安卓源码+内核修改编译(修改内核调试标志绕过反调试)

转载

Fly20141201 于 2017-02-17 14:24:05 发布 6194 收藏 6
分类专栏： Android Hook学习 文章标签： Android源码 androd内核 反调试


Android Hook学习 专栏收录该内容
29 篇文章 3 订阅
订阅专栏

标 题：【分享】安卓源码+内核修改编译(修改内核调试标志绕过反调试)
作 者：koflfy
时 间：2016-10-26,18:05:19
链 接：http://bbs.pediy.com/showthread.php?t=213481

历经两天时间，终于完整的编译完安卓操作系统源码+内核，并修改了内核的几个调试标志以达到绕过一些反调试的目的。
在此感谢同学辉哥以及群友f8的指点，如有错误或者遗露的地方，欢迎网友跟贴指正。

编译环境：
Ubuntu 14.04.4 for 64 http://releases.ubuntu.com/14.04/ubuntu-14.04.4-desktop-amd64.iso
jdk-6u45-linux-x64 http://download.oracle.com/otn/java/jdk/6u45-b06/jdk-6u45-linux-x64.bin?AuthParam=1467420548_35233aa4ba06eb66eb56a2a30757134b

手机：nexus 5
官方指导网址：http://source.android.com/source/index.html
编译 OS : Ubuntu 14 或者 Mac

安装vim编辑器
```
sudo apt-get install vim
```

一、环境配置
安装JDK
创建安装目录，在/usr/java下建立安装路径(主目录下，并非home目录)，并将下载好的jdk文件考到该路径下：

```
mkdir /usr/java
```

jdk-6u45-linux-x64 这个是自解压的文件，在linux上安装如下：

```
chmod 755 jdk-6u45-linux-x64.bin
```

（注意，这个步骤一定要在jdk-6u45-linux-x64 .bin所在目录下）

```
./jdk-6u45-linux-x64.bin
```

配置JDK，编辑/etc/profile文件

```
sudo vim /etc/profile
```

在里面添加如下内容

```
export JAVA_HOME=/usr/java/jdk1.6.0_45
export JAVA_BIN=/usr/java/jdk1.6.0_45/bin
export PATH=$PATH:$JAVA_HOME/bin
SSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export JAVA_HOME JAVA_BIN PATH CLASSPATH
```

让/etc/profile文件修改后立即生效 ,可以使用如下命令:

```
. /etc/profile
```

注意： . 和 /etc/profile 有空格.

重启查看java版本

```
java -version
```

屏幕输出:

```
java version "1.6.0_45"
Java(TM) SE Runtime Environment (build 1.6.0_45-b06)
Java HotSpot(TM) 64-Bit Server VM (build 20.45-b01, mixed mode)
```

安装编译库

```
sudo apt-get install git-core gnupg flex bison gperf build-essential \
zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 \
lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z-dev ccache \
libgl1-mesa-dev libxml2-utils xsltproc unzip
```

配置 USB 端口（查看后面 fastboot模式下不识别手机问题，该文件内容需要修改不能直接用于Nexus5）
下载51-android.rules文件(见附件，下载后把.doc扩展名去掉就行)，放
到 Ubuntu 的/etc/udev/rules.d/51-android.rules 目录下，
并根据手机的机型进行配置(把最后一个username字符串改成自己用户名)

repo源码
使用 repo 工具进行源码下载。
下面的方法是不需要翻墙的， 如果你认为自己的 VPN 够强大的话可以直接按照官网的指导来下载
设置 github 邮箱姓名使用 git 工具（不是必要的）

```
git config --global user.email "<你的 github 的 email 地址>"
git config --global user.name "<你的 github 的 name 名称>"
```

下载 repo 工具（清华的源）

```
git clone git://aosp.tuna.tsinghua.edu.cn/android/git-repo.git/
git clone https://aosp.tuna.tsinghua.edu.cn/android/git-repo.git
```

得到一个 git-repo 项目，找到里面的 repo 文件，复制到 ~/bin/repo 中
赋予执行命令

```
chmod a+x ~/bin/repo
```

修改下载 URL
双击打开 repo,修改 REPO_URL 为

```
REPO_URL = 'git://aosp.tuna.tsinghua.edu.cn/android/git-repo'
REPO_URL = 'https://aosp.tuna.tsinghua.edu.cn/android/git-repo'
```

然后使用

```
export PATH=~/bin:$PATH
```

导出 bin 执行目录

下载源码
新建一个目标文件夹用于存放源码文件，命令行中用 cd 指令进入。执行下面指令

```
repo init -u https://aosp.tuna.tsinghua.edu.cn/android/platform/manifest -
b android-4.4.4_r1
```

