自动化检测 Android 应用反射型跨站脚本漏洞的方法①

王 岩, 程绍银, 蒋 凡

(中国科学技术大学 计算机科学与技术学院, 合肥 230027)

摘 要: 提出一种自动化检测 Android 应用反射型跨站脚本漏洞的方法, 通过对 Android 应用组件的识别和分类, 自动化输入测试例和点击与输入框关联的按钮, 监测运行结果判断应用是否具有潜在的反射型跨站脚本漏洞, 并通过图像处理方法实现了对 WebView 的支持. 基于该方法实现了一个原型工具. 实验表明, 该方法可以有效的检测 Android 应用的反射型跨站脚本漏洞, 具有较高的实用性.

关键词: Android 应用; 反射型跨站脚本; 自动化测试; 漏洞

Automated Method for Detecting Reflected XSS Vulnerabilities of Android Apps

WANG Yan, CHENG Shao-Yin, JIANG Fan

(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

Abstract: This paper presents an automated method for detecting reflected XSS vulnerabilities of Android Apps. Through identifying and classifying Android Apps components, automatically inputting test cases, clicking on the input box-related buttons and monitoring the results, to determine whether the applications have potential reflected XSS vulnerabilities. Moreover this method implements support for WebView by image processing. Based on this method, a prototyping tool is also implemented. The experiment results demonstrate that this proposed method can detect reflected XSS vulnerabilities of Android Apps with high practicability and effectiveness.

Key words: Android App; reflected XSS; automated testing; vulnerability

随着 Android 应用数目的增长, 应用漏洞引发的安全性问题也愈来愈严重. 目前 AVD[1]中可查的 Android 平台的漏洞超过 660 个, 其中第三方漏洞比例超过 80%. 这些漏洞可能导致用户数据泄露、会话劫持、恶意代码执行等问题, 造成用户不同程度的损失. 在第三方漏洞中, 第三方应用漏洞接近 60%, 其中跨站脚本(Cross Site Scripting, 通常将其缩写成 XSS)这类传统的 WEB 应用漏洞也在 Android 应用中出现. 例如 CVE[2]公布的基于 Android 平台上的 Mozilla Firefox 23.0.1 及之前的版本中存在安全绕过漏洞 (CVE-2013-1727), 攻击者可利用该漏洞绕过同源策略, 进而进行 XSS 攻击或获得密码和 cookie 信息. 类似上述的 XSS 称为反射型 XSS. 发出请求时, XSS 代码出现在 URL 中, 作为输入提交到服务端, 服务端解析后响应, 在响应内容中出现这段 XSS 代码, 最后浏览器解释执行. 这个过程就像一次反射, 故称为反射型 XSS. 由于反射型 XSS 漏洞普遍存在且方便利用, 因此, 对于 Android 应用的反射性 XSS 漏洞的研究具有重要的现实意义和广泛的应用需求.

在 Android 应用程序安全性分析方面, 目前的研究多集中在应用的恶意行为检测方面, 安全漏洞分析方面的研究比较少, 但是随着披露的应用安全漏洞的增加, 这方面的问题越来越引起重视. 周莉捷、雷友珣分析了 Android 浏览器的工作原理, 并仿真验证了在 WebView 中存在用户隐私信息丢失的危险[3]. Bhavani A B 对 Android 的 WebView 和 HttpClient 的 XSS 攻击进行了研究[4], 攻击的结果是可以获取 cookie 和其他的敏感信息. Stanzein Sedol 和 Rahul Johari 分析了 Android 应用的 XSS 攻击的根本原因, 并提出了潜在的解决方案[5]. 但是以上的研究主要是针对于 WebView

① 收稿时间:2014-11-18;收到修改稿时间:2014-12-22

类型, 忽略了对 Android 自带的输入控件 EditText 的研究, 而且这些研究只是方法或者理论的研究, 并未实现自动化检测.

