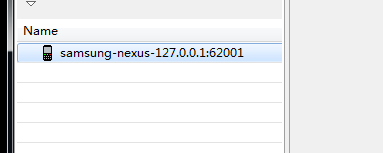
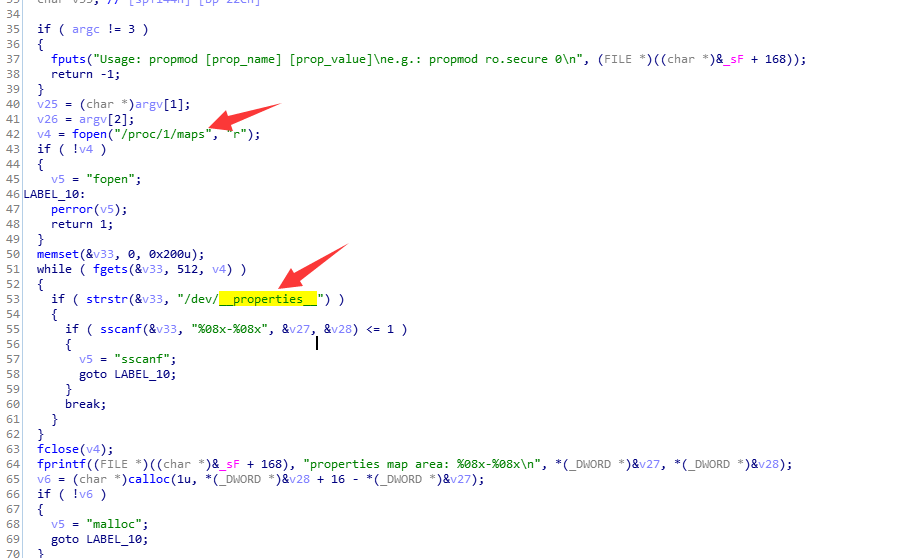
android ro属性调试修改(mprop逆向)

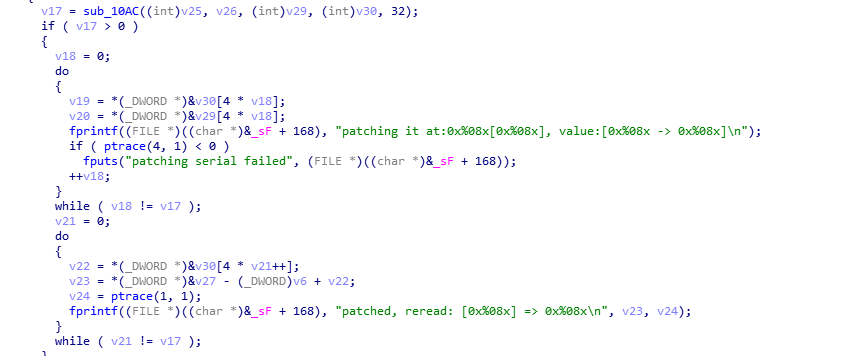
大家都知道如果需要调试android 的程序，以下两个条件满足一个就行。第一是apk的配置文件内的AndroidManifest.xml的 android:debuggable=”true”，第二就是/default.prop中ro.debuggable=1。两种方式第一种通常是解包添加属性再打包，随着加壳软件以及apk校验等，容易出现安装包异常。第二种由于一般的手机发布时ro.debuggable一般是0 也就是不允许调试，通过修改rom的办法在手机上比较麻烦，需要刷机等等，模拟器上一般是vmdk的虚拟机，也没法修改rom。这样我们就无法调试应用了，比如使用DDMS时，只能看到手机，看不到进程信息。



后来发现有人提供了一个软件叫做mprop，但是只有二进制的文件arm架构的，没有源码，一个问题是在有些手机上测试是无效的，第二个问题是在X86模拟器上调试就不可能了。所以尝试一下逆向mprop并且解决这个调试问题。

打开mprop发现主要是附加了1号进程也就是init进程。通过对/dev/\_\_properties\_\_属性段执行修改来修改对应的数值。





针对于android4.4版本的x86模拟器。

我们进入源码中看看为何在init进程这里修改就可以了。在

system/core/init/property\_service.c

system/core/init/init.c

中

Main执行时

property\_init->init\_property\_area-> \_\_system\_property\_area\_init函数（位于bionic/libc/bionic/system\_properties.c）->map\_prop\_area\_rw fd = open(property\_filename, O\_RDWR | O\_CREAT | O\_NOFOLLOW | O\_CLOEXEC | O\_EXCL, 0444)

打开了/dev/\_\_properties\_\_

随后调用property\_load\_boot\_defaults加载了default.prop调用load\_properties 通过property\_set(key, value);将对应的键值设置好。

在 property\_set 函数中 ro.开头的函数只能设置一次。否则就返回失败。所以我们后续调用setprop就是无效的。否则将调用\_\_system\_property\_update执行更新。因此我们可以模拟\_\_system\_property\_find 得到prop\_info结构

\_\_system\_property\_update prop\_info结构中的数据

（bionic/libc/bionic/system\_properties.c）实现。

关键就是 这个结构。name中是ro.debuggable value中是0

**struct** prop\_info

{

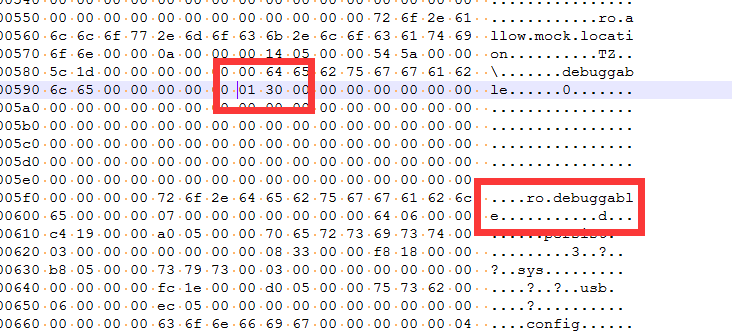
**unsigned** **volatile** serial;

**char** value[PROP\_VALUE\_MAX];

**char** name[0];

};

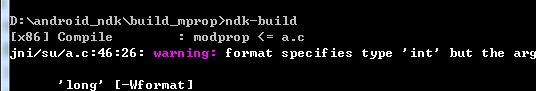
因此我们只要在 /dev/\_\_properties\_\_ 段中找到 对应的name然后向上移动PROP\_VALUE\_MAX(android4.4中是96)个字节。修改value为对应的值即可。我们dump可以看出来.

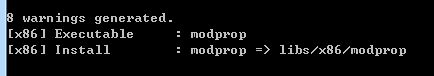


剩下就是编码 主要是通过ptrace进入init然后修改memory具体可以下代码

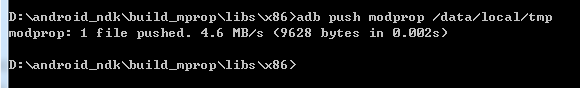
后续的操作如下

编译modprop

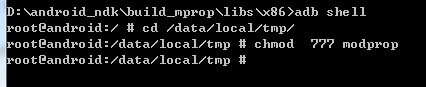




Push到/data/local/tmp



修改可执行



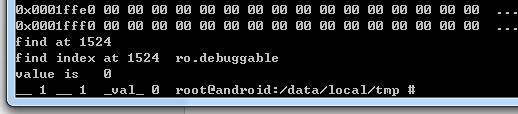
查看当前值



执行修改



修改完毕



获取修改后的值



运行微信。可以调试了

