

面向服务端私有 Web API 的自动发现技术研究

陈佳¹, 郭山清^{1,2}

(1. 山东大学计算机科学与技术学院, 山东 济南 250101;

2. 山东大学教育部直属密码学和信息安全重点实验室, 山东 济南 250101)

摘要: 移动应用和服务器交互的接口大多都采用 Web API 进行通信, 但这些移动应用所引入的 Web API 会带来一些新的安全问题。为方便研究这些 Web API 的安全问题, 在常规的 Android 程序测试框架的基础上, 设计并实现了一个自动发现 APK 中面向服务器端 Web API 接口的系统, 该系统有助于开展对服务器端私有 Web API 接口方面的安全研究。

关键词: Web API; 安卓应用; 静态分析; 动态分析

中图分类号: TP393

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2016.00134

Toward discovering and exploiting private server-side Web API

CHEN Jia¹, GUO Shan-qing^{1,2}

(1. School of Computer Science and Technology, Shandong University, Jinan 250101, China;

2. Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250101, China)

Abstract: Most of the interfaces for mobile application and server interaction use the Web API for communication, but the Web API introduced by these mobile applications may introduce new security issues. To facilitate the study of the security of Web API, a system for automatically discovering the server-side Web API interface in APK files based on the conventional Android program testing framework was designed and implemented. This system can help to develop the research on private server-side Web API interface security.

Key words: Web API, Android App, static analysis, dynamic analysis

1 引言

在个人计算机时代, 用户通常使用一个浏览器, 如 Internet Explorer、Chrome 或者 Firefox 去浏览不同的网站。浏览器通常提供一个地址栏, 让用户输入网址以便于访问与网址对应的网络资源或服务。然而, 在小屏幕且没有实体键盘的移动设备(如手机或平板)中使用浏览器是非常不方便的。并且手机的流行将大量用户的日常事务处理、娱乐等从计算机转移至手机上进行操作^[1]。

因此越来越多的网络服务提供者都提供了自己的手机应用来让用户更为便捷地使用自己的服务, 从而提高用户体验来吸引更多用户。

应用程序通常都会使用系统网络框架去调用服务器提供的 Web 应用程序接口(API, application programming interface), 在安卓平台中通常使用的就是 WebView 控件和 DefaultHttp Client 组件。当服务商将自己的程序发布到应用市场后, 他们错误地认为来自这些 Web API 的请求都是可信的。Web API 通常被设计为只能

收稿日期: 2016-11-12; 修回日期: 2015-12-05。通信作者: 郭山清, guoshanqing@sdu.edu.cn

基金项目: 国家自然科学基金资助项目(No.91546203); 山东省重点研发计划基金资助项目(No.2015GGE27033); 山东省自主创新及成果转化专项基金资助项目(No.2014CGZH1106); 山东省自然科学基金面上资助项目(No.ZR2014FM020)

Foundation Items: The National Natural Science Foundation of China (No. 91546203), The Key Science Technology Project of Shandong Province (No. 2015GGE27033), The Independent Innovation Foundation of Shandong Province (No. 2014CGZH1106), The Natural Science Foundation of Shandong Provincial (No. ZR2014FM020)

被自己的应用私有调用。然而服务器端程序却很难验证调用 Web API 的来源是否真的是自己的合法应用。实践论证了这是一个糟糕的安全实现。攻击者不仅能够获取到这些 API 接口, 同时也能创建恶意的请求或者发送原始应用所发起的被篡改后的请求。目前在移动端, 应用的使用占据了 80% 的用户使用时间而浏览器只占用了 20%^[1]。如果本文的研究能够让开发者认识到移动端的私有 Web API 应该得到和原来通过浏览器访问的常规网络接口同样等级的安全检查和验证, 那这就是一个很有必要的研究。