对于 Android 自动化测试方面, 开源测试框架主要有 Activity Instrumentation、Robotium、Robolectric 及 TMTS, 该类自动化测试框架均需要测试人员通过编写测试代码来实现测试用例, 测试效率低下. 侯菊敏提出基于 Android 的关键字驱动自动化测试框架[6], 但是这种方法针对的是 GUI 测试. 马红素使用动态测试技术与终端监控技术相结合的方法[7], 设计并实现了一个自动化检测 Android 应用恶意行为的测试系统. 这种方法需要反编译 App, 获取应用的源码, 并且不支持对 WebView 的测试.

在传统的WEB应用漏洞检测上, 自动化检测工具的工作原理都是扫描用户界面输入入口, 构造 XSS 漏洞触发字符串作为测试例, 发送测试并监测返回结果, 以判断是否存在此类漏洞. 因此在 Android 平台上也可以采用类似的思路. 本文针对 Android 应用的反射型 XSS 漏洞提出了一种自动化的检测方法, 该方法不需要反编译 App, 不涉及 App 的源码与 Android 底层代码, 自动识别输入框和输入测试例, 然后自动点击与输入框关联按钮, 监测运行结果判断是否具有潜在漏洞, 并使用图像处理的方法实现了对 WebView 的支持. 通过该方法实现的原型工具成功挖掘和验证了多个 Android 应用的反射型 XSS 漏洞.

1　系统概述

图 1 描述了自动化检测 Android 应用反射型 XSS 漏洞方法的结构图. 该结构主要分为界面元素抽取模块、识别分类模块、测试例库、结果检测模块 4 个部分.

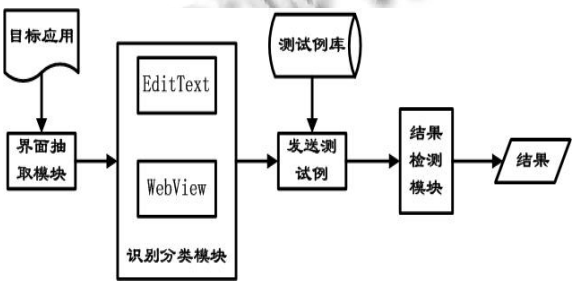

图 1　自动化检测 Android 应用反射型 XSS 漏洞方法

界面元素抽取模块的功能是自动化抽取当前界面所有的控件, 并提取出需要的节点属性. 识别分类模块按照 EditText 和 WebView 两种控件的类型进行输入框和按钮的识别与组合. 测试例库中存放的是设计好的典型测试例, 通过自动化操作, 进行输入和点击. 结果检测模块监测运行结果, 判断应用是否具有潜在的反射型 XSS 漏洞, 返回输入页面, 删除已经输入的测试例, 重新开始测试.

2　各模块算法设计和实现

2.1　界面元素抽取模块

此模块是通过扩展工具 uiautomatorviewer 功能实现的. uiautomatorviewer 是一个用来扫描 UI 上控件的工具. 其返回的是一个按照层次关系建立的控件树. 这棵树的叶子节点就是 Android 的控件, 比如 EditText、WebView、Button 等等. 这些节点包括 index、text、class、package 等属性, 某些属性是本文的方法不需要的. 为了方便获取节点属性, 设计了获取节点名字、是否可点击信息、text 内容、坐标值、高度、宽度、祖先、子孙等属性的方法. 名字用来区分控件. 是否可点击信息在分类时使用. 坐标为绝对坐标值, 为输入测试例和自动点击按钮操作提供焦点.

2.2　识别分类模块

由于在界面上填写数据后, 还需要单击相应的交互控件才能使得输入数据被程序处理, 所以确定行为相关的控件也是一个必备步骤. 在一个 Activity 界面上可能存在多个控件相互关联, 比如在登录界面上必须同时填写用户名和密码再单击登录按钮才能向服务器发送数据. 识别分类模块的作用就是首先识别出输入框, 再将相互关联的控件组合在一起, 简化测试的流程, 提高测试的准确性.

