Android属性（property）机制

 (2011-11-17 16:45:39)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
|  |  |

1. 属性简介

Android里有很多属性（property），每个属性都有一个名称和值，他们都是字符串格式。这些属性定义了Android系统的一些公共系统属性。比如：

[dalvik.vm.dexopt-flags]: [m=y]

[dalvik.vm.heapgrowthlimit]: [48m]

[gsm.operator.iso-country]: []

[gsm.operator.isroaming]: [false]

[gsm.operator.numeric]: []

[gsm.sim.operator.alpha]: []

[gsm.sim.operator.iso-country]: []

[gsm.sim.operator.numeric]: []

这些属性多数是开机启动时预先设定的，也有一些是动态加载的。

系统启动时以下面的次序加载预先设定属性：

/default.prop

/system/build.prop

/system/default.prop

/data/local.prop

/data/property/\*

后加载的如果有重名的则覆盖前面的。

有两种属性值得一提：

persist.\* : 以persist开始的属性会在/data/property存一个副本。也就是说，如果程序调property\_set设了一个以persist为前缀的属性，系统会在/data/property/\*里加一个文件记录这个属性，重启以后这个属性还有。如果property\_set其它属性，因为属性是在内存里存，所以重启后这个属性就没有了。

ro.\* :以ro为前缀的属性不能修改。

2. 应用程序属性使用方法

在java应用里设置属性：

import android.os.SystemProperties;

SystemProperties.set("persist.sys.country",”china”);

在java里取得属性：

String vmHeapSize = SystemProperties.get("dalvik.vm.heapgrowthlimit", "24m");

也可以用SystemProperties.getBoolean，getInt等。

在native C中设置属性：

#include "cutils/properties.h"

property\_set("vold.decrypt", "trigger\_load\_persist\_props");

在C中取得属性：

  char encrypted\_state[32];

  property\_get("ro.crypto.state", encrypted\_state, "");

最后一个参数是默认值。

3. 启动脚本中属性使用方法

一般property启动应该加在init.<your hardware>.rc而不是直接init.rc里。下面是一个init.rc里的例子：

# adbd on at boot in emulator

on property:ro.kernel.qemu=1

start adbd

意思是如果ro.kernel.qemu=1，也就是当前是模拟器的话，则启动adb服务。

4. property权限

只有有权限的进程才能修改属性，要不随便写一个就改系统属性那当黑客也太容易了。

权限在system/core/init/property\_service.c里定义：

property\_perms[] = {

    { "net.rmnet0.",      AID\_RADIO,    0 },

    { "net.gprs.",        AID\_RADIO,    0 },

    { "net.ppp",          AID\_RADIO,    0 },

    { "net.qmi",          AID\_RADIO,    0 },

    { "ril.",             AID\_RADIO,    0 },

    { "gsm.",             AID\_RADIO,    0 },

    { "persist.radio",    AID\_RADIO,    0 },

    { "net.dns",          AID\_RADIO,    0 },

    { "net.",             AID\_SYSTEM,   0 },

    { "dev.",             AID\_SYSTEM,   0 },

    { "runtime.",         AID\_SYSTEM,   0 },

    { "hw.",              AID\_SYSTEM,   0 },

    { "sys.",             AID\_SYSTEM,   0 },

    ...

其实一般应用程序都不会去修改系统属性，所以也不用太在意。

5. 属性实现原理

属性初始化的入口点是property\_init ，在system/core/init/property\_service.c中定义。它的主要工作是申请32k共享内存，其中前1k是属性区的头，后面31k可以存247个属性（受前1k头的限制）。property\_init初始化完property以后，加载/default.prop的属性定义。

其它的系统属性（buil

简要分析一下build.prop是如何生成的。Android的build.prop文件是在Android编译时收集的各种property（LCD density/语言/编译时间, etc.），编译完成之后，文件生成在out/target/product/<board>/system/目录下。在Android运行时可以通过property\_get()[c/c++域] / SystemProperties\_get\*()[Java域]读取这些属性值。

（1）build.prop的生成是由make系统解析build/core/Makefile完成。Makefile中首先定义各种变量，这在下一步执行时会用到。比如：

[cpp] [view plain](http://blog.csdn.net/zhandoushi1982/article/details/7378264) [copy](http://blog.csdn.net/zhandoushi1982/article/details/7378264)

[在CODE上查看代码片](https://code.csdn.net/snippets/1582714)

...

