浅谈 Android 软件安全自动化审计

Author： riusksk（泉哥）
Email： riusksk@qq.com
Team： 腾讯安全应急响应中心
Blog： http://hi.baidu.com/riusksk
Date： 2012-05-25

【目录】

【正文】

0x00 前言

随着移动互联网的发展，移动终端安全也越来越受到关注。特别是 Android 系统的崛起，互联网上的各类 Andriod 软件数量迅速上升。因 Android 系统是开源的，导致各种 android 恶意软件迅猛增加，成为手机系统的最大受害者。与此同时，android 系统和软件本身的漏洞也进一步危害到用户的隐私安全。本文主要针对 Android 软件安全进行审计，提供一些常见的安全检测点，并借此实现自动化审计工具，最后评估下业界常见 android 软件安全的现状。

0x10 Android 软件常见漏洞原理及检测

0x11 敏感信息明文保存

敏感信息明文保存一直是 Andriod 软件常容易出现的，这些敏感信息包括用户密码、住址、姓名等等内容，特别是密码问题。有些支持“记住密码”功能的软件常常将密码明文保存在软件目录下的某 xml 文件中，若手机中毒或者被其它恶意软件利用，直接读取 xml 文件中的明文密码然后远程发送给攻击者，这将会严重影响到用户帐号安全。以下是用于检测敏感信息的关键代码：

```
foreach $msg (@msgs) {
```

```
    my $base64msg = encode_base64($msg);
    my $md5msg = md5_hex($msg);
    if(/$msg/)
    {
        print "[高危]：$curfile: 行 $lines: 发现明文敏感信息：$msg !!!\n";
        $SearchStrResult .= "[高危]：$curfile: 行 $lines: 发现明文敏感信息：$msg !!!\n";
        push (@risk_level,4);
        $highnum++;
    }

    if(/$base64msg/)
    {
        print "[高危]：$curfile: 行 $lines: 发现 Base64 敏感信息：$base64msg !!!\n";
        $SearchStrResult .= "[高危]：$curfile: 行 $lines: 发现 Base64 敏感信息：
$base64msg !!!\n";
        push (@risk_level,4);
        $highnum++;
    }

    if(/$md5msg/)
    {
        print "[安全]：$curfile: 行 $lines: 发现 MD5 加密信息：$md5msg !!!\n";
        $SearchStrResult .= "[安全]：$curfile：行 $lines：发现 MD5 加密信息：$md5msg !!!\n";
        push (@risk_level,1);
        $secnum++;
    }
 }
```

以下是当检测到明文密码时的输出结果（已过滤掉包名）：

```
[高危]： data/data/com.***.v7/shared_prefs/rem_password.xml: 行 3: 发现明文敏感信息：
passwd123 !!!
```

0x12 程序文件及进程权限问题

如果我们限制程序目录的权限，禁止其它第三方软件访问，也能够在一定程度上防止明文密码泄露，除非软件已获得 root 权限，而软件一般无需 root 权限，大多是一些恶意软件或者杀毒软件才会用到，对于普通软件应该遵循“最低授权原则”，尽量以最低权限来完成所需操作，避免权限被恶意滥用。因此也可将程序目录权限和进程是否为 root 权限作为一个检测点，以下是 DroidAppAuditter 检测到问题时的输出结果：

```
[低危]：-rw-rw--w- ：data/data/com.snda.cloudary/shared_prefs/UserInfo.xml
[低危]：存在 Root 权限 !
```

0x13 网络数据明文传输

在 2012 年 2 月 UCWEB 浏览器被爆存在明文密码网络传输漏洞，若开启“云端加速”功能，在登录一些 SSL 站点时，它会将用户发送的 WEB 请求会转发到 UC 的代理服务器上，并且未进行任何加密处理，可能导致用户密码泄露。最初 UC 不承认此问题，后来在微博上确认存在此漏洞。对于存在明文网络传输的软件，若结合中间人攻击就极有可能造成密码泄露，特别是在 KFC、麦当劳等公共场所。在 Android 中本身自带有 tcpdump 可用于网络抓包，再借助 perl 模块去解析 cap，或者利用 wireshark 的命令行工具来解析也是可行的（以下为示例代码，不考虑命令注入问题）：

```
print "[*] 捕获网络数据包……\n\n";
system("adb shell tcpdump -p -vv -s 0 -c 500 -w /sdcard/$targetfile.cap");

print "\n[*] 提取 cap 文件至 PC 端分析……\n\n";
system("adb pull sdcard/$targetfile.cap $cwd/TestSoft/$targetfile/$targetfile.cap");

print "\n[*] 解析 cap 数据包……\n\n";
system("$cwd/tshark/tshark.exe -r $cwd/TestSoft/$targetfile/$targetfile.cap -V > $cwd/TestSoft/$targetfile/$targetfile.txt");
```