Android 应用的输入框和按钮有以下两种情况:

① 以 Android 独立的控件形式存在. 比如输入框为 EditText, 按钮为 Button、TextView 或者 ImageView.

②WebView 加载的页面中存在输入框及按钮.

对于第一种情况, 通过控件的名字进行控件的识别, 对相关联控件的组合使用图 2 的方法.

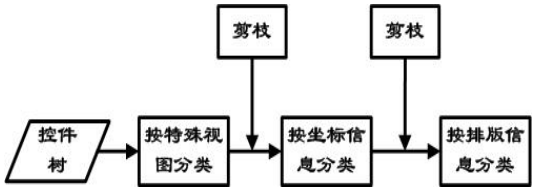

图 2　独立控件分类方法流程

具体流程描述如下所示:

算法 1. 独立控件分类方法

输入: 控件树.
输出: 输入框与按钮的组合.
1. if 控件树包含滚动视图 ScrollView 或网格视图 GridView 或列表 ListView
   输入框和按钮按照相同父辈节点的层次关系进行分类;
   剪枝;
2. 按坐标信息分类;
3. 剪枝;
4. 按排版信息分类.

通过算法 1 的操作, 可以有效的将以独立控件形式存在的输入框和按钮组合在一起.

第二种情况, 对于输入框的识别采用图像处理的方法, 以是否有软键盘弹出来判断是否是输入框.

算法 2. 对 WebView 中输入框的识别

输入: WebView 页面.
输出: 输入框中心坐标(x, y).
1. w = WebView 页面宽度;
   h = WebView 页面高度;
   snapshot = 手机屏幕截图.
2. *snapshot* 二值化.
3. for 在 WebView 范围内的黑色横线 do
   hlength = 横线 *hline* 长度;
   if (*hlength*>=w/2)
   保存在 *hList* 中;
4. for *hline* in *hList* do
   if (|*hline1*.*top*-*hline2*.*top*|<6)
   hList.remove(*hline2*);
5. for *hline* in *hList* do
   hleft = *hline*.*left*;
   hright = *hline*.*right*;
   以 *hleft*±5、*hright*±5 为起点, 寻找黑色竖线 *vline*;
   vlength = *vline* 的长度;
   if(*vlength* >=30)
   保存在 *vList* 中;
6. for *vline* in *vList* do
   筛选 *vline*;
   将 *vline*、*hline* 的坐标修改为边界坐标;
7. x=(*hleft*+*hright*)/2;
   y=(*vtop*+*vbottom*)/2;
   adb shell input tap (x, y);
8. *w2* = 点击后 WebView 页面宽度;
   h2 = 点击后 WebView 页面高度;
   if(*w2*<w || *h2*<h)
   return (x, y);

以 iReader 为例, 对算法 2 进行说明.

图 3 屏幕截屏

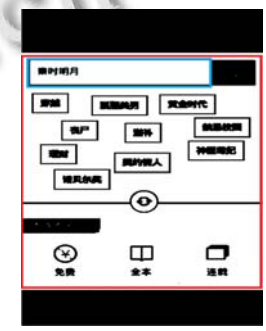
图 4 二值化图片

①w=480, h=600; 图 3 为屏幕截屏.
②二值化图片如图 4 所示, 其中红色标记范围为 WebView.
③*hList*.size()=4;具体坐标值如下所示:

表 1 iReader 的 hList

坐标 / hline	top	left	bottom	right	length
Line1	129	16	129	463	448
Line2	191	14	191	465	452
Line3	192	14	192	465	452
Line4	193	15	193	464	450

④*hList*.size()=2;具体坐标值如下所示:

表 2 筛选后的 hList

坐标 / hline	top	left	bottom	right	length
Line1	129	16	129	463	448
Line2	191	14	191	465	452

⑤、⑥经过筛选, 判断的输入框的坐标为

表 3 输入框坐标

top	left	bottom	right
129	14	191	374

此时, 已经将输入框的坐标范围限制在图 4 的蓝色范围内.

⑦ x=194; y=160.

