TECHNICAL NOTE

Use Guide For AndroidSecurityTools

	AUTHOR	APPROVALS		QUALITY
		LEVEL 1	LEVEL 2	
NAME	Song jinshi			
FUNCTION	Software Engineer			
DATE				
SIGNATURE				

ALCATEL
mobile phones

DOCUMENT HISTORY

Version	Date	Author	Type of Modification
0.1	03/19/14	Song jinshi	Creation

- 什么是 **AndroidSecurityTools**

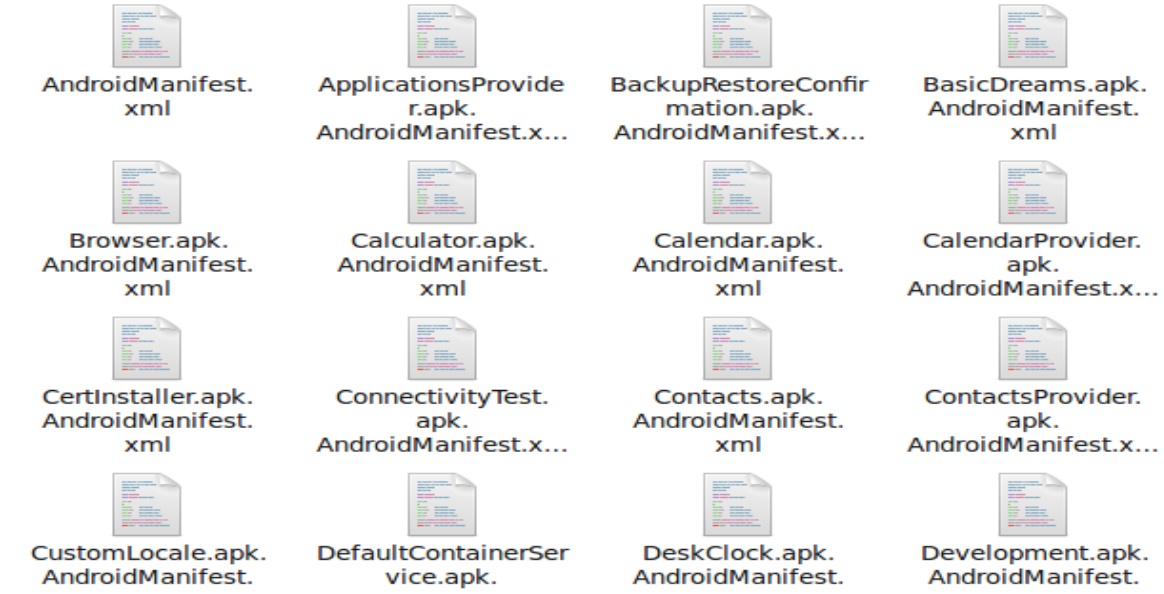
AndroidSecurityTools 是根据运营商（当前版本是根据 AT&T）的需求，将系统中有关安全、敏感以及数据保护相关的信息自动导出到指定格式的文档中，开发此工具的目的是实现自动化，一键导出，省去人工检查和导出的时间，同时能够避免人为检查和导出大量数据时的错误。

