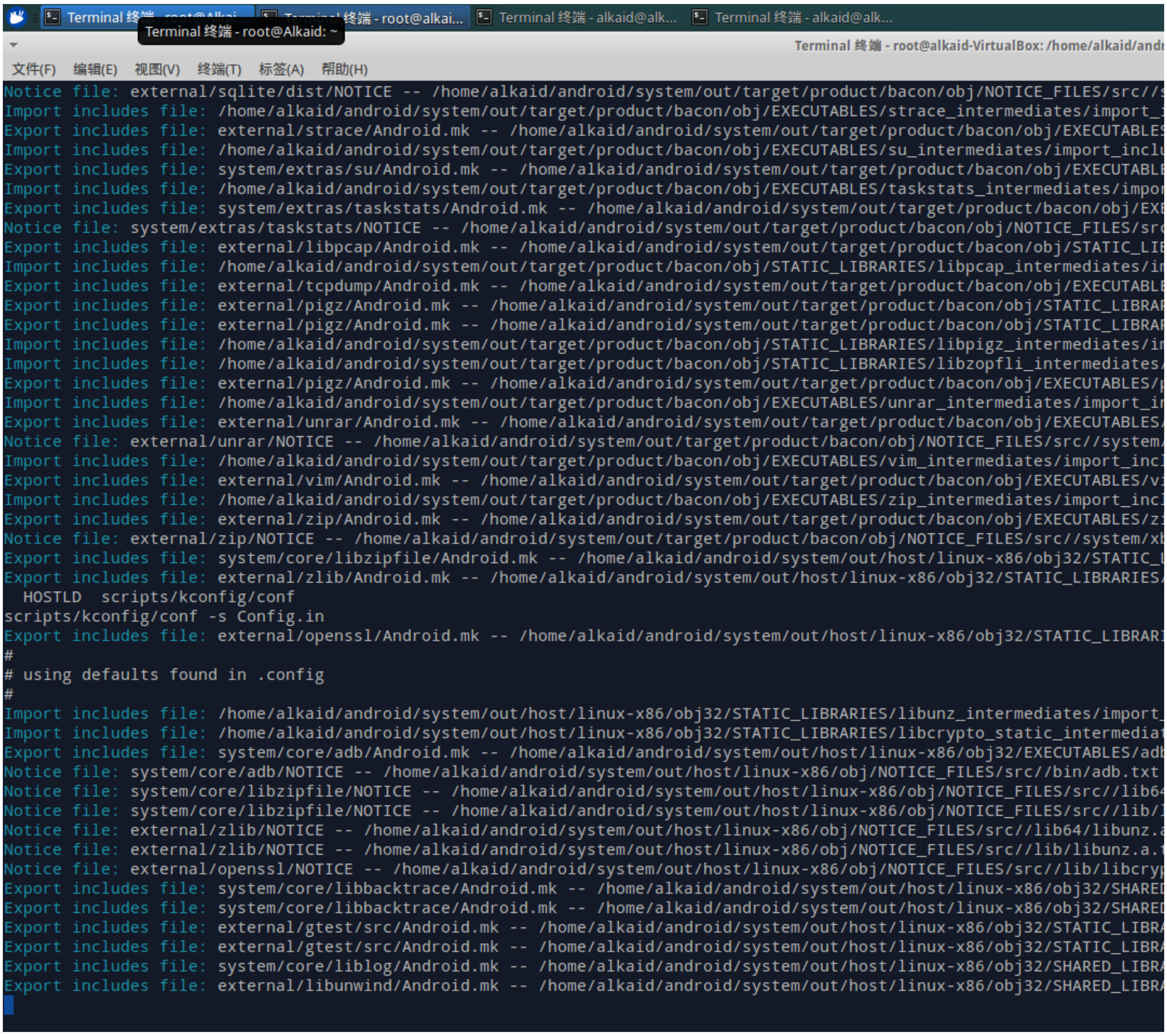
Kali Nethunter是一款用于安全研究的手机固件包，可以使Android设备增加“无线破解”、“HID攻击”、“伪造光驱”等的硬件功能以及metasploit等软件工具，目前官方只支持少量的Android手机，然而，通过重新编译Kali Nethunter源代码，可以将其编译到其他型号的手机上