为了回答这个问题, 第一步就是去发现这些私有 API 的网址和参数。由于这些 API 都是私有的, 无法通过查阅文档来获取这些数据。所以设计了一个通过函数钩子去监听安卓系统中有关于 HTTP 请求和响应的相关方法, 来自动化地从应用中发现和获取相关的私有 API。随后设计了一个新的系统, 这个系统首先使用静态分析技术从应用的安装文件中找到那些含有调用网络接口操作的 Activity。然后系统将会在定制的安卓系统中运行这些 Activities, 并且监听他们的操作。定制的安装系统能够记录 HTTP 请求和响应信息, 如网址、参数、HTTP header 等。通过分析这些 HTTP 的请求和响应, 能够检查出它们是否存在和网络相关的攻击漏洞, 如会话劫持、API 接口被滥用等。虽然使用了安卓应用和平台去验证该想法是可行的, 但是这些方法和思路同样能被应用到 iOS 平台或其他手机系统中。在其他平台中发现私有 API 漏洞也是不可知的问题。

为了评估本文的方法, 从 Google Play 中免费版前 120 名的应用中选取了 76 个应用作为测试集合。此系统成功运行了 48 个应用, 并且在 30 多个程序中发现了私有 API 接口。手动分析了这些被发现的私有 API 接口, 找出了 9 个应用中的接口是存在可以被攻击的漏洞, 如被会话劫持和接口滥用。同时也测试了国内应用市场上的部分应用程序, 也发现了许多存在同类问题的程序。

本文的贡献有以下两点。

1) 设计并且实现了一个能够自动化处理和发现安卓应用中私有 Web API 的系统。

2) 用 Google Play 上最流行的 76 个应用来测试此系统。深入调查表明, 私有的 Web API 所暴露出来的漏洞是可以被利用的。

2 背景和实例

2.1 背景知识

1) Web API

一个服务端 Web API 是一个编程接口, 包括一个或多个公共的在网络中暴露出的接口。一个 Web API 被一个 URI 唯一标识, 它可以成为一个标准的 HTTP 方法, 如 GET 请求、POST 请求、PUT 请求、它们的响应信息通常是 JSON 或者 XML 格式。

2) 安卓框架中的 HTTP 客户端

一个安卓应用需要获得网络请求许可 (Android permission internet) 后才能进行网络操作。安卓系统框架提供了 2 种基本的 HTTP 客户端给开发者进行使用。它们分别是 APACHE HTTP client 和 HttpURLConnection。DefaultHttpClient 和 AndroidHttpClient 继承于 APACHE HTTP client。对比两者, HttpURLConnection 有更轻量级的代码和更好的可用性, 所以在 Android2.3 版本以后, HttpURLConnection 的使用率不断提高。

3) 异步操作 (asynoc task)

网络操作通常存在延迟, 这种情况会阻塞线程。所以为了避免用户界面停止响应或者响应时间过久, 安卓提供了异步操作的类, 强制要求所有的网络请求都需要异步实现, 而不能直接在主界面进程中进行网络请求操作, 以此保证流畅的用户体验。异步操作类运行在单独的线程上, 并且提供了 onPreExecute、doInBackground 和 onPostExecute 等虚拟方法来给开发者进行对应操作的实现。

2.2 实例分析

优酷网是中国最流行的在线视频网站之一。根据 Alexa2016 的报告, 优酷网被访问排名是全球第 110 名。同时优酷的安卓应用在 Google Play 上被下载了超过 5 000 万次。

如图 1 所示, 在优酷网页注册账号的页面, 需要输入验证码后才能进行注册。然而, 在优酷的手机客户端, 注册账号的时候并没有被要求输

入验证码。如果攻击者获取到优酷手机端使用的私有注册接口的话，他们就能绕过这一层验证码机制。同时能够看到网页版和手机端的界面都需要手机短信验证码，然而这样的限制并不能区分出注册的用户是人还是机器。攻击者可以使用短信验证码平台提供的接口来自动获取手机号和对应的短信验证码去注册僵尸账号。



(a) 网页版



(b) 手机客户端

图 1 优酷用户注册页面

同时，这个私有的注册接口还能被攻击者用于发送垃圾短信。获取验证码的网页端接口通常有 IP 地址限制和时间限制来防止不停地发送短信。经过多次请求之后，2 次连续请求的时间间隔会变长。但是手机端的发送验证码接口却没有应用这些限制，这就让攻击者去利用该接口发送垃圾验证码骚扰短信成为可能。