- **AndroidSecurityTools** 的框架结构

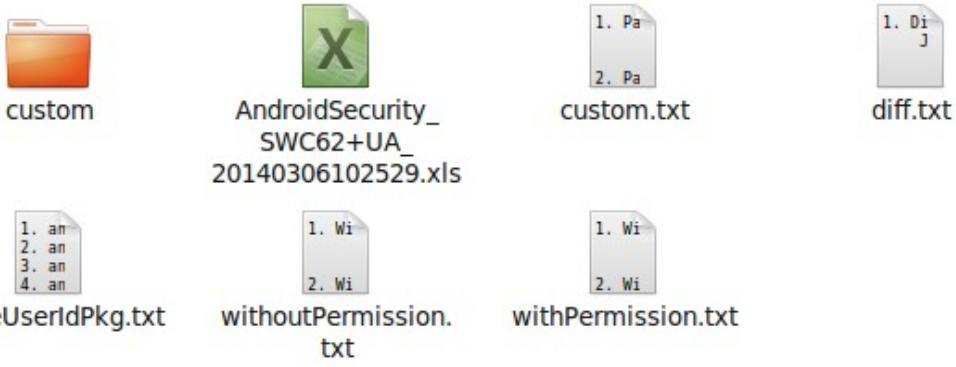
AndroidSecurityTools 的目录框架结构参照下图：

- **AndroidSecurityTools** 的目录模块介绍

manifestList_emu：此目录主要存放的是从 AVD（AndroidVirtualDevice）中 pull 出的所有 APK 的 AndroidManifest.xml 文件，pull 方法下面会有介绍，主要是使用 PullAndroidManifestTool 目录下提供的工具。运行整个工具之前必须保证此目录存在并且包含正确版本的原生文件，此目录下的文件可以根据不同的需求进行替换，大概目录结构如下图：