本文将以Oneplus one为例讲解Kali Nethunter代码的移植方式，同时也在三星的i9100g手机上测试成功


Terminal 终端 - root@Alkai...
Terminal 终端 - root@alkai...
Terminal 终端 - alkaid@alk...
Terminal 终端 - alkaid@alk...
Terminal 终端 - root@Alkaid: ~
Terminal 终端 - root@alkaid-VirtualBox: /home/alkaid/andr
文件(F)
编辑(E)
视图(V)
终端(T)
标签(A)
帮助(H)

```
Notice file: external/sqlite/dist/NOTICE -- /home/alkaid/android/system/out/target/product/bacon/obj/NOTICE_FILES/src//
Import includes file: /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES/strace_intermediates/import_
Export includes file: external/strace/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLE
Import includes file: /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES/su_intermediates/import_incl
Export includes file: system/extras/su/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABL
Import includes file: /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES/taskstats_intermediates/impo
Export includes file: system/extras/taskstats/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/EX
Notice file: system/extras/taskstats/NOTICE -- /home/alkaid/android/system/out/target/product/bacon/obj/NOTICE_FILES/sr
Export includes file: external/libpcap/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/STATIC_LI
Import includes file: /home/alkaid/android/system/out/target/product/bacon/obj/STATIC_LIBRARIES/libpcap_intermediates/i
Export includes file: external/tcpdump/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABL
Export includes file: external/pigz/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/STATIC_LIBRA
Export includes file: external/pigz/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/STATIC_LIBRA
Import includes file: /home/alkaid/android/system/out/target/product/bacon/obj/STATIC_LIBRARIES/libpigz_intermediates/i
Import includes file: /home/alkaid/android/system/out/target/product/bacon/obj/STATIC_LIBRARIES/libzopfli_intermediates
Export includes file: external/pigz/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES/
Import includes file: /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES/unrar_intermediates/import_i
Export includes file: external/unrar/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES
Notice file: external/unrar/NOTICE -- /home/alkaid/android/system/out/target/product/bacon/obj/NOTICE_FILES/src//system
Import includes file: /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES/vim_intermediates/import_incl
Export includes file: external/vim/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES/v
Import includes file: /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES/zip_intermediates/import_incl
Export includes file: external/zip/Android.mk -- /home/alkaid/android/system/out/target/product/bacon/obj/EXECUTABLES/z
Notice file: external/zip/NOTICE -- /home/alkaid/android/system/out/target/product/bacon/obj/NOTICE_FILES/src//system/x
Export includes file: system/core/libzipfile/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/STATIC_
Export includes file: external/zlib/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/STATIC_LIBRARIES
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf -s Config.in
Export includes file: external/openssl/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/STATIC_LIBRAR
#
# using defaults found in .config
#
Import includes file: /home/alkaid/android/system/out/host/linux-x86/obj32/STATIC_LIBRARIES/libunz_intermediates/import_
Import includes file: /home/alkaid/android/system/out/host/linux-x86/obj32/STATIC_LIBRARIES/libcrypto_static_intermedia
Export includes file: system/core/adb/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/EXECUTABLES/ad
Notice file: system/core/adb/NOTICE -- /home/alkaid/android/system/out/host/linux-x86/obj/NOTICE_FILES/src//bin/adb.txt
Notice file: system/core/libzipfile/NOTICE -- /home/alkaid/android/system/out/host/linux-x86/obj/NOTICE_FILES/src//lib6
Notice file: system/core/libzipfile/NOTICE -- /home/alkaid/android/system/out/host/linux-x86/obj/NOTICE_FILES/src//lib/
Notice file: external/zlib/NOTICE -- /home/alkaid/android/system/out/host/linux-x86/obj/NOTICE_FILES/src//lib64/libunz.
Notice file: external/zlib/NOTICE -- /home/alkaid/android/system/out/host/linux-x86/obj/NOTICE_FILES/src//lib/libunz.a.
Notice file: external/openssl/NOTICE -- /home/alkaid/android/system/out/host/linux-x86/obj/NOTICE_FILES/src//lib/libcry
Export includes file: system/core/libbacktrace/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/SHARE
Export includes file: system/core/libbacktrace/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/SHARE
Export includes file: external/gtest/src/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/STATIC_LIBR
Export includes file: external/gtest/src/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/STATIC_LIBR
Export includes file: system/core/liblog/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/SHARED_LIBR
Export includes file: external/libunwind/Android.mk -- /home/alkaid/android/system/out/host/linux-x86/obj32/SHARED_LIBR
```

0x00 Kali Nethunter架构分析

Kali Nethunter大体分为三个部分

1. 定制过的手机内核：由于需要使用OTG外接usb网卡、用手机模拟HID设备、用手机模拟CDROM光驱，必须对手机内核进行修改添加对应的驱动，以及patch相关的代码，如果没有这一部分修改过的内核，Kali Nethunter将无法使用与硬件相关的所有功能
2. chroot环境的文件系统：所有相关的软件程序以ARM架构的Kali为基础集成在一个文件系统内，通过linux的chroot功能可以在Android中跳转至Kali文件系统，然后把内核相关的东西mount进kali的文件系统，之后就可以在这个chroot后的Kali中执行对应的命令了
3. 用于提供界面的APK手机APP，仅仅是作为UI界面起展示作用，实际是通过调用chroot的Kali中的命令来实现所有功能的，当然这些APP也提供了一键挂载并启用Kali chroot的功能

其中修改内核要求手机内核代码必须开源，尽管内核代码根据GPL协议必须开源，国内某些手机仍然不公开其内核源代码，这也是Kali Nethunter仅支持Oneplus、三星和谷歌Nexus的原因

0x01 准备CM 12.1源码以及相关环境

首先，我们为了简化修改内核的过程，可以直接编译一份CyanogenMod 12.1的代码，其中就包括了对应机型（Oneplus one、三星i9100g）的内核代码。而且鉴于Kali Nethunter本身采用的就是CM 12.1的ROM，为了避免其他ROM中可能存在的兼容性问题，建议选择CyanogenMod支持的手机来进行移植

你可以参考CyanogenMod官方wiki上对应机型的编译说明，这里在Ubuntu上以Oneplus和i9100g为例演示编译过程

亲测编译时硬盘至少需要100G可用空间，虚拟机至少需要4GB内存，不要问我如果达不到要求会发生什么……全是眼泪啊

在开始配置环境之前，先提示大家后文会有简化的环境配置方式，不过建议大家还是按部就班地用“官方”方式配置，因为将来CM可能会有新的版本出现，而官方配置方式可以获取到最新的代码

比如对于Oneplus one的官方示范：http://wiki.cyanogenmod.org/w/Build_for_bacon

```
sudo apt-get install bison build-essential curl flex git gnupg gperf libesd0-dev liblz4-tool libncurses5-dev libsdl1.2-dev libwxgtk2.8-dev libxml2 libxml2
```

然后64位系统还需要：

```
sudo apt-get install g++-multilib gcc-multilib lib32ncurses5-dev lib32readline-gplv2-dev lib32z1-dev
```

