

# 反反调试修改 tracerPid

参见: <https://www.cnblogs.com/jiaoxiake/p/6790803.html>

## 1. 反反调试原理&&实现

原理:

根据/proc/pid/status 文件中的 TracerPid 字段判断是否应用被调试: 当值为 0 时, 非调试转台, 值不为 0 时, 则是在调试。

实现:

- (1) 内核源码 kernel 修改代码
- (2) 提取内核 boot.img 并修改

## 2. 实现一: 修改内核源码

修改源码方式肯定能成功, 由于网络问题, 为下载 kernel 代码, 该方式没尝试, 记录一下, 详情参见

<https://www.jianshu.com/p/26c16c747720>

修改文件:

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

base.c 在 line285 处修改如下:

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

array.c 在 line134 处修改如下:

```
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 */
};
```

在 line187 处修改如下：

```
"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);
```

修改后，重新编译 kernel，替换 zImage-dtb,重新编译 AOSP，刷机即可。

### 3. 方式二：提取 zImage 内核文件

#### (1) 提取 boot.img

使用 Nexus 5 手机获取 root 权限：刷 super-su

root 权限下输入命令：

```
ls -l /dev/block/platform/msm_sdcc.1/by-name/ (如图 1)
```

将 boot 导出为 boot.img

```
dd if=/dev/block/mmcblk0p19 of=/data/local/tmp/boot.img
```

```
adb pull /data/local/tmp/boot.img boot.img
```

```

lrwxrwxrwx root root 1970-01-05 01:37 DDR -> /dev/block/mmcblk0p24
lrwxrwxrwx root root 1970-01-05 01:37 aboot -> /dev/block/mmcblk0p6
lrwxrwxrwx root root 1970-01-05 01:37 abootb -> /dev/block/mmcblk0p11
lrwxrwxrwx root root 1970-01-05 01:37 boot -> /dev/block/mmcblk0p19
lrwxrwxrwx root root 1970-01-05 01:37 cache -> /dev/block/mmcblk0p27
lrwxrwxrwx root root 1970-01-05 01:37 crypto -> /dev/block/mmcblk0p26
lrwxrwxrwx root root 1970-01-05 01:37 fsc -> /dev/block/mmcblk0p22
lrwxrwxrwx root root 1970-01-05 01:37 fsg -> /dev/block/mmcblk0p21
lrwxrwxrwx root root 1970-01-05 01:37 grow -> /dev/block/mmcblk0p29
lrwxrwxrwx root root 1970-01-05 01:37 imgdata -> /dev/block/mmcblk0p17
lrwxrwxrwx root root 1970-01-05 01:37 laf -> /dev/block/mmcblk0p18
lrwxrwxrwx root root 1970-01-05 01:37 metadata -> /dev/block/mmcblk0p14
lrwxrwxrwx root root 1970-01-05 01:37 misc -> /dev/block/mmcblk0p15
lrwxrwxrwx root root 1970-01-05 01:37 modem -> /dev/block/mmcblk0p1
lrwxrwxrwx root root 1970-01-05 01:37 modemst1 -> /dev/block/mmcblk0p12
lrwxrwxrwx root root 1970-01-05 01:37 modemst2 -> /dev/block/mmcblk0p13
lrwxrwxrwx root root 1970-01-05 01:37 pad -> /dev/block/mmcblk0p7
lrwxrwxrwx root root 1970-01-05 01:37 persist -> /dev/block/mmcblk0p16
lrwxrwxrwx root root 1970-01-05 01:37 recovery -> /dev/block/mmcblk0p20
lrwxrwxrwx root root 1970-01-05 01:37 rpm -> /dev/block/mmcblk0p3
lrwxrwxrwx root root 1970-01-05 01:37 rpmb -> /dev/block/mmcblk0p10
lrwxrwxrwx root root 1970-01-05 01:37 sb11 -> /dev/block/mmcblk0p2
lrwxrwxrwx root root 1970-01-05 01:37 sb11b -> /dev/block/mmcblk0p8
lrwxrwxrwx root root 1970-01-05 01:37 sdi -> /dev/block/mmcblk0p5
lrwxrwxrwx root root 1970-01-05 01:37 ssd -> /dev/block/mmcblk0p23
lrwxrwxrwx root root 1970-01-05 01:37 system -> /dev/block/mmcblk0p25
lrwxrwxrwx root root 1970-01-05 01:37 tz -> /dev/block/mmcblk0p4
lrwxrwxrwx root root 1970-01-05 01:37 tzb -> /dev/block/mmcblk0p9
lrwxrwxrwx root root 1970-01-05 01:37 userdata -> /dev/block/mmcblk0p28