-b 后面的参数是刚才查到的 branch 号

repo 更新使用 repo sync 指令即可完成下载，也可以使用下面的脚本

```
export PATH=~/bin:$PATH
repo sync -j4
while [ "$?" != "0" ]; do
  sleep 30
  repo sync -j4
done
```

接下来在**官网下载手机对应的驱动**(如下)：
https://developers.google.com/android/nexus/drivers#hammerhead
找到 Nexus 5 (GSM/LTE) binaries for Android 4.4.4 (KTU84P)
下面三个文件都要下载，解压后是三个.sh文件，放到安卓源码目录下，分别对三个文件chmod a+x 文件名 赋权限，然后分别执行三个文件，此时会生成 vendor文件夹。

编译代码如果前面的步骤没出问题，那么用下面的指令就可以直接进行编译（aosp_arm-eng为模拟器，nexus5为aosp_hammerhead-userdebug），编译完后操作系统路径为 out/target/product/hammerhead下。

```
source build/envsetup.sh #设置编译环境
lunch aosp_hammerhead-userdebug #设置编译选项
export USE_CCACHE=1 #使用缓存，可以加快以后的编译速度
prebuilts/misc/linux-x86/ccache/ccache -M 100G #使用 100GB 来作为缓存的空间
export CCACHE_DIR=/<path_of_yourt_choice>/.ccache #设置缓存地址,可以不要这个命令而使用默认缓存路径
make -j4
```

以上可用初始化脚本如下：(init)

```
export JAVA_HOME=/usr/local/java/jdk1.6.0_45
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
. build/envsetup.sh
lunch aosp_hammerhead-userdebug
make -j4
```

此时编译的操作系统自带默认的内核，要修改内核的话，得重新下载内核源码并修改编译。在Android源码文件夹下创建kernel文件夹，并下载内核源码：（参考http://source.android.com/source/building-kernels.html）

1、git clone https://android.googlesource.com/kernel/msm.git
（由于实验手机设备为Nexus 5，因此我们选择内核代码为msm.git）

2、checkout所选内核版本

```
cd ~/source/kernel/msn
git branch -a
git checkout remotes/origin/android-msm-hammerhead-3.4-kitkat-rm1
```

3、**修改内核调试标志，绕过反调试**(Kernel proc)

要修改的文件：

```
kernel/msm/fs/proc/base.c
kernel/msm/fs/proc/array.c
```

要修改对以下文件的写入

```
Status,stat,
```

修改点：

base.c 第285行改成如下：

```
else {
            if (strstr(symname, "trace")) {
                 return sprintf(buffer, "%s", "sys_epoll_wait");
            }
            return sprintf(buffer, "%s", symname);
      }
```

array.c第134行改成如下：

```
      static const char * const task_state_array[] = {
            "R (running)",         /*    0 */
            "S (sleeping)",        /*    1 */
            "D (disk sleep)",      /*    2 */
            "S (sleeping)",        /*    4 */
            "S (sleeping)", /*     8 */
            "Z (zombie)",          /*  16 */
            "X (dead)",            /*  32 */
            "x (dead)",            /*  64 */
            "K (wakekill)",        /* 128 */
            "W (waking)",          /* 256 */
      };
```

array.c第187行改成如下：

```
      "Gid:\t%d\t%d\t%d\t%d\n",
                  get_task_state(p),
                  task_tgid_nr_ns(p, ns),
                  pid_nr_ns(pid, ns),
                  ppid, /*tpid*/0,
                  cred->uid, cred->euid, cred->suid, cred->fsuid,
                  cred->gid, cred->egid, cred->sgid, cred->fsgid);
```

修改完成后，按下面步骤编译内核，刷入系统

4、修改Makefile文件支持交叉编译器，编译内核。可用内核初始化脚本如下：

```
#How to build
export PATH='/home/coffee/source/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6/bin':$PATH
export ARCH=arm
export SUBARCH=arm
export CROSS_COMPILE=arm-eabi-
make hammerhead_defconfig
make -j4
```

编译成功后可以看到下面的输出：

```
OBJCOPY  arch/arm/boot/zImage
Kernel:     arch/arm/boot/zImage is ready
CAT           arch/arm/boot/zImage-dtb
Kernel:     arch/arm/boot/zImage-dtb is ready
```

编译完内核后，内核生成的路径为：kernel/msm/arch/arm/boot目录下的zImage-dtb文件(nexus 5是这个文件，nexus 4是zImage文件，请注意别弄错了)，把该文件复制到源码下的device/lge/hammerhead-kernel夹下（注意：nexus 4则为mako-kernel文件夹），覆盖掉同名文件，然后重新按上面步骤编译一次安卓源码（这时候编译很快完成）即可。