3 系统实现方法

本文设计了一个包含混合分析的移动应用的

自动化发现私有 Web API 的系统来验证此方法。通过系统发现的 Web API 将会帮助人们理解一个应用程序的功能和工作原理以及其对安全审计的态度。

3.1 系统架构和挑战

需要获取到私有 API 接口的网址和参数才能够恢复或重现私有 API。在分析安卓程序的时候，静态分析方法和动态分析方法都被广泛地使用^[2-4]。静态分析方法和动态分析方法相比有更好的速度和性能。然而 API 的网址和参数通常是由多个字段组合而成，静态地去模拟拼接过程效率低并且成功率低。同时开发者很少会把所有的 Web API 接口的网址或参数直接写在代码中，许多程序都是在运行的时候通过访问其服务器来获取最新的私有 Web API 列表，这样能保证其程序使用的接口是最新且安全的。所以在这种情况下加入了动态分析的方法。

动态分析安卓应用面临 2 个挑战：一是如何模拟应用程序的界面交互，二是如何在交互的时候获取到 Web API 的网址和参数信息。对于第二个挑战通常的做法是破坏 SSL 证书链，通过设立中间代理来获取^[2,5,6]。然而如果 Web API 采用加密的连接方式，那么代理就无法获取到连接中的有用信息，如网址或参数信息。

如图 2 所示，此系统架构融合了静态分析方法和动态分析方法。在静态分析环节中，系统进行了 5 步操作。①分析出哪些 Activities 含有调用 Web API 的操作。②对目标的 Activities 建立函数调用图。③创建网络调用树。④对 Smali 代码进行搜索网址并验证。⑤修改 Manifest 文件。完成这些操作以后，对应用程序的安装包进行重打包，使之能被系统通过命令直接唤起对应的 Activity。在动态分析阶段，系统会结合静态分析的数据，通过 UI 分析等操作，动态地生成测试用例并编译为测试 APK 文件，并和重打包的 APK 文件同时在修改的 Android 系统上允许记录关于私有 Web API 的信息。为了解决在动态分析中遇到的第一个挑战，通过界面中每个元素的状态和模型转换关系来重现每一个界面的状态；对于第二个挑战，在应用程序调用系统函数发送网络请求之前就进行了中断获取其将要发送的

信息。

3.2 识别目标 Activity

系统首先需要识别出哪些 Activities 调用了 Web API 的网络请求操作。正如第2节所述, 安卓提供了3种类型的 HTTP 客户端来给开发者使用。其中2种是由 Apache HTTP client 继承而来, 另一种是 HttpURLConnection。为了发现这些类的使用情况, 系统反编译应用程序的安装包, 获取反编译结果的 Smali 代码文件来识别这些 HTTP 客户端的实例。图3显示了通过这些客户端进行 HTTP 请求所对应的 Smali 代码。

在这3种客户端中, HttpClient 在 Adnroid5.0 版本前一直是系统自带依赖库, 并且 HttpURLConnection 是 Google 推荐开发者使用的联网方式。它们被广泛用于安卓应用的开发, 甚至许多优化的第三方网络类库的底层也是基于它们实现的。例如 Volley^[7], 一个 Google 开源的网络连接框架, 在 Android2.3 系统版本前使用 HttpClient 来实现底层网络请求, 在该版本后的系统都使用了

HttpURLConnection 来实现底层网络请求。所以, 本文的方法依然能够发现使用这些方法的第三方提供的联网框架的应用网络的使用情况。

3.3 为目标 Activities 建立函数调用图

系统会为每个目标的 Activity 都建立函数调用图。通过分析 Smali 代码中的 invoke 指令来构建各个类之间的调用关系。如果方法 A 调用了方法 B, 系统会在函数调用图中的方法 A 和方法 B 之间构建一条由 A 指向 B 的有向边。由于安卓系统存在回调机制, 所以函数的调用关系并不总是明确地直接调用。仅仅关注直接的调用机制可能会忽略很多用户定义的事件处理, 如继承监听类的操作, 在异步类中调用的网络操作等。为了解决这些非直接调用关系, 系统特别处理了两大类调用: ①在安卓事件监听器类的函数中调用的方法; ②在异步类中被调用的方法。