如果遇到软件包不存在，可以apt-cache search一下包名，看看是不是改名字了

之后同步Android代码（代码从google上拿，该怎么访问google是你自己要解决的问题，下同，不再提示）

```
$ mkdir -p ~/bin
$ mkdir -p ~/android/system
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

然后把~/bin加进PATH中：

添加到~/.profile中

```
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

然后配置repo

```
$ cd ~/android/system/
$ repo init -u https://github.com/CyanogenMod/android.git -b cm-12.1
```

然后有两种方式可以获取特定设备（Oneplus One）的相关代码文件，一种是按照CyanogenMod Wiki上的做，另一种可以去百度（来自一加社区的《玩机组出品，CyanogenMod12编译教程》），请自行百度自己设备的代码获取方式

```
mkdir ~/android/cm/.repo/local_manifests
gedit ~/android/cm/.repo/local_manifests/local_manifests.xml
```

添加

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
<project name="CyanogenMod/android_device_oneplus_bacon" path="device/oneplus/bacon" remote="github" />
<project name="CyanogenMod/android_device_qcom_common" path="device/qcom/common" remote="github" />
<project name="CyanogenMod/android_device_oppo_msm8974-common" path="device/oppo/msm8974-common" remote="github" />
<project name="CyanogenMod/android_device_oppo_common" path="device/oppo/common" remote="github" revision="cm-12.0" />
<project name="CyanogenMod/android_kernel_oneplus_msm8974" path="kernel/oneplus/msm8974" remote="github" />
<project name="TheMuppets/proprietary_vendor_oppo" path="vendor/oppo" remote="github" />
<project name="TheMuppets/proprietary_vendor_oneplus" path="vendor/oneplus" remote="github" />
<project name="CyanogenMod/android_frameworks_opt_connectivity" path="frameworks/opt/connectivity" remote="github" revision="cm-11.0" />
</manifest>
```

之后就可以开始同步Android代码了

```
$ repo sync
```

注意，到这步之后不需要再get prebuilt了，因为我们用的是CM 12.1，老版本的CM才需要get prebuilt

之后就是编译了（先不要修改内核，确保能够编译成功），进入android/system文件夹

```
source build/envsetup.sh
```

使用CCACHE可以加快编译速度，但会吃掉硬盘空间，为了速度推荐设置为50G到100G之间，非必须

```
export USE_CCACHE=1
prebuilts/misc/linux-x86/ccache/ccache -M 50G
```

之后就可以开始编译了，bacon是设备名

```
$ croot
$ brunch bacon
```

编译成功后将会生成cm-12.1-20151127-UNOFFICIAL-bacon.zip，这个就是ROM的刷机包了

0x02 简易的CM 12.1编译环境及代码

所有环境配置问题都可以用Woobuntu系统来解决，直接安装Woobuntu后内部已经集成了Android编译环境（包括adb）

代码我给大家打包好了，27个G，解压缩就好（已经把内核patch过了），在Woobuntu中编译命令如下：

```
tar -jxvf cm12_bacon_source.tar.bz2
cd android/system/
sudo su root
source build/envsetup.sh
export USE_CCACHE=1
prebuilts/misc/linux-x86/ccache/ccache -M 20G
```

```
croot
brunch bacon
```

简单解释一下，由于压缩档的文件属主是我，而不是你，部分文件权限会有问题，所以建议直接root权限编译（其实更优雅的方式是chown -R，只不过27个G代码都来chown一遍实在受不了），sudo时会要求输入你的密码，而我CCACHE只用了20G是因为我的硬盘实在太小，我最后亲自从安装Woobuntu开始测试了一遍，确认可以编译成功。

简而言之，这个简化版本只需要你自己处理设备相关代码就好了，比如说i9100g就只需要重新按照wiki设置breakfast i9100g和extract-files（如果网上能搜到更简单的方式更好），之后brunch i9100g即可

0x03 修改Android内核

在修改Android内核时，我们需要交叉编译ARM架构的代码

cd到对应机型的kernel文件夹下，例如演示的oneplus内核目录为 alkaid@alkaid-VirtualBox:~/android/system/kernel/oneplus/msm8974

```
export ARCH=arm
```

之后在arch/arm/configs文件夹里面找到cm所用的defconfig，然后如下所示

```
make cyanogenmod_bacon_defconfig
```

之后就可以make menuconfig了

```
make menuconfig
```

然后就是根据Nethunter的github WIKI上的说明自由选择驱动程序了，不过我只需要ATH9K的芯片驱动，所以直接在Device drivers -> Network device supports -> Wireless lan 里面选中Atheros Wireless Cards ，如果你使用其他芯片的无线网卡，请自己去选中对应的驱动程序

然后在networking support -> Wireless 里面把Generic IEEE 802.11 Networking Stack (mac80211) 选中

由于Oneplus自己默认选中了OTG驱动，如果您为别的机型编译，请检查device driver里面usb的otg选项有没有选中

设置完毕后保存退出

```
make savedefconfig
```

```
cp defconfig arch/arm/configs/cyanogenmod_bacon_defconfig
```

```
make mrproper
```

您如果是第一次尝试，可以重新编译一遍CM看看有无错误，然后我们就可以开始patch内核了

```
wget http://patches.aircrack-ng.org/mac80211.compat08082009.wl_frag+ack_v1.patch
patch -p1 < mac80211.compat08082009.wl_frag+ack_v1.patch
```

然后到https://github.com/pelya/android-keyboard-gadget里面寻找自己内核版本的patch，这里是3.4内核