以下是检测到明文敏感信息网络传输时的输出结果：

```
[中危]：存在明文传输内容:
password=passwd123&provider=sdo&login=13613******

[低危]：存在 IMEI（国际移动设备身份码）隐私信息窃取行为、IP 地址隐私信息窃取行为、SIM 序列号隐私信息窃取行为:
channel=6666&model=sdk&ip=10.0.2.15&mac=&sim=8901410321111851072 0&imei=000000000000000&size=480*800&os=2.2&platform=Android&version=1.0.5.1&ua=Mozilla/5.0 (Linux; U; Android 2.2; zh-cn; sdk Build/FRF91) AppleWebKit/533.1
```

0x14 组件权限安全问题

在 Android 中存在多种组件，比如 Content Provider、Broadcast Receiver 等等，这些组件可能因权限设置不当导致信息泄露或者钓鱼欺骗等攻击。在 2011 年，香港理工大学的安全研究人员陆续报告了许多 android 软件中存在的 Content Provider 信息泄露漏洞（参见链接：http://www4.comp.polyu.edu.hk/~appsec/），由于程序对 Provider 的权限设置不当，导致第三方软件可读取 Content Provider 提供的信息，其危害程度取决于 Content Provider 提供的信息内容，比如联系人、电话、短信等隐私信息就可能包含其中。默认情况下，Content Provider 的权限为 android:protectionLevel=”normal”，最好的权限设置应为 signature 或者 signatureOrSystem，进而避免被第三方恶意软件利用。下面是 DroidAppAuditter 中用于检测 Content Provider 权限的关键代码：

```
while(index($cont[$line],"<permission android:name=\"$ProviderName\"") == -1){
```

```
    if ($line <= $linecount) {
        $line++;
    }
    else{
        print "[中危]：Content Provider：$ProviderName 默认设置为\“normal\”权限，可能导致敏感信息泄露!\n";
        $CheckComponentResult .= "[中危]：Content Provider：$ProviderName 默认设置为\“normal\”权限，可能导致敏感信息泄露，建议修改为\“signature\”或者\“signatureOrSystem\”!\n";
        ……省略……
        last;
            }
        }
    if(index($cont[$line],"android:protectionLevel=\"normal\"") > -1){
        print "[中危]：Content Provider：$ProviderName 权限为\“normal\”，可能导致敏感信息泄露！\n";
        $CheckComponentResult .= "[中危]：Content Provider：$ProviderName 为\“normal\”权限，可能导致敏感信息泄露，建议修改为\“signature\”或者\“signatureOrSystem\”！\n";
        ……省略……
    }
```

除 protectionLevel 权限问题外，<grant-uri-permission>权限问题也可作为一个检测点，若设为 true，可被其它程序通过 URL 访问到 content provider 的内容，容易造成信息泄露。

2011 年盛大的安全研究人员 DoDo 在其博客写一系列关于 Android 安全的文章，其中有篇就提到关于 Broadcast Receiver 的问题，链接见：http://www.sectop.com/?p=128，这个依然是组件权限控制的问题，由于 Broadcast Receiver 被允许外部调用，可能被第三方恶意软件标记相关 action 给正常应用发送 broadcast，那么正常应用就会对此进行响应处理，其危害程度也是取决于正常应用对广播接收采取的操作，比如发送短信、任务栏通知等操作，就有可能被用于钓鱼欺骗。防御此问题最简单的方法就是通过 android:exproted 来关闭 receiver 对外部应用的响应，但可能在同一公司开发的一些产品之间需要进行相互广播通讯，此时就可采取 DoDo 在文中提到的另一种方法：在 receiver 方的 androidmanifest.xml 中增加<permission>自定义权限项，并在 sendbroadcast 方得 androidmanifest.xml 中增加 <uses-permission> 匹配此权限。下面是 DroidAppAuditter 用于检测 Broadcast Receiver 权限的部分检测代码：

```
if(index($cont[$line],"android:exported=\"flase\"") > -1){
    print "[安全]：Broadcast Receiver：$Receiver 禁止被外部调用！\n";
    $CheckComponentResult .= "[安全]：$Receiver 禁止被外部调用！\n";
}
elsif(index($cont[$line],"android:permission=") == -1){
    print "[低危]：Broadcast Receiver：$Receiver 可被外部调用，可能造成钓鱼欺骗，建议添加 android:exported=\"flase\"！\n";
    $CheckComponentResult .= "[低危]：Brodcat Receiver：$Receiver 可被外部调用，可能造成钓鱼欺骗，建议添加 android:exported=\"flase\"！\n";
    ……省略……
```

```
}
```

除 receiver 和 content provider 的权限问题外，其它组件如 service 和 activies 也可能存在权限安全问题，亦可将其列入检测范围。