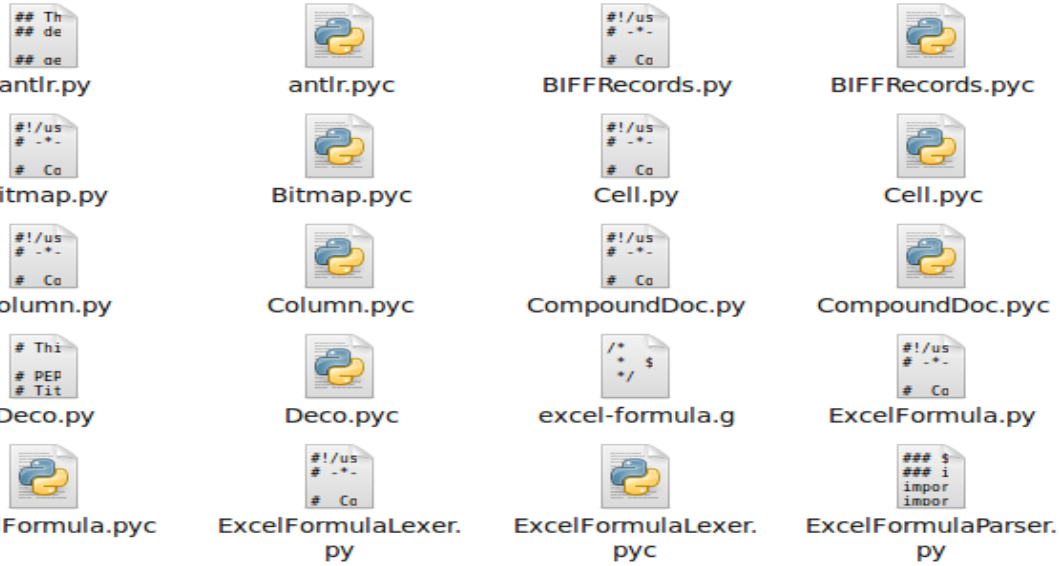
out：此目录存放的是整个工具最终产生的输出结果，大概目录结果参照下图：



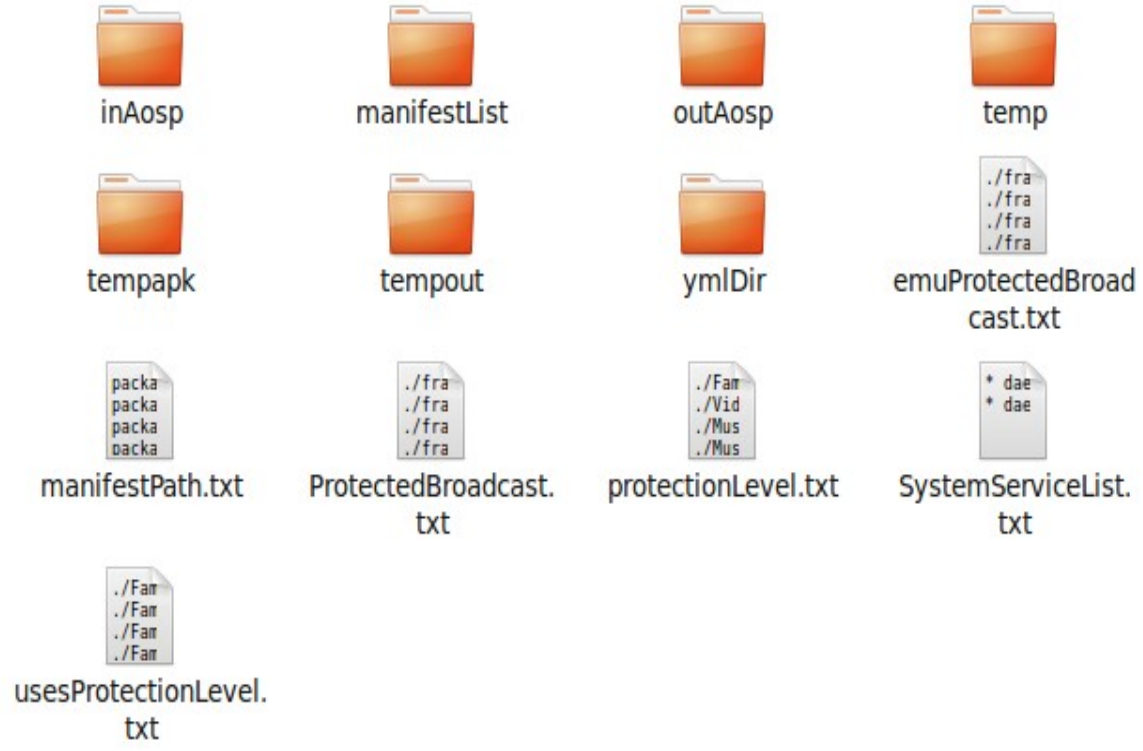
PullAndroidManifestTool：此目录存放的是整个工具的基础部分，包括循环遍历 pull 手机中指定目录下的 APK，然后将 APK 解码得到 AndroidManifest.xml 文件以及.yml 文件，manifestList_emu 目录下的所有文件就是使用此目录下的工具得出，运行此工具之前必须保证手机与电脑相连，同时 adb debug 已经打开，否则将会产生错误，具体的目录结构参照下图：

PyExcelerator：PyExcelerator 是一个第三方库，用来处理 Excel 文件，它的主要优势是写入 Excel 文档，在相关接口上面提供的比较完善。官方主页是 http://sourceforge.net/projects/pyexcelerator/

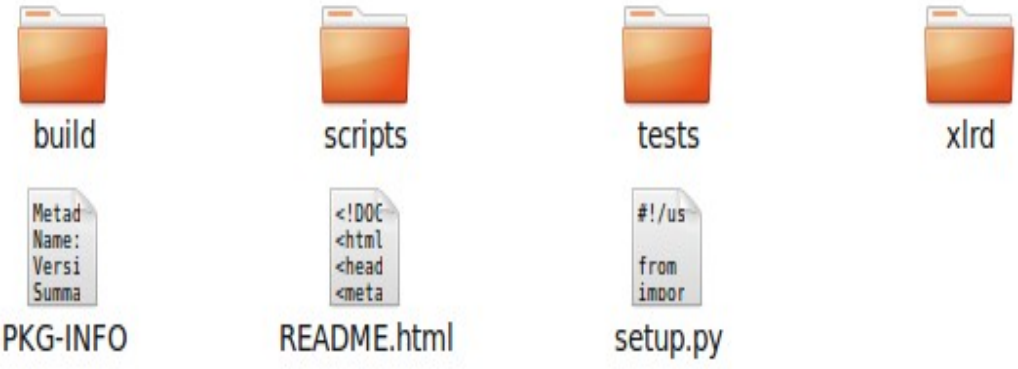
，引用方式为 from PyExcelerator import *，具体的目录结构参照下图：