刷机

插上手机连接上电脑，确认正常连接上电脑后，进行刷机

转到源码img生成路径

```
cd source/out/target/product/hammerhead/
Adb reboot bootloader
Fastboot -w flashall
```

大概两分钟后，系统即刷成功。

附：
在 bootloader 下连不上手机的参考前两面的配置 USB 端口一节
编译指令
部分编译的
m：编译所有的模块
mm：编译当前目录下的模块，当前目录下要有 Android.mk 文件
mmm：编译指定路径下的模块，指定路径下要有 Android.mk 文件
部分编译完以后，需要使用
Make snod 来把编译的东东整合到镜像中

Bootloader解锁
我们买来的Nexus4默认都是锁了bootloader的，但是为了烧自己编的系统或者其他第三方ROM就需要解锁bootloader，google提供了接口让开发者可以方便的解锁。
先把Nexus4关机，然后同时按下两个音量键和power键，一直按住直到界面上出现bootloader的画面。
这时用usb线连上手机，在shell里执行：

```
$ fastboot oem unlock
```

fastboot模式下不识别手机问题。但adb模式正常

```
coffee @ coffee -pc:~$ lsusb
Bus 001 Device 010: ID 18d1:4ee0 Google Inc.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

修改51-android.rules内容为以下两种之一（mode必须是0666，idVendor 与 idProduct（如果有）要与实际一一对应）

```
http://stackoverflow.com/questions/7641939/android-fastboot-waiting-for-devices
# fastboot protocol on manta (Nexus 5)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4ee0", MODE="0666",
< coffee >"
# fastboot protocol on manta (Nexus 5)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", MODE="0666", OWNER="<coffee>"
```

修改完后用命令

```
sudo service udev restart
```

重启服务进程，重新拔插数据线再

```
fastboot devices
```

查看是否识别出来
上面make步骤若出现 make: nothing to be done for 'droid'
就mak clean一下。

附上手机USB识别的需要的51-android.rules文件：

```
# adb protocol on passion (Nexus One)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e12", MODE="0600", OWNER="coffee"
# fastboot protocol on passion (Nexus One)
SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", ATTR{idProduct}=="0fff", MODE="0600", OWNER="coffee"
# adb protocol on crespo/crespo4g (Nexus S)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e22", MODE="0600", OWNER="coffee"
# fastboot protocol on crespo/crespo4g (Nexus S)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e20", MODE="0600", OWNER="coffee"
# adb protocol on stingray/wingray (Xoom)
SUBSYSTEM=="usb", ATTR{idVendor}=="22b8", ATTR{idProduct}=="70a9", MODE="0600", OWNER="coffee"
# fastboot protocol on stingray/wingray (Xoom)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="708c", MODE="0600", OWNER="coffee"
# adb protocol on maguro/toro (Galaxy Nexus)
SUBSYSTEM=="usb", ATTR{idVendor}=="04e8", ATTR{idProduct}=="6860", MODE="0600", OWNER="coffee"
# fastboot protocol on maguro/toro (Galaxy Nexus)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e30", MODE="0600", OWNER="coffee"
# adb protocol on panda (PandaBoard)
SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d101", MODE="0600", OWNER="coffee"
# adb protocol on panda (PandaBoard ES)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="d002", MODE="0600", OWNER="coffee"
# fastboot protocol on panda (PandaBoard)
SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d022", MODE="0600", OWNER="coffee"
# usbboot protocol on panda (PandaBoard)
SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d00f", MODE="0600", OWNER="coffee"
# usbboot protocol on panda (PandaBoard ES)
SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d010", MODE="0600", OWNER="coffee"
# adb protocol on grouper/tilapia (Nexus 7)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e42", MODE="0600", OWNER="coffee"
# fastboot protocol on grouper/tilapia (Nexus 7)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e40", MODE="0600", OWNER="coffee"
# adb protocol on manta (Nexus 10)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4ee2", MODE="0600", OWNER="coffee"
# fastboot protocol on manta (Nexus 10)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4ee0", MODE="0600", OWNER="coffee"
# adb protocol on hammerhead (Nexus 5)
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4ee1", MODE="0600", OWNER="coffee"d
```

今天翻了下自己的收集的资料，发现看雪上有一篇比较详细的关于Android的源码和内核源码编译的帖子。尽管已经了解了怎么折腾，但是还是记录一下，谢谢看雪的这位兄弟。不管怎样，Andorid的源码和内核源码的编译还是谷歌官方的文档为参考基准。