⑧ $w2$=480; $h2$=232;

$h2<h$,说明有软键盘弹出, 证明此坐标值为输入框范围内的坐标值, 即找到输入框.

实验表明, 这种方法可以有效的识别 WebView 页面中的输入框.

在 WebView 页面中, 可以通过模拟“确定”按钮来实现点击.

2.3 测试例库

根据反射型 XSS 漏洞的原理, 针对 Android 应用的反射型 XSS 漏洞设计的测试例格式如下所示.

表 4　测试例格式

id	type	content	needclick

id 表示测试例的序号, 整型, 是标示测试例的关键字, 按 0,1,2…自然顺序增长.

type 表示测试例的类型, 字符串型. “xss”代表反射型跨站脚本测试例. 此字段作为测试例扩展的区分标志使用, 比如工具对 SQL 注入漏洞的扩展, 可以用“sql”代表 SQL 注入的测试例.

content 表示测试例的内容, 字符串型. 对于 XSS, 主要采用注入 alert 语句的测试例. 在该类测试例的设计中, 需要考虑简单过滤器的存在, 即还需要设计转义后的注入字符串.

例如, <script>alert(vulnerable)</script>

对于需要转义的字符串, 比如" onmouseover="javascript:alert('1'), 因为输入框会对空格进行转义, 所以对空格采取 URL 编码, 得到新的输入:

"%20onmouseover="javascript:alert('1')

needclick 是点击按钮后是否需要再次点击输入框的标志, 布尔型. needclick=true 表示需要再次点击, needclick=false 表示不需要再次点击. 比如" onmouseover="javascript:alert('xss'), 点击按钮后需要再次点击输入框, 才能显示结果.

设计的测试例存入本地, 作为测试例库. 工具提供了查询、添加、删除、更新测试例的功能, 使得测试例库为可扩展的.

2.4 自动化操作

自动输入测试例和点击按钮的是通过扩展 monkeyrunner[8] 功能和发送 adb 命令实现的. monkeyrunner 是 Android SDK 中自带的测试工具, 可应用于功能测试, 回归测试, 并且可以自定义测试扩展. monkeyrunner 工具提供了一个 API, 运用该 API 编写的程序可以不用通过 Android 代码来直接控制 Android 设备和模拟器, 对 Android 应用程序或测试包进行安装、运行、发送模拟按键, 对用户界面进行截图等操作. 但是 monkeyrunner 在输入字符串时, 不支持某些字符和对 WebView 的操作. 所以需要扩展 monkeyrunner 的功能, 实现输入和点击, 同时需要结合 adb 命令来支持 WebView.

具体方法如下所示:

算法 3.　自动化操作算法

输入: 输入框焦点坐标(ix, iy), 按钮焦点坐标(bx, by).
输出: 输入和点击操作.

```
1.  if 独立控件
        device.touch(ix,iy,'DOWN_AND_UP');
        输入测试例;
        device.touch(bx,by,'DOWN_AND_UP');
2.  else
        if  WebView 页面
            adb shell input tap ix iy;
            输入测试例;
            device.press(Android KeyCode);
```

2.5 结果检测模块

针对设计的测试例, 通过监测是否出现弹窗以及弹窗的内容即可辨别是否为潜在的 XSS 漏洞. 弹窗作为控件树的根节点, 尺都要比屏幕的小, 但是又不能只用此标准作为判断 XSS 发生的唯一依据, 有些应用在点击按钮后, 会以弹窗的形式出现提示信息. 所以需要进一步判断弹窗的内容, 才能确认是否发生 XSS 漏洞.

算法 4.　结果检测算法

输入: 当前控件树根节点尺寸 *rootsize*.
输出: 检测结果.

```
1.  if  (rootsize<屏幕尺寸 screensize)
            if  ( TextView 内容=TestCase 内容)
                return true;
            else
                return false;
2.  else
            return false;
```

检测结束后, 如果是弹窗, 自动点击弹窗的确定按钮返回输入页面. 如果是页面, 通过模拟返回操作, 返回到输入页面. 点击输入框后, 将光标移动到输入

框的最后, 获取输入框的 text 的长度, 与输入的测试例长度的最大值进行比较, 模拟删除操作, 将输入的测试例删除. 模拟操作通过发送 Android KeyCode 代码实现.