temp：此目录是整个工具在过滤时的临时目录，其中 manifestList 是手机中 pull 出的 AndroidManifest 文件，inAosp 文件夹存放原生代码中已存在 apk 的 AndroidManifest 文件，outAosp 文件夹存放 Jrd 新增的 apk 的 AndroidManifest 文件，ymlDir 存放的是解码时得到的 yml 文件，manifestList 存放的是解码时得到的 manifest 文件，temp 存放的是从手机中 pull 出来的所有文件包括非 apk 文件，tempapk 存放的是从 temp 目录中过滤出的 APK 文件，tempout 存放的是 apktool 在解码时的输出文件，另外还有 emuProtectedBroadcast.txt、manifestPath.txt、ProtectedBroadcast.txt、protectionLevel.txt、SystemServiceList.txt 以及 usesProtectionLevel.txt 等文件，分别是不同功能需要的字典文件，具体目录结构如下 图：



xlrd-0.9.2：xlrd-0.9.2 是一个第三方库，用来处理 Excel 格式的文件，它的主要优势是读取 Excel 文档，在相关接口上面提供的比较完善。官方主页是 https://pypi.python.org/pypi/xlrd，引用方式是先按照官方的指导文档安装到本机，然后再使用 import xlrd，具体的目录结构如下图：



OneKeyAndroidSecurity.py：此文件是整个工具的入口，提供整合后的一键导出功能，其中就包括引用其他 4 个模块以及相关字典的处理操作，大概代码结构如下图：

```
#!/usr/bin/python
#Output ProtectedBroadcast Excel Table
#jinshi.song

import os
import sys
import re
import time
import shutil
import codecs
from PyExcelerator import *
import FilterSensitiveContentProvider as P
import xlrd
import ProtectedBroadcast as PB
import SystemService as SS
import BundledPackages as BP

outXls = P.outdir + "/AndroidSecurity_"+sys.argv[1]+time.strftime('_%Y%m%d%H%M%S')+".xls"


def main():
    P.prepareFilesFromPhone()
    if not os.path.exists(P.EmuListPath):
        print "Please copy emu android manifest running this script! Directory path is:\n" +     P.EmuListPath
        return
    else:
        P.prepareDirsAndDicts()
        _wb = Workbook()
        PB.Output(_wb)

        P.Output(_wb)
        SS.Output(_wb)
        BP.Output(_wb)
        _wb.save(outXls)
        print "Generate xls table successed!! --> %s" % outXls

if __name__ == '__main__':
    main()
```

BundledPackages.py： BundledPackages 模块的完整处理过程，提供接口给 OneKeyAndroidSecurity.py，用以整合，主要接口的代码结构如下图：

```
def Output(_wb):
    #P.prepareFilesFromPhone()
    P.getProtectLevelFromManifest('permission ', P.protectionLevelTxt)
    P.getProtectLevelFromManifest('uses-permission', usesProtectionLevelTxt)
    if not os.path.exists(P.EmuListPath):
        print "Please copy emu android manifest running this script! Directory path is:\n" +     EmuListPath
        return
    else:

        #add by jinshi.song
        DictExcel = xlrd.open_workbook(P.DictXls)
        #print DictExcel.sheet_names()
        BundlePackageSheet = DictExcel.sheet_by_name(u'bundlepackage')

        for rownum in range(BundlePackageSheet.nrows):
            #print BundlePackageSheet.row_values(rownum)
            key=BundlePackageSheet.row(rownum)[1].value
            #print key
            if not BundlePackageDict.has_key(key):
                BundlePackageDict[key]=BundlePackageSheet.row_values(rownum)
        #P.prepareDirsAndDicts()
        #P.getProtectLevelFromManifest('permission ', protectionLevelTxt)
        P.generatePackageInstallationToPathDict()
        P.generateProtectionLevelToProtectionLevelDict()

        pkgProtectionLevelDict = genPkgPermissionProtectionLevelDict(P.protectionLevelTxt)
        pkgPermissionDict = genPkgAndPermssionDict(P.protectionLevelTxt)
        pkgUsesPermissionDict = genPkgAndPermssionDict(usesProtectionLevelTxt)

        P.filterCustomOEM()
        pkgSourceDict = P.genPkgSourceDict(P.outList)
        #print pkgSourceDict

        outList = genBundledPkgInfo(pkgPermissionDict, pkgUsesPermissionDict, pkgSourceDict, pkgProtectionLevelDict)
        style = P.setStyles(False)
        style_title = P.setStyles(True)
        style_pkg = setPkgStyle()
        initWorkbook(style, style_title, style_pkg, outList,BundlePackageDict,_wb)
```

