"
```

打开IDA，Debugger->attach选择GDB Romote Debugger

还是一样输入IP和端口

选择上面的Debug options



勾选如图两个事件，为啥呢，因为你附加的总要有中断的位置吧，就尝试在调试开始和程序入口点

在点Set specific options设置一下处理器为啥要自己设置？因为IDA没有事先分析代码呀，所以只能自己来啦，具体如红线所指

然后还是熟悉的提示

点OK 中断到调试器

吐槽与总结

1. Debian是我认为适合技术人员使用的发行版，没有之一，我还记得乌版图没完的bug,Arch升级之后各种程序挂了一片，gentoo繁琐的安装，等等等

2. binwalk，很好用，嗯，我第一点没有提Kali,现在说说，Kali默认安装了binwalk,但是始终不能把固件完全解包，出来的使各种html或者狗血文件，原因是什么?Kali中的binwalk也就是个binwalk，没有其他INSTALL.md中的各种模块，这点严重的坑到了我，之前用的kali 1.0，装模块是还会出现依赖问题，现在新版出来不知道解决没有，反正我用它祖宗Debian了。

3. linux坑太多，一定要慎重选择发行版，就算你选择Debian，也一定选择stable，testing对人的品格是一种拷问。

4. 善于翻文档，文档中有绝大部分问题的说明与解决办法。 5. 善于使用Google,尤其是硬件安全这种新兴方向，百度你能度出个鸡毛来。啥都别想，果断Google。

6. 英语，使用Google时经常出来的是英文页面，甚至是日文，日文怎么会，但是英文一定要看懂意思，不行就直接上Google翻译，虽然蛋疼点，总比一点都不懂强

7. 关于调试，很明显直接调试比附加调试好一点，直接调试断点随便设置，还能利用IDA强大的符号分析，岂不美哉！