0x15 其它

除上文中提到的各个安全检查点之外，读者还可进行扩充，比如 XSS、SQL 注入检测、内存泄露、intent fuzzing 等多个方向进行分析，DoDo’s Blog 上面也有一些安全文章可作为参考，大家如果有什么好的思路或检测点也可邮件与本人交流：riusksk@qq.com。

0x20 自动化审计工具——DroidAppAuditter

对于业务量较大的公司，Android 软件产品可能会很多，若一一通过手工检测，就会影响到效率，因此实现一款自动化审计工具是非常有必要的，至少可以减轻工作量，以腾出更多的时间出来看片……笔者将此审计工具命名为 DroidAppAuditter，主要用 Perl 语言编写的，支持 Android 模拟器和实体机，可自动安装 apk 文件并运行软件，然后实现动态自动化检测，并图文并茂地输出分析报告。最后这里附上一份输出的完整分析报告，如图 1 所示，报告中的漏洞已于 2011 年报予盛大公司修复：

图 1

0x02　业界 Android 软件安全现状

2011 年我们借助自动化审计工具 DroidAppAuditter，对业界各个主流 Android 软件进行安全审计，共审计了 29 个常用 Android 软件，其中存在明文保存密码的有 10 个（包括盛大、360、百度、Opera、Google……），其它 XSS 漏洞、明文传输，以及隐私窃取的行为共有 12 个，所有被审计的软件有超过一半是存在安全漏洞的。由此可见，目前 android 软件安全还未受到业界重视，但这也将成为今后发展的趋势。最后附上审计结果的部分截图，如图 2 所示：

1	软件名	版本号	用户隐私保护	文件权限保护	网络通讯保护	运行时解释保护
2	[redacted]	1.0.0	明文保存密码	合格	明文传输	合格
3		1.0.1	明文保存密码	部分文件可被第三方读取	MD5加密	合格
4		2.1.0	未见明文密码	合格	加密传输	无
5		3.0.0.1012	未见明文密码	合格	SSL加密传输	无
6		1.2.4	未见明文密码	部分文件可被第三方读取	未见明文传输	合格
7		1.0.5	未见明文密码	部分文件可被第三方写入	SSL加密传输	合格
8		1.2	未见明文密码	合格	明文传输	存在XSS漏洞
9		1.0.5.1	明文保存密码	合格	将IP、SIM序列号\IMEI明文传输至网站	无
10		3.0.3	明文保存密码	合格	未见明文传输	合格
11		2.3.1	未见明文密码	部分文件可被第三方读取	无	合格
12		2.0.160	MD5加密	部分文件可被第三方读写	MD5加密	无
13		4.2.1	未见明文密码	合格	MD5加密	合格
14		2.5.2	未见明文密码	库文件可被第三方读取	明文传输	合格
15		7.9.3.103	未见明文密码	库文件可被第三方读取	未见明文传输	合格
16		1.7	未见明文密码	库文件可被第三方读取	明文传输	合格
17		3.0 Beta4	未见明文密码	部分文件可被第三方读取	未见明文传输	无
18		1.3.0	未见明文密码	合格	密码加密，IMEI明文传输至网站	无
19		2.51	未见明文密码	部分文件可被第三方读写	将IMEI、IMSI明文传输至网站	合格
20		3.0.0	明文保存密码	部分文件可被第三方写入	将IMEI、IMSI明文传输至网站	合格
21		1.1.374	未见明文密码	部分文件可被第三方写入	明文传输	无
22		1.0.3.50	明文保存密码	库文件可被第三方读取	无	无
23		6.2	明文保存密码	库文件可被第三方读取	无	无
24		2.3.2	明文保存密码	合格	将IMEI、IMSI明文传输至网站	无
25		2.3	未见明文密码	合格	未见明文传输	无
26		2.4.4	明文保存密码	合格	未见明文传输	无
27		1.2	未见明文密码	合格	无	无
28		1.4	明文保存密码	合格	明文传输，并将IMEI明文传输至网站	无
29		1.0.7	未见明文密码	合格	未见明文传输	无
30		1.5	未见明文密码	合格	无	无

图 2

0x03 总结

本文主要探讨了 Android 软件的一些常用安全检测点，并通过代码实现自动化审计工具，希望对各位有所帮助，如果你有更好的检测方案，也可与本人探讨。至于审计工具，由于各种原因就不公开了，因为我相信：“授人以鱼不如授人以渔”。

0x04 关于我们

腾讯安全应急响应中心（Tencent Security Response Center）简称 TSRC，是腾讯公司负责突发安全事件处理的团队，如果您对腾讯产品和业务有任何安全上的建议和意见可以与我们联系，也欢迎有意在互联网安全行业发展的同学加入我们。

邮箱：security#tencent.com

主页：http://security.tencent.com