FilterSensitiveContentProvider.py： FilterSensitiveContentProvider 模块的完整处理过程，提供接口给 OneKeyAndroidSecurity.py， 主要接口的代码结构如下图：

```
def Output(_wb):
        filterCustomOEM()
        #print "Filter CustomOEM content provider successed!!!\n"

        style = setStyles(False)
        style_title = setStyles(True)
        initWorkbook(style, style_title, outList,_wb)
```

ProtectedBroadcast.py：ProtectedBroadcast 模块的完整实现处理过程，提供接口给 OneKeyAndroidSecurity.py，主要接口的代码结果如下图：

```
def Output(_wb):
    #P.prepareFilesFromPhone()
    #get ProtectedBroadcast From EmuManifestListPath --> emuProtectedBroadcastTxt
    P.grepTagToOutputByPath(P.EmuListPath,'protected-broadcast', P.emuProtectedBroadcastTxt)
    #get ProtectedBroadcast From ManifestListPath --> protectedBroadcastTxt
    P.grepTagToOutputByPath(P.ManifestListPath,'protected-broadcast', P.ProtectedBroadcastTxt)

    if not os.path.exists(P.EmuListPath):
        print "Please copy emu android manifest running this script! Directory path is:\n" +     EmuListPath
        return
    else:
        protectedBroadcastDict = getProtectedBroadcastDict(P.ProtectedBroadcastTxt,P.emuProtectedBroadcastTxt)

        style = P.setStyles(False)
        style_title = P.setStyles(True)
        initWorkbook(style, style_title, protectedBroadcastDict,_wb)
```

SystemService.py ：SystemService 模块的完整实现处理过程，提供接口给 OneKeyAndroidSecurity.py，主要接口的代码结果如下图：

```
def Output(_wb):
    #P.prepareFilesFromPhone()
    os.system("adb shell service list > %s" % (P.SystemServiceTxt))
    f = open(P.SystemServiceTxt, 'r')
    f.readline()
    while True:
        line=f.readline()
        if not line:
            break
        if line.find(':') > -1:
            #print line
            idx1 = line.find(':')
            key=line[:idx1]
            idx1=key.index('    ')
            key=key[idx1:]
            key=key.strip()
            SystemSerivceList.append(key)
            #print key

    if not os.path.exists(P.EmuListPath):
        print "Please copy emu android manifest running this script! Directory path is:\n" +     EmuListPath
        return
    else:
        #protectedBroadcastDict = getProtectedBroadcastDict(P.ProtectedBroadcastTxt,P.emuProtectedBroadcastTxt)
        DictExcel = xlrd.open_workbook(P.DictXls)
        #print DictExcel.sheet_names()

        SystemServiceSheet = DictExcel.sheet_by_name(u'systemservice')

        for rownum in range(SystemServiceSheet.nrows):
            #print SystemServiceSheet.row_values(rownum)
            key=SystemServiceSheet.row(rownum)[0].value
            #print key
            if not SystemSerivceDict.has_key(key):
                SystemSerivceDict[key]=SystemServiceSheet.row_values(rownum)

        style = P.setStyles(False)
        style_title = P.setStyles(True)
        initWorkbook(style, style_title, SystemSerivceList,SystemSerivceDict,_wb)
```

SystemServiceAndBundlePackageDict.xls：此文件中包括两张表格，都是需要人工填写的目的、理由或者说明，它的主要作用就是为自动化工具提供需要人工填写的内容。

END OF DOCUMENT