```
wget https://raw.githubusercontent.com/pelya/android-keyboard-gadget/master/kernel-3.4.patch
```

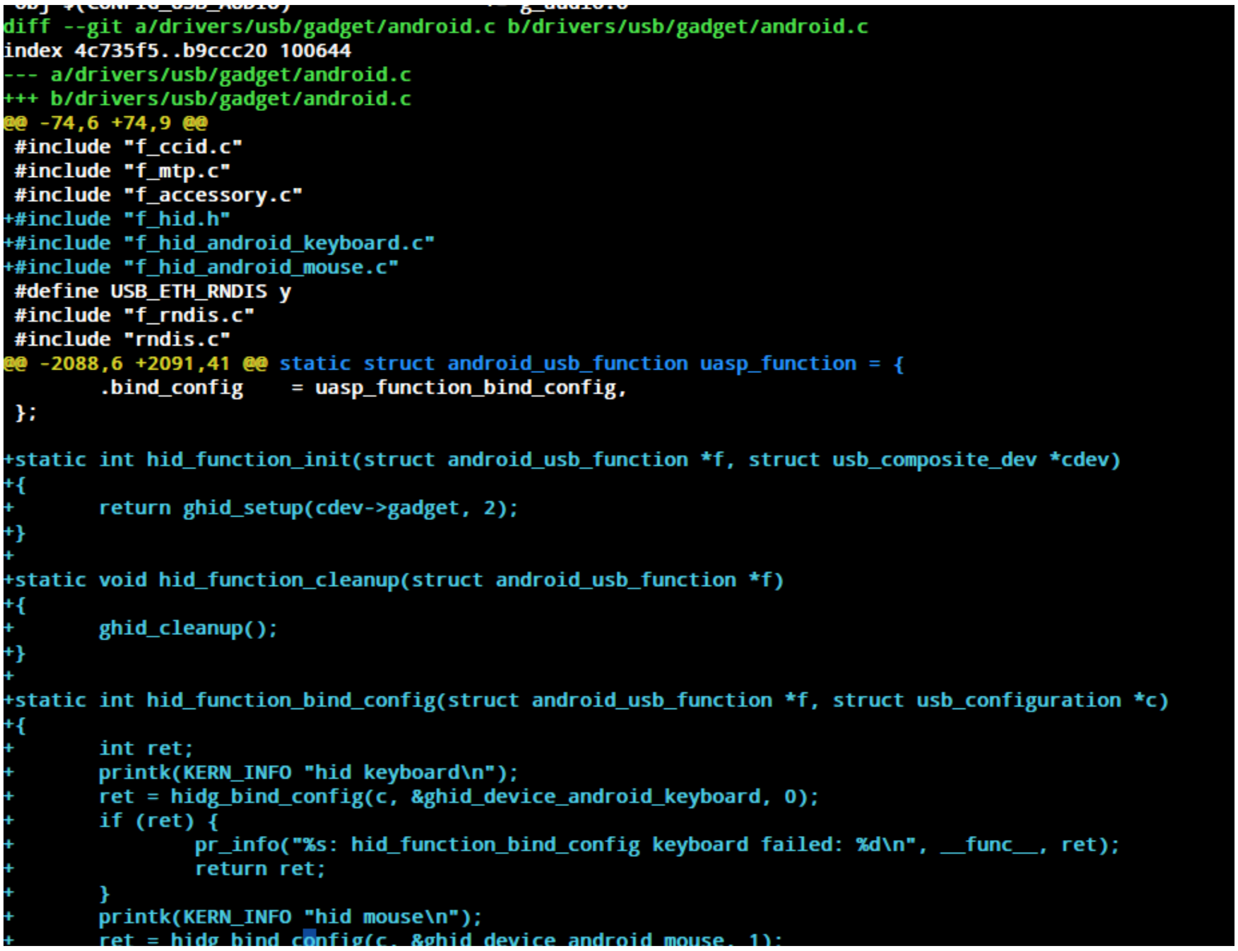
```
patch -p1 < kernel-3.4.patch
```

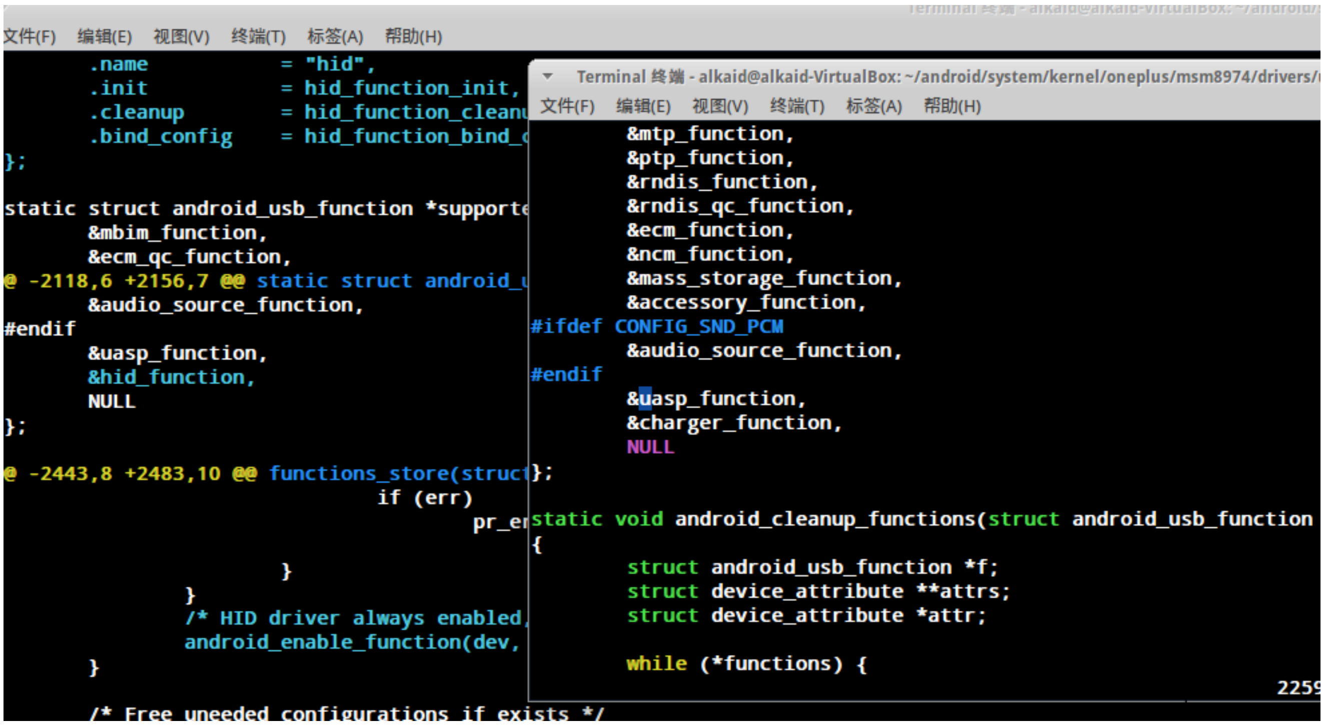
如果不出意外的话，这个patch一定会报错，因为kernel代码是不断更新着的，不过别担心，我们可以手动去patch

报错：