1) 事件监听类

在安卓应用中, 开发者定义了许多继承于事件监听类的事件的处理方法。这些方法多在匿名类、

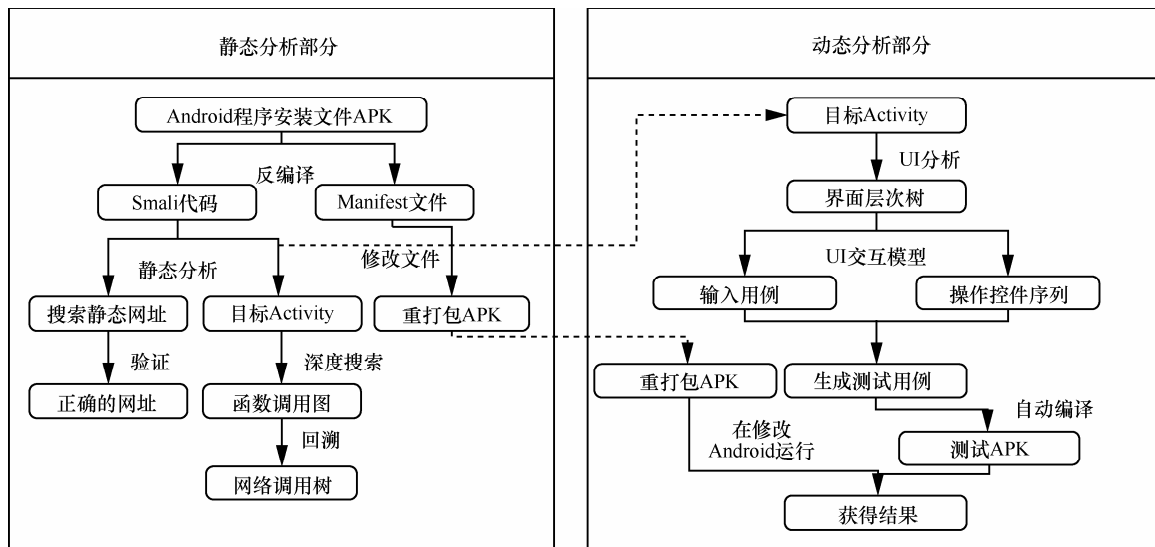


图2 系统架构

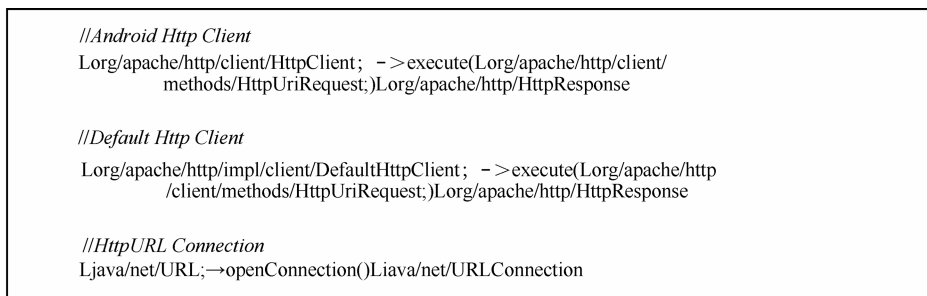


图3 在 Smali 代码中, 3 种网络客户端发送请求所对应的代码

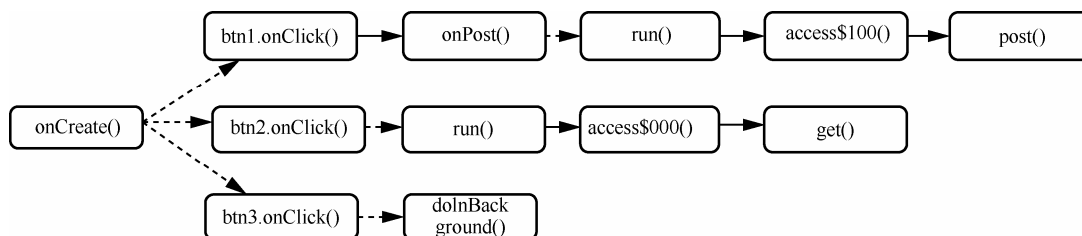


图 4 由 Smali 代码生成的网络调用树结构

```

MainActivity$1:
...
# annotations
.annotation system Ldalvik/annotation/EnclosingMethod;
value = Lcom/example/urldemorunable/MainActivity;->onCreate(Landroid/os/Bundle;)V
.end annotation
...
# virtual methods
.method public onClick(Landroid/view/View;)V
...
invoke-direct {v1, p0}, Lcom/example/urldemorunable/MainActivity$1$1;-><init>(Lcom/example/urldemorunable/ MainActivity$1$1;)V
...

MainActivity$1$1:
...
# annotations
.annotation system Ldalvik/annotation/EnclosingMethod;
value = Lcom/example/urldemorunable/MainActivity$1;->onClick(Landroid/view/View;)V
.end annotation
...
# virtual methods
.method public run()V
...
# invokes: Lcom/example/urldemorunable/MainActivity;->get(Ljava/lang/String;)Ljava/lang/String;
invoke-static {v1}, Lcom/example/urldemorunable/MainActivity;->access$0(Ljava/lang/String;)Ljava/lang/String;
...

MainActivity:
.method static synthetic access$0(Ljava/lang/String;)Ljava/lang/String;
...
invoke-static {p0}, Lcom/example/urldemorunable/MainActivity;->get(Ljava/lang/String;)Ljava/lang/String;
...
.end method