```

图 1

使用 bootimg.exe 解开 boot.img : ( .exe 和 .img 同一个目录下)

bootimg.exe --unpack-bootimg

```

.\bootimg.exe --unpack-bootimg
arguments: [bootimg file]
bootimg file: boot.img
output: kernel[.gz] ramdisk[.gz] second[.gz]
found nonstandard ramdisk_addr
found nonstandard tags_addr
base: 0x0
ramdisk_addr: 0x2900000
second_addr: 0xf00000
tags_addr: 0x2700000
page_size: 2048
name: ""
cmdline: "console=ttyHSL0,115200,n8 androidthboot.hardware=hammerhead user_debug=31 maxcpus=2 msm_watchdog_v2.enable=1"
padding_size=2048
arguments: [ramdisk file] [directory]
ramdisk file: ramdisk.gz
directory: initrd
output: cpio1ist.txt
compress: True

```

## (2) 提取 kernel

将 kernel 备份, 将 kernel(其他文章可能说 zImage), kernel 中含有 .gz 文件, 使用 010editor 提取出 .gz 文件,

010Editor 中搜索 1F 8B 08 00,将之前的 Hex 数据删除掉并保存, 修改 kernel 为 k.gz, 使用解压缩 k.gz,解压出的 k 为 nexus 手机的 kernel.

### (3) 修改 kernel 指令

ida 加载 k 文件（如图 2），ARM Little-endian，在 ROM start address 和 Loading address 填 `0xc0008000` (如图 3)，点【ok】