```
alkaid@alkaid-VirtualBox:~/android/system/kernel/oneplus/msm8974$ patch -p1 < kernel-3.4.patch
patching file drivers/usb/gadget/Makefile
patching file drivers/usb/gadget/android.c
Hunk #1 succeeded at 75 (offset 1 line).
Hunk #2 succeeded at 2192 with fuzz 2 (offset 101 lines).
Hunk #3 FAILED at 2156.
Hunk #4 FAILED at 2481.
2 out of 4 hunks FAILED -- saving rejects to file drivers/usb/gadget/android.c.rej
patching file drivers/usb/gadget/f_hid.c
Hunk #7 succeeded at 403 (offset -9 lines).
Hunk #8 succeeded at 422 (offset -9 lines).
Hunk #10 succeeded at 594 (offset -4 lines).
Hunk #11 succeeded at 614 (offset -6 lines).
Hunk #12 succeeded at 662 (offset -8 lines).
Hunk #13 succeeded at 713 (offset -8 lines).
patching file drivers/usb/gadget/f_hid.h
patching file drivers/usb/gadget/f_hid_android_keyboard.c
patching file drivers/usb/gadget/f_hid_android_mouse.c
alkaid@alkaid-VirtualBox:~/android/system/kernel/oneplus/msm8974$
```

可以看到是android.c第3和第4处patch失败，我们去手动patch


```
diff --git a/drivers/usb/gadget/android.c b/drivers/usb/gadget/android.c
index 4c735f5..b9ccc20 100644
--- a/drivers/usb/gadget/android.c
+++ b/drivers/usb/gadget/android.c
@@ -74,6 +74,9 @@
 #include "f_ccid.c"
 #include "f_mtp.c"
 #include "f_accessory.c"
+#include "f_hid.h"
+#include "f_hid_android_keyboard.c"
+#include "f_hid_android_mouse.c"
 #define USB_ETH_RNDIS y
 #include "f_rndis.c"
 #include "rndis.c"
@@ -2088,6 +2091,41 @@ static struct android_usb_function uasp_function = {
        .bind_config    = uasp_function_bind_config,
 };

+static int hid_function_init(struct android_usb_function *f, struct usb_composite_dev *cdev)
+{
+       return ghid_setup(cdev->gadget, 2);
+}
+
+static void hid_function_cleanup(struct android_usb_function *f)
+{
+       ghid_cleanup();
+}
+
+static int hid_function_bind_config(struct android_usb_function *f, struct usb_configuration *c)
+{
+       int ret;
+       printk(KERN_INFO "hid keyboard\n");
+       ret = hidg_bind_config(c, &ghid_device_android_keyboard, 0);
+       if (ret) {
+               pr_info("%s: hid_function_bind_config keyboard failed: %d\n", __func__, ret);
+               return ret;
+       }
+       printk(KERN_INFO "hid mouse\n");
+       ret = hidg_bind_config(c, &ghid_device_android_mouse, 1);
```
前两处都已经patch成功了，所以跳过就好了


文件(F) 编辑(E) 视图(V) 终端(T) 标签(A) 帮助(H)
Terminal 终端 - alkaid@alkaid-VirtualBox: ~/android/system/kernel/oneplus/msm8974/drivers/
文件(F) 编辑(E) 视图(V) 终端(T) 标签(A) 帮助(H)
```
       .name           = "hid",
       .init           = hid_function_init,
       .cleanup        = hid_function_cleanu
       .bind_config    = hid_function_bind_c
};

static struct android_usb_function *supporte
       &mbim_function,
       &ecm_qc_function,
@ -2118,6 +2156,7 @@ static struct android_u
       &audio_source_function,
#endif
       &uasp_function,
       &hid_function,
       NULL
};

@ -2443,8 +2483,10 @@ functions_store(struct
                       if (err)
                               pr_er

                       }
               }
               /* HID driver always enabled,
               android_enable_function(dev,
       }