.method private static get(Ljava/lang/String;)Ljava/lang/String;
...
invoke-interface {v1, v6}, Lorg/apache/http/client/HttpClient;->execute(Lorg/apache/http/client/methods/HttpRequest;) Lorg/
apache/http/HttpResponse;

```

图 5 Smali 代码片段

用户界面进行解析，获取其界面中的元素。随后系统利用这些信息来模拟 UI 交互。

1) 界面控件解析

Android 应用是事件驱动的，且很多重要的操作都是由界面交互触发的。因此对每一个目标的 Activity，系统首先分析它的 UI 界面，提取其中的 UI 元素。系统在动态运行的时候分析每个 Activity，这样可以获取静态和动态生成的界面元

素，使用 UIAutomator 去获取 UI 的层次结构。然后将获取到的信息存储为和 Activity 同名的 XML 格式的文件。目标 Activity 中所有的元素信息都包含在通常以 LinearLayout 等容器组件开头的 XML 层次结构信息文件中。关注于包含开发者自定义信息的控件属性，如控件的“resource-id”以应用程序的包名或者 activity 名称（resource-id="com.example.click:id/button1"）。最后，将在解析

得到的映射表中匹配静态分析中得到的可能调用网络请求的可执行控件。

2) UI 交互模拟

UI 交互模拟分 2 个步骤来实现。

①确定输入类型。系统会根据控件的属性来生成测试数据。例如，如果一个控件有提示信息(hint)，那么系统就会根据提示信息来判断数据类型。如果一个 EditText 在一个 TextView 控件之后，那么这个 TextView 的文字就可能会告诉用户应该在这个 EditText 输入什么。同时还会根据编程人员对该控件的命名和其他输入类型的限制来分析出最为合适的输入用例，并将这些控件和对应的输入用例保存在系统中以便使用。

②UI 状态的记录和切换。使用每一个可执行的控件状态去记录每一个 UI 界面的状态。每一个可执行的控件都有 2 种状态：已单击与未单击。初始化时，每个控件的状态都是未单击过的。当获取到一个界面所有输入框的输入类型后，就会在开始模拟单击之前先用测试数据将所有的输入框都填上对应的数据。然后，开始单击每个可执行的控件去触发该控件所绑定的事件监听方法。在单击一个控件后，该控件的状态就变为已单击状态，同时会判定当前显示的 Activity 的名字是否是单击之前的 Activity 的名字，如果不一样说明该控件触发了界面切换，那么在这种情况下系统就会在单击下一个可执行控件之前先触发一个返回操作，让程序回到测试的 Activity 界面。这样的模拟交互持续到所有的可执行控件的状态都是已单击状态。算法 1 展示了 UI 界面交互模拟的过程。通过这一模型，就能生成对应的测试用例代码并且将其编译成一个 APK 安装包。