3 实验结果

利用上述方法, 实现了一个自动化检测 Android 应用反射型 XSS 漏洞的工具. 工具在 Linux 环境下使用 Java 语言开发, 使用的 Android 模拟器为 4.1.2 版本, CPU/ABI 为 ARM(armeabi-v7a), 存储测试例库的本地数据库系统为 MySQL, 通过 JDBC 技术进行操作.

将工具检测结果与 CVE 和乌云网[9]中公开的 XSS 漏洞进行比较, 部分统计结果如下表 5 所示.

表 5 部分实验结果统计表

应用名称	公开漏洞编号	工具检测结果
UC 浏览器	WooYun-2014-75143	检测出
chrome	CVE-2012-4905	检测出
遨游	WooYun-2014-75184	检测出
百度浏览器	WooYun-2014-75184	检测出
网易邮箱	WooYun-2014-75143	检测出
FireFox	CVE-2013-1727	检测出
欧朋浏览器	WooYun-2014-75184	检测出
新浪微博	Wooyun-2012-04745	未检测出
360 浏览器	WooYun-2014-75184	检测出

与公开的漏洞进行比较, 工具的检测准确率约为 89%. 工具还成功的发现了若干未知漏洞, 例如掌阅 iReader 和 2345 浏览器. 与已公开漏洞的比较和成功挖掘出未知漏洞验证了方法的有效性.

以掌阅 iReader 和 UC 浏览器 XSS 为典型漏洞进行分析. 图 6 为掌阅 iReader 的反射型 XSS 漏洞. 图 7 为 UC 浏览器的反射型 XSS 漏洞, 利用该漏洞可以读取 cookie 信息, 可能会造成密码的泄露或者数据信息被读取.

图 5 iReader XSS 漏洞

图 6 UC 浏览器 XSS 漏洞

4 结语

该方法采用的测试例库为可扩展的, 所以在配备相应的结果检测方法的情况下, 对于类似 XSS 这种需要在界面上填写数据后单击相应的交互控件来挖掘的漏洞都是有效的, 比如 SQL 注入漏洞. 该方法也为自动化检测存储型 XSS 提供了借鉴, 未来会在已有工作的基础上研究自动化挖掘存储型 XSS 漏洞的方法.

本文提出一种自动化检测 Android 应用反射型 XSS 漏洞的方法, 通过对 Android 应用组件的识别和分类, 自动化输入测试例和点击按钮, 监测运行结果判断应用是否具有潜在的漏洞, 并通过图像处理方法实现了对 WebView 的支持. 实现了该方法的工具原型, 通过测试, 成功的发现了 iReader、UC 浏览器等应用的反射型 XSS 漏洞, 验证了方法的有效性.

参考文献

1 AVD. http://android.scap.org.cn/.

2 CVE. http://cve.scap.org.cn/.

3 周莉婕,雷友珣.Android 操作系统浏览器安全漏洞的分析. 软件,2013,34(5):107–111.

4 Bhavani AB. Cross-site scripting attacks on Android web view. International Journal of Computer Science and Network, 2013, 2(2): 1–5.

5 Sedol S, Johari R. Survey of cross-site scripting attack in Android apps. International Journal of Information & Computation Technology, 2014, 4(11): 1079–1084.

6 侯菊敏.基于 Android 的关键字驱动自动化测试框架研究[硕士学位论文].广州:中山大学,2012.

7 马红素.Android 开放平台应用程序的安全检测系统设计与实现研究[硕士学位论文].北京:北京邮电大学,2013.

8 monkeyrunner. http://api.apkbus.com/guide/developing/tools/monkeyrunner_concepts.html.

9 WooYun. http://www.wooyun.org/.