       /* Free uneeded configurations if exists */
```
```
        &mtp_function,
        &ptp_function,
        &rndis_function,
        &rndis_qc_function,
        &ecm_function,
        &ncm_function,
        &mass_storage_function,
        &accessory_function,
#ifdef CONFIG_SND_PCM
        &audio_source_function,
#endif
        &uasp_function,
        &charger_function,
        NULL
};

static void android_cleanup_functions(struct android_usb_function
{
        struct android_usb_function *f;
        struct device_attribute **attrs;
        struct device_attribute *attr;

        while (*functions) {
                                                              2259
```
解释一下：左边是patch代码，右边是目前的android.c，patch的意思是要在&uasp_function,下面加一个&hid_function,，第四处也去找一下用大脑找到位置敲上代码即可

然后就是重新编译CM，然后生成的刷机包就是Nethunter超强定制内核的CM 12.1了

i9100g编译时直接成功，而Oneplus则会报个错，是跟usb驱动有关的，找到报错位置代码，然后发现是一个结构体还有一个函数在使用前没有定义，于是我无比潇洒地把这个调用函数的代码！删！掉！了！，经测试删掉该代码不影响手机正常功能，Nethunter功能也经过测试非常稳定

0x04 编译Kali Nethunter的rootfs

在Kali Nethunter的github页面内有详细的说明，强烈建议在Kali中进行Nethunter的编译

