Root技术与Android应用开发

百度手机卫士 涂勇策

提纲

- *Android系统权限体系*
- Root技术生态
- Root技术应用场景
- Root技术如何使用
- 产品上的思考

Android系统权限体系

- 两个层面
  - Linux权限
  - Android权限

- Linux权限
  - 基于用户身份的访问控制
    - 用途之一：隔离不同应用，为应用建立边界
  - 基于文件权限的访问控制

Android系统权限体系（续）

- Android权限
  - 保护Android API和未公开接口的访问

```
<!--Allows an application to call
     {@link android.app.ActivityManager#forceStopPackage}.
     @hide-->
<permission android:name="android.permission.FORCE_STOP_PACKAGES"
     android:permissionGroup="android.permission-group.SYSTEM_TOOLS"
     android:protectionLevel="signature"
     android:label="@string/permlab_forceStopPackages"
     android:description="@string/permdesc_forceStopPackages" />
```

(frameworks/base/core/res/AndroidManifest.xml)

```
@Override
public void forceStopPackage(final String packageName, int userId) {
    if (checkCallingPermission(android.Manifest.permission.FORCE_STOP_PACKAGES)
            != PackageManager.PERMISSION_GRANTED) {
        String msg = "Permission Denial: forceStopPackage() from pid="
                + Binder.getCallingPid()
                + ", uid=" + Binder.getCallingUid()
                + " requires " + android.Manifest.permission.FORCE_STOP_PACKAGES;
        Slog.w(TAG, msg);
        throw new SecurityException(msg);
    }
```

(frameworks/base/services/java/com/android/server/am/ActivityManagerService.java)

提纲

- Android系统权限体系
- *Root技术生态*
- Root技术应用场景
- Root技术如何使用
- 产品上的思考

Root技术生态

- 为什么需要Root
  - 产品价值
    - 扩展系统功能，满足用户需求
  - Root具有的能力
    - root为系统最高权限，进而可获取系统任意权限
    - 可访问任意系统服务、资源和数据

Root技术生态 (续)

- Root技术生态
  - Root手机
    - 通过刷机方式植入Root能力
    - 利用系统漏洞破解系统获取Root能力
      - 手机版、PC版
  - “su binary” + 授权管理应用
    - 负责向其它应用授予Root权限
    - 更进一步，接管系统中更多的权限授予
      - 权限管理类应用
  - Root应用
    - 申请Root权限，并执行任务
    - 系统工具类应用、安全类应用

提纲

- Android系统权限体系
- Root技术生态
- *Root技术应用场景*
- 应用开发中使用Root技术
- 产品上的思考

Root技术应用场景

- Root技术常见应用场景
  - 流量防火墙
  - 应用管理
  - 进程管理
  - 权限管理
  - 快捷键
  - 系统诊断工具
  - ……
- 恶意应用
  - 一切皆有可能

提纲

- Android系统权限体系
- Root技术生态
- Root技术应用场景
- *Root技术如何使用*
- 产品上的思考

Root技术如何使用

- Root技术常见应用方式
  - 按需启动su进程执行命令
  - 常驻su进程执行命令
  - 执行Jar中代码
  - 向系统注入服务
- 本质
  - 通过启动su进程来拿到root权限
    - Runtime.exec(“su”)

Root技术如何使用 (续)

- 启动su进程执行命令
  - 优点
    - 原理和实现都简单
    - 有现成的开源库可用
  - 缺点
    - 多次请求Root授权
    - 请求Root授权可能会对产品的交互设计产生影响
- 按需执行和常驻su进程

```
String[] cmds = new String[] {
        "pm install -r \"" + apkFile.getAbsolutePath() + "\""
};
Shell.SU.run(cmds);
```

Root技术如何使用 (续)

- 执行Jar中代码
  - 通过su启动Shell进程
  - 执行app_process命令
  - 在app_process命令中指定要执行的Jar（或APK）和代码

```
ublic void run() {
    String jarFile = appContext.getPackageCodePath();
    String[] cmds = new String[] {
            "export CLASSPATH=" + jarFile,
            "/system/bin/app_process /system/bin me.ycdev.demo.rootapp.jar.TaskExecutor reboot"
    };
    Shell.SU.run(cmds);
```

Root技术如何使用 (续)

- 向系统注入服务
  - 优点
    - 注入时申请Root授权一次即可
    - 随时可用，且可在应用间共享
    - 对产品的交互设计影响最小
  - 缺点
    - 实现较其它方式麻烦很多，没有开源方案，需要自己实现
  - 安全性
    - 需要做好权限验证，不要留下安全漏洞

Root技术如何使用 (续)

- 参考资料
  - How-To SU (http://su.chainfire.eu/)
    - 如何正确使用su
    - SEAndroid对root应用的影响
    - libsuperuser库
      - Github: https://github.com/Chainfire/libsuperuser
  - RootAppDemo
    - 为本幻灯写的Demo
      - Github: https://github.com/ycdev/RootAppDemo
    - 演示了
      - 通过su进程执行命令
      - 通过app_process执行Jar中代码
- 技术兼容性
  - 关注SEAndroid

提纲

- Android系统权限体系
- Root技术生态
- Root技术应用场景
- Root技术如何使用
- ***产品上的思考***

产品上的思考

- 作为一个用户
  - 树立财产安全、隐私和数据安全意识
  - 评估Root手机的安全风险
  - 评估授予Root权限的安全风险
  - 考虑专业可靠的安全工具来降低风险
- 作为一个开发者
  - 创造产品价值
  - 保护手机安全
  - 保护用户隐私

Don’t be evil!

Thanks

Email: tuyongce@gmail.com