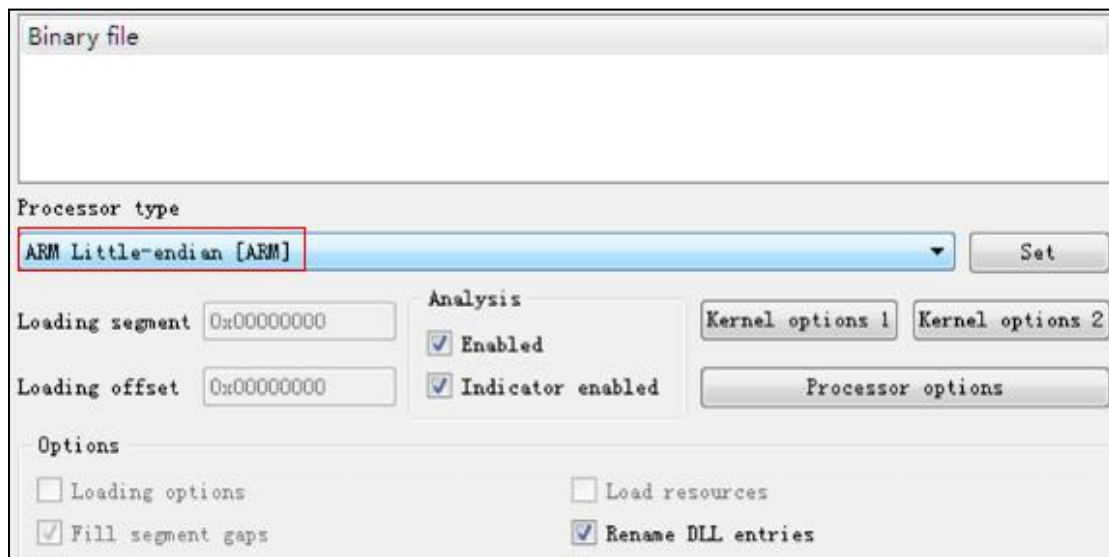


图 2

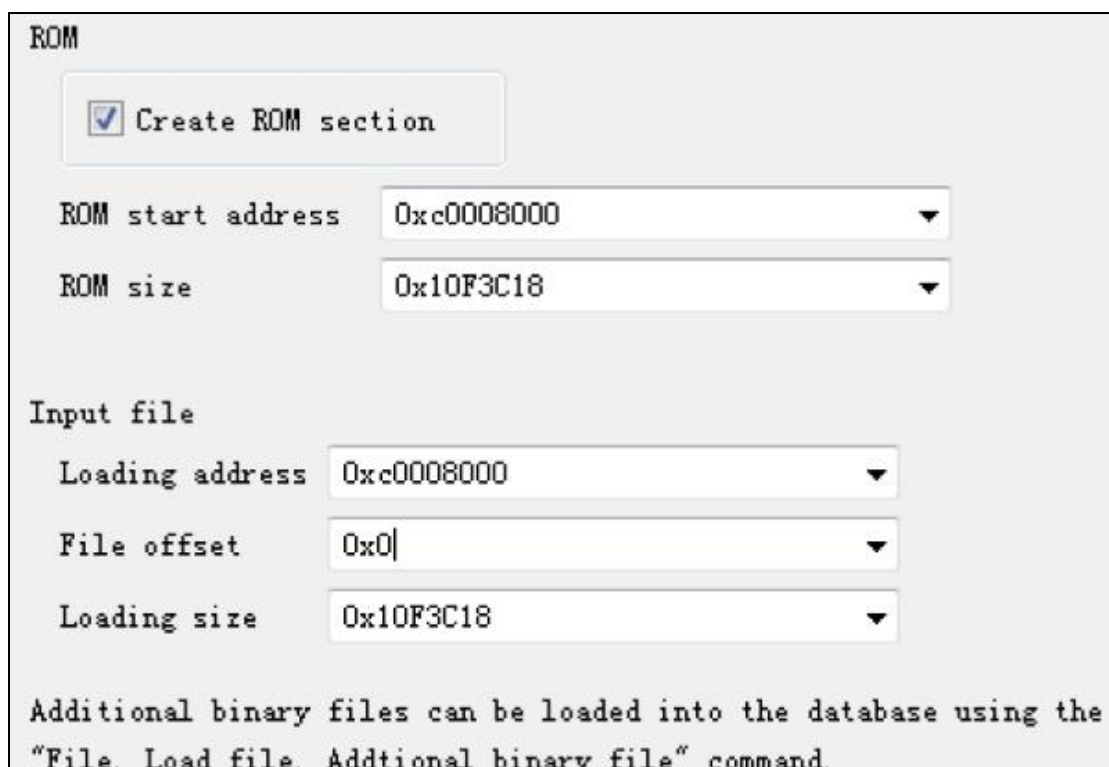


图 3

使用 root 权限在 adb shell 中，关闭符号屏蔽。输入命令：

```
echo 0 > /proc/sys/kernel/kptr_restrict
```

获取两个函数的地址：proc\_pid\_status

\_\_task\_pid\_nr\_ns

```
cat /proc/kallsyms |grep proc_pid_status
```

--> c02ba04c T proc\_pid\_status (c02ba04c 值可能不一样)

```
cat /proc/kallsyms |grep __task_pid_nr_ns
```

--> c01b0884 T \_\_task\_pid\_nr\_ns (c01b0884 值可能不一样)

Ida: 按 g ,goto address : c02ba04c ,按 c(decode as asm),  
其中该处的函数调用，aStateSTgidPid,

```
C02BA1CC      STR      R3, [R11,#var_7C]
C02BA1D0      BL       sub_C01B07CC
C02BA1D4      STMFA   SP, {R9,R10}
C02BA1D8      LDR      R2, [R11,#var_78]
C02BA1DC      LDR      R1, =aStateSTgidDPid ; "State:\t%s\nTgid:\t%d\nPid:\t%d\nPPid:"...
C02BA1E0      LDR      R3, [R11,#var_7C]
C02BA1E4      STR      R0, [SP,#0xB0+var_B0]
C02BA1E8      MOV      R0, R4
C02BA1EC      LDR      R12, [R6,#4]
C02BA1F0      STR      R12, [SP,#0xB0+var_A4]
C02BA1F4      LDR      R12, [R6,#0x14]
```

```
4      DCB      0
4      aStateSTgidDPid DCB "State:",9,"%s",0xA ; DATA XREF: sub_C02BA04C+190fo
4                                          ; ROM:off_C02BA5E0fo
4      DCB      "Tgid:",9,"%d",0xA
4      DCB      "Pid:",9,"%d",0xA
4      DCB      "PPid:",9,"%d",0xA
4      DCB      "TracerPid:",9,"%d",0xA
4      DCB      "Uid:",9,"%d",9,"%d",9,"%d",9,"%d",0xA
4      DCB      "Gid:",9,"%d",9,"%d",9,"%d",9,"%d",0xA,0
9      DCB      0
A      DCB      0
B      DCB      0
```