```
mkdir ~/arm-stuff
cd ~/arm-stuff
```

```
git clone https://github.com/offensive-security/gcc-arm-linux-gnueabihf-4.7
export PATH=${PATH}:/root/arm-stuff/gcc-arm-linux-gnueabihf-4.7/bin
git clone https://github.com/offensive-security/kali-nethunter
cd ~/arm-stuff/kali-nethunter
./build-deps.sh
./androidmenu.sh
```

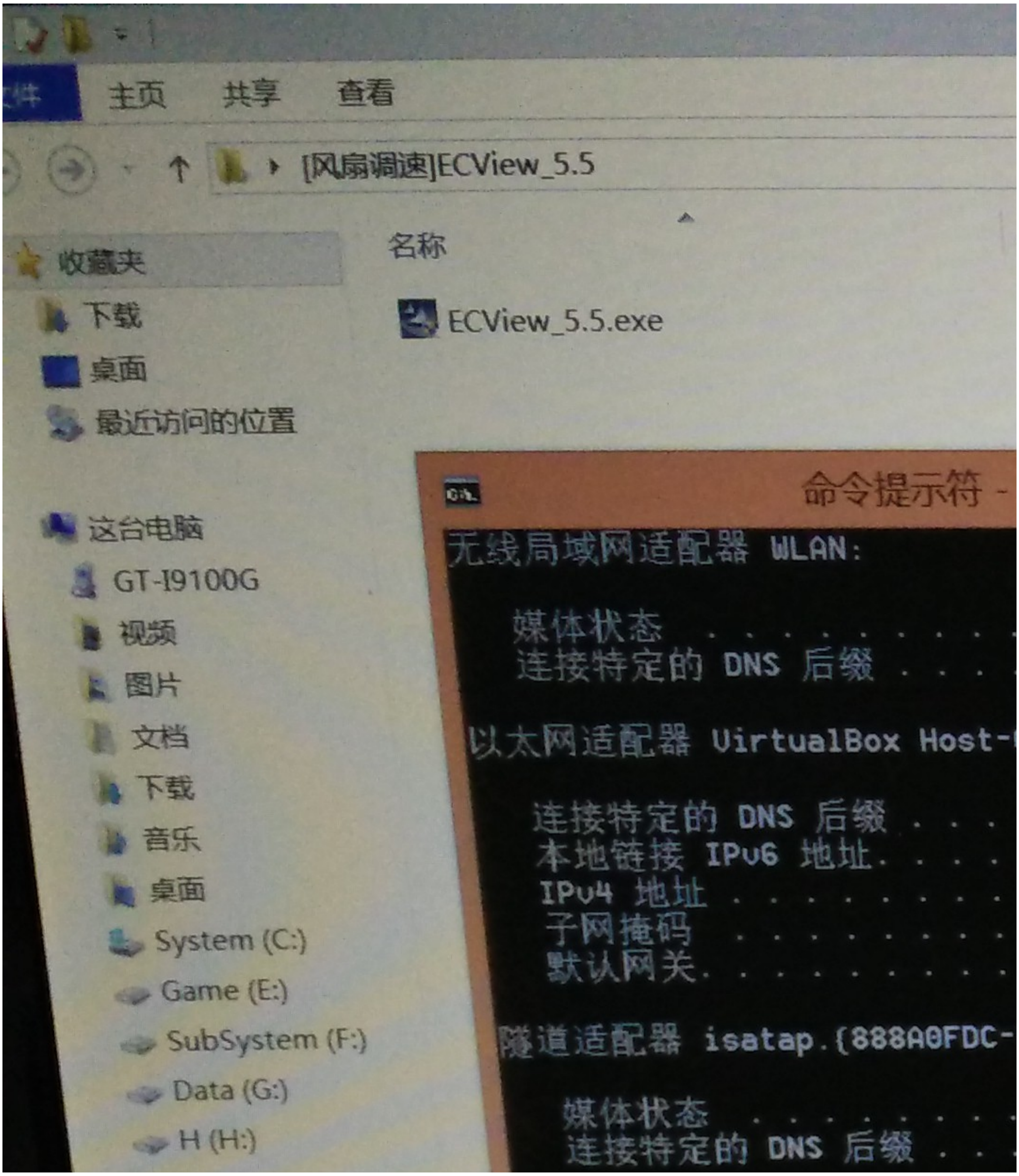
在编译时选择仅编译rootfs（因为我们要自己操心内核的事情了）

0x05 刷机测试

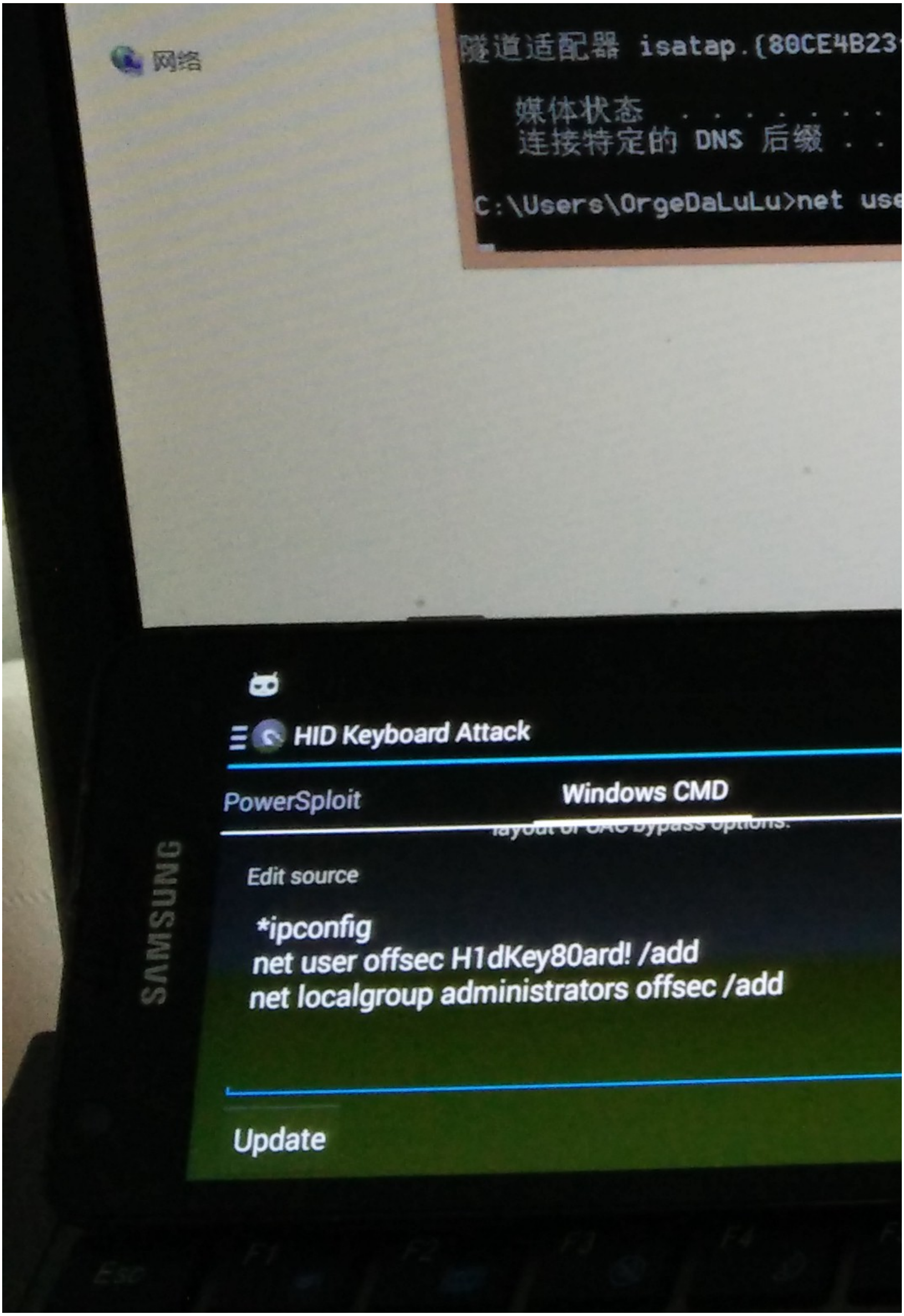
如果你不会刷机，请先百度一下如何正常刷机，比如unlock bootloader啦，TWRP啦，这些百度去吧

刷机顺序为：先刷CyanogenMod，再刷Kali Nethunter的rootfs

下图是在测试HID攻击的i9100g

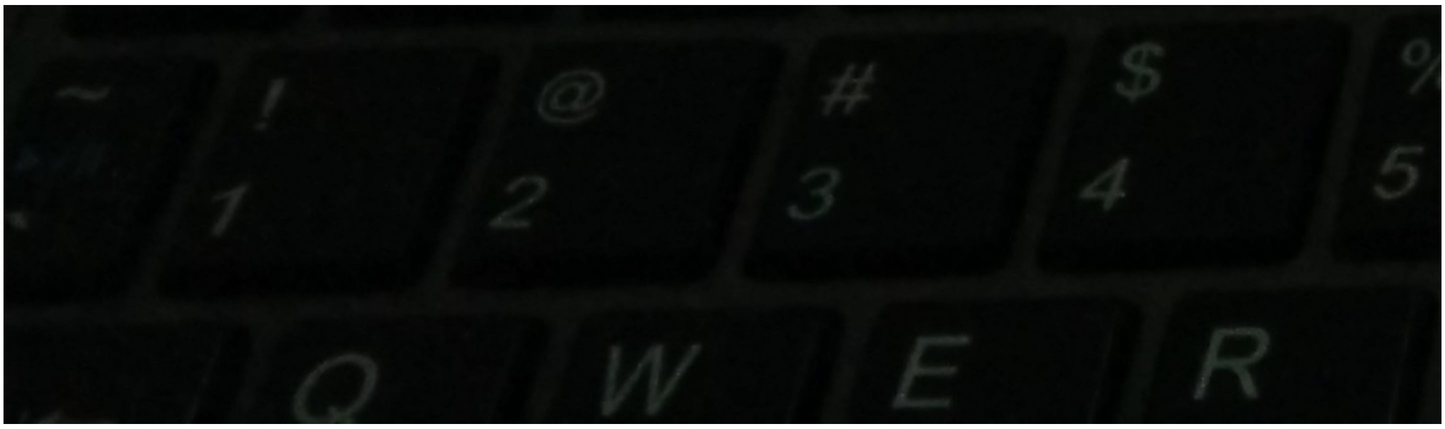


```
无线局域网适配器 WLAN:

   媒体状态  . . . . . . . . . . . .
   连接特定的 DNS 后缀 . . . .