PRODUCT\_DEFAULT\_LANGUAGE="$(calldefault-locale-language,$(PRODUCT\_LOCALES))" \

PRODUCT\_DEFAULT\_REGION="$(calldefault-locale-region,$(PRODUCT\_LOCALES))" \

...

（2）Makefile中调用build/tools/buildinfo.sh执行脚本，并输出到build.prop。Buildinfo.sh很简单，只是echo一些属性，比如：

[cpp] [view plain](http://blog.csdn.net/zhandoushi1982/article/details/7378264) [copy](http://blog.csdn.net/zhandoushi1982/article/details/7378264)

[在CODE上查看代码片](https://code.csdn.net/snippets/1582714)

...

echo"ro.product.locale.language=$PRODUCT\_DEFAULT\_LANGUAGE"

echo"ro.product.locale.region=$PRODUCT\_DEFAULT\_REGION"

...

         ro.product.locale.language/ ro.product.locale.region就是些属性，等号后面是值。

（3）Makefile中直接把$(TARGET\_DEVICE\_DIR)/system.prop的内容追加到build.prop中，还会收集ADDITIONAL\_BUILD\_PROPERTIES中的属性，追加到build.prop中。

ADDITIONAL\_BUILD\_PROPERTIES又会收集PRODUCT\_PROPERTY\_OVERRIDES中定义的属性，如下：

[cpp] [view plain](http://blog.csdn.net/zhandoushi1982/article/details/7378264) [copy](http://blog.csdn.net/zhandoushi1982/article/details/7378264)

[在CODE上查看代码片](https://code.csdn.net/snippets/1582714)

ADDITIONAL\_BUILD\_PROPERTIES:= \

        $(ADDITIONAL\_BUILD\_PROPERTIES)\

        $(PRODUCT\_PROPERTY\_OVERRIDES)

      通过build.prop生成过程的分析，可知哪里可以修改原有的属性或加入自己定义属性，那就是2) buildinfo.sh; 3) system.prop; 4) ADDITIONAL\_BUILD\_PROPERTIES或PRODUCT\_PROPERTY\_OVERRIDES。

（4）属性（property）都有一个名称和值，他们都是字符串格式，用来记录系统设置或进程之间的信息交换。属性是在整个系统中全局可见的。

        在系统初始化时，Android将分配一个共享内存区来存储的属性。这些是由“init”守护进程完成的，“init”守护进程将启动一个属性服务。任何客户端想获得属性信息，可以从共享内存直接读取。客户端应用程序可以调用libcutils中的API函数以GET/SET属性信息：

int property\_get(const char \*key, char \*value, const char \*default\_value);

int property\_set(const char \*key, const char \*value);

       当启动属性服务时，将从以下文件中加载默认属性：

 /default.prop

/system/build.prop

/system/default.prop

/data/local.prop

 属性将会以上述顺序加载，后加载的属性将覆盖原先的值。特别属性如果属性名称以“ro.”开头，那么这个属性被视为只读属性，比如ro.mediatek.version.release=ALPS.ICS2.MP.V1就是指示版本号，应用中用property\_get("ro.mediatek.version.release", val, "unknown");即可用来获得版本信息；属性“ ctrl.start ”和“ ctrl.stop ”是用来启动和停止服务。每一项服务必须在/init.rc中定义，系统启动时init守护进程将解析init.rc和启动属性服务。一旦收到设置“ ctrl.start ”属性的请求，属性服务将使用该属性值作为服务名找到该服务，启动该服务。客户端应用程序可以轮询那个属性值，以确定结果。

==================================================================================================

       Log的调用及等级介绍  
(1)、Log的等级有Verbose,Debug,Info,Warn,Error。  
(2)、java层调用：在java层调用import android.util.Log，在需要打印Log的地方执行Log.v,Log.d,Log.i,Log.w,Log.e.  
(3)、c、c++层调用：在c,c++层包含此头文件：#include <cutils/log.h>，在需要调用Log的地方执行：ALOGV,ALOGD,ALOGI,ALOGW,ALOGE。  
(4)、各个Log等级的使用  
      Verbose: 开发调试过程中一些详细信息，不应该编译进产品中，只在开发阶段使用。（参考api文档的描述：Verbose should never be compiled into anapplication except during development）  
      Debug: 用于调试的信息，编译进产品，但可以在运行时关闭。（参考api文档描述：Debug logs are compiled in but stripped atruntime）  
      Info:例如一些运行时的状态信息，这些状态信息在出现问题的时候能提供帮助。  
      Warn：警告系统出现了异常，即将出现错误。  
      Error：系统已经出现了错误。