,在该函数中找到 地址 c01b0884 的 sub\_C01B0884 函数(即  
函数\_\_task\_pid\_nr\_ns)

```

ROM:C02BA5A8 00 20 A0 E1      MOV      R2, R0
ROM:C02BA5AC 8A FF FF EA      B        loc_C02BA3DC
ROM:C02BA5B0      ; -----
ROM:C02BA5B0      loc_C02BA5B0      ; CODE XREF: sub_C02BA04C+50C↑j
ROM:C02BA5B0 A1 61 FB EB      BL        sub_C0192C3C
ROM:C02BA5B4      loc_C02BA5B4      ; CODE XREF: sub_C02BA04C+128↑j
ROM:C02BA5B4 34 02 95 E5      LDR      R0, [R5,#0x234]
ROM:C02BA5B8 00 00 50 E3      CMP      R0, #0
ROM:C02BA5BC 00 A0 A0 01      MOVEQ    R10, R0
ROM:C02BA5C0 EC FE FF 0A      BEQ      loc_C02BA178
ROM:C02BA5C4 00 10 A0 E3      MOV      R1, #0
ROM:C02BA5C8 07 20 A0 E1      MOV      R2, R7
ROM:C02BA5CC AC D8 FB EB      BL        sub_C01B0884 __task_pid_nr_ns
ROM:C02BA5D0 00 A0 A0 E1      MOV      R10, R0
ROM:C02BA5D4 E7 FE FF EA      B        loc_C02BA178
ROM:C02BA5D4      ; End of function sub_C02BA04C
ROM:C02BA5D4      ; -----
ROM:C02BA5D8 9C 28 D2 C0 off_C02BA5D8 DCD aName_1      ; DATA XREF: sub_C02BA04C+58↑r
ROM:C02BA5D8      ; "Name:\t"

```

两处修改指令:

MOVEQ R10, R0      替换为 MOV R10, #0      机器码为 00 A0 A0 E3

BL sub\_C01B083C    替换为 MOV R0, #0      机器码为 00 00 A0 E3

ida 保存修改指令后的 k 文件。

附件:

k 为修改后的 kernel



#### (4) 重新生成 kernel

将 k 文件使用 gzip 命令压缩, `gzip -n -f -9 k -> k.gz`

将最初的备份的 kernel.bak, 复制出一份来, 010Editor 中打开 kernel.bak, 将新的 k.gz 文件放入到 kernel.bak 中。

方法: k.gz 文件的字节数 Count 相比于原.gz 文件小, 因此在 kernel.bak 中找到 1F 8B 08 00 位置, 并删除 Count 个字节, 并将 k.gz 的 hex 形式复制到删除位置, 以保证新生成的 kernel 大小

不变（否则手机变砖，还得刷 boot.img）。

## (5) 重新打包 boot.img 并刷机

bootimg.exe --repack-bootimg

效果如下：

调试时：TracerPid: 6652

```
shell@hammerhead:/ $ ps|grep mail
u0_a25      1695  177    882584 38504 ffffffff 00000000 t com.android.email
shell@hammerhead:/ $ cat /proc/1695/status
Name:      com.android.email
State:     t (tracing stop)
Tgid:      1695
Pid:       1695
PPid:      177
TracerPid: 6652
Uid:       10025 10025 10025 10025
Gid:       10025 10025 10025 10025
FDSize:    256
Groups:    1015 1028 3003 50025
VmPeak:    884636 kB
VmSize:    882584 kB
VmLck:     0 kB
VmPin:     0 kB
```

调试时：TracerPid: 0

```
shell@hammerhead:/ $ ps|grep mail
u0_a25      1736  182    881544 38612 ffffffff 00000000 t com.android.email
shell@hammerhead:/ $ cat /proc/1736/status
Name:      com.android.email
State:     t (tracing stop)
Tgid:      1736
Pid:       1736
PPid:      182
TracerPid: 0
Uid:       10025 10025 10025 10025
Gid:       10025 10025 10025 10025
FDSize:    256
```