以太网适配器 VirtualBox Host-

   连接特定的 DNS 后缀 . . . .
   本地链接 IPv6 地址. . . . .
   IPv4 地址 . . . . . . . . . . .
   子网掩码  . . . . . . . . . . .
   默认网关. . . . . . . . . . . .

隧道适配器 isatap.{888A0FDC-

   媒体状态  . . . . . . . . . . .
   连接特定的 DNS 后缀 . . .
```



```
隧道适配器 isatap.{80CE4B23

   媒体状态 . . . . . . .
   连接特定的 DNS 后缀 . .

C:\Users\OrgeDaLuLu>net use
```

```
*ipconfig
net user offsec H1dKey80ard! /add
net localgroup administrators offsec /add
```



检查各项功能：

1. 检查DriveDroid能否模拟USB光驱
2. 检查UsbKeyboard能否伪造鼠标和键盘设备
3. 插上OTG和USB无线网卡，看看能不能使用Wifite破解无线密码

0x06 可能遇到的问题

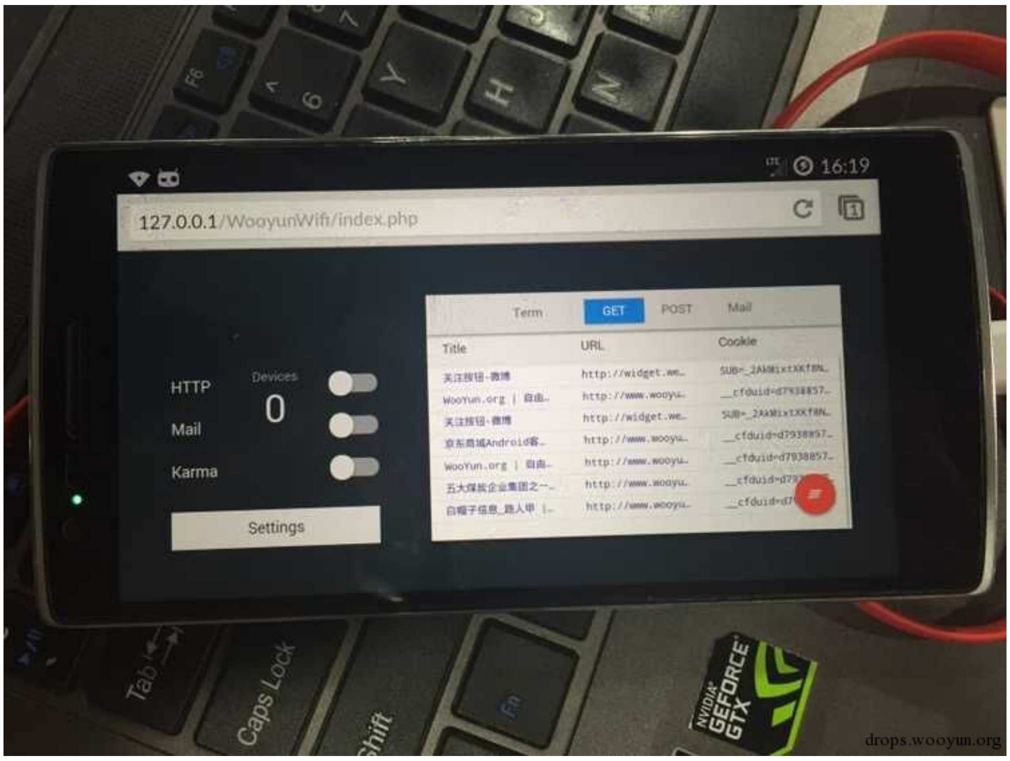
在移植至三星i9100g手机时，由于手机内存储空间不够大（Kali Nethunter刷机方式要求至少有2GB的可用空间），因此一直刷机失败，当终于定位到问题所在后，最终决定采用其他的方式加载Kali Nethunter的rootfs。首先把kali nethunter的rootfs编译出来，然后做成一个img磁盘镜像文件，之后把这个img文件放在sd卡中，动态挂载到/data/local/kali-armhf中，这样就解决了内存储空间不够的问题

如果编译内核出错，您在debug的时候可以只编译内核，toolchains在prebuilt文件夹里面，自己设置cross_compile变量吧（常见是把乱选的驱动清除掉，能解决绝大部分报错）

0x07 后话

以后我们就可以开始玩耍Nethunter了，比如把整个WooyunWifi都移植到手机上，再做上离线劫持钓鱼的功能……不过这些都是编译完Nethunter之后的事情了

结尾图片：在手机上的WooyunWifi，开启了热点用于钓鱼



Title	URL	Cookie
关注按钮-微博	http://widget.we…	SUB=_2AkMixtXKf8N…
WooYun.org | 自由…	http://www.wooyu…	__cfduid=d7938857…
关注按钮-微博	http://widget.we…	SUB=_2AkMixtXKf8N…
京东商城Android客…	http://www.wooyu…	__cfduid=d7938857…
WooYun.org | 自由…	http://www.wooyu…	__cfduid=d7938857…
五大煤炭企业集团之一…	http://www.wooyu…	__cfduid=d79
白帽子信息_路人甲 |…	http://www.wooyu…	__cfduid=d7