算法 1 自动 UI 交互处理

输入：转换的 JSON 文件

可执行控件列表 L

当前 Activity 的名字 N

输出：所有触发 HTTP 请求的 UI 状态

function UI INTERACTION

for all $w \in L$ do

$w.status = false$

end for

用获取到的数据填充所有的 EditText

while do $\exists w \text{ st. } w.status = false$

单击 W 去触发其监听事件

$w.status = true$

获得当前 activity 的名字 N0

if $N \neq N0$ then

触发返回事件

end if

end while

end function

3) Fragment 处理

当程序界面中使用了 Fragment，那么当触发某个按钮事件后，会导致界面内容大面积被更换。但这并不是 Activity 和 Activity 之间的切换，无法使用触发返回事件回到切换前的页面。这样就会导致之前页面的后续按钮无法单击，而新 Fragment 中的元素不在列表。如果不处理的话，程序会按照之前页面未单击的按钮坐标在新的界面上去单击。这样是不正确的。因此，在静态部分和动态部分分别进行处理来解决这个问题。

①静态部分。在构造好网络调用树后，系统会自动对所有的 Activity 进行搜索，观察期代码中是否有调用 Fragment 或者含有切换 Fragment 的代码，如果存在这类代码，将会提取出其对应的键值对关系(<Activity Name, Fragment Name>)。

②动态部分。在每次单击操作后，根据 FragmentManager 所提供的日志信息判定该次单击是否造成 Fragment 视图的切换。如果存在 Fragment 的切换，那么系统会把这个操作保存到一个额外队列中，该队列中存储了遍历所有目标 Activity 后还应该再次遍历 Fragment 的状态信息。在保存好信息后，重新启动测试的应用，按照操作序列到达切换前的状态，直接进行后续的事件触发，从而越过 Fragment 切换来保证测试的完整性。

3.8 在修改的 Android 上执行重打包安装程序

最后一步就是在修改过的 Android 系统中执行重打包的 APK 安装程序。修改了的 Android 系统能够自动记录网络连接信息。这一步通过修改系统中执行 HTTP 请求函数所调用的方法。修改这些函数，添加 Logger 类来记录包块 URI、请求头部和请求内容等信息。对比其他获取网络信息

的方法（如代理），本文的方法有以下几个优点。

1) 在 HTTP 请求发送前就能够获取到其数据，这时候网址和参数信息都已经被程序构建结束，但是还没有被混淆或者伪装。而像 Burp Suite^[12]等工具是在 HTTP 请求发送后才拦截。在这个时候请求的数据可能已经被分段或者加密（如使用 VPN）。2) 不需要构建虚假的证书来对 HTTPS 做中间人攻击以获取 HTTPS 的请求数据。3) 一些程序（如 com.creditkarma.mobile）检测到系统使用代理后就会自动停止运行来保证其数据安全。

因为本文主要关注 HTTP 的请求，所以本文修改了 HttpClient 和 HttpURLConnection 的相关函数。对于 HttpClient，修改了 httpClient.execute() 和 defaultHttpClient.execute() 等方法。对于 HttpURLConnection，修改了 URL.openConnection() 的实现类并且追踪其参数传递过程中的方法。在每一步的 UI 交互都会收集系统记录、交互产生的网络请求信息队列。

3.7 节提到了 Fragment 这个问题的处理，其中为了获取 FragmentManager 切换时其各个组件的状态以及加载、删除、替换时所传入的 Fragment 参数的类名，修改了 /frameworks/base/core/java/android/app/FragmentManager.java 中对应的方法，使之能够在进行这些操作的时候输出对应的信息。这样就能根据对应的信息来判定是否存在 Fragment 的切换。

4 实验和发现

本文从 Google Play 市场的畅销排行榜前 120 的程序中选择了非游戏的 76 个应用作为测试集。没有选择游戏程序是因为大部分的游戏都使用如 Unity 等游戏框架，这些框架的网络连接不同于正常系统中实现的网络连接。本文的自动化系统成功运行了 48 个应用程序，并且有超过 30 个程序被恢复出私有 Web API 信息。另一些程序失败是由于反编译失败或者使用了自定义框架导致的重打包失败。进一步分析恢复出的私有 Web API 接口，手动测试这些接口发现 9 个应用接口存在可能被利用的漏洞，如 API 滥用和会话劫持。同时也获取到了一些 WebView 的网址。本文对国内安卓市场上比较流行的程序进行了同

样的测试，发现这些问题也在许多应用中存在。

下文给出发现的漏洞例子。为了更好地说明，本文对比了应用程序的 Web API 和其浏览器端程序服务的接口。

4.1 API 滥用

由于现在安全问题往往会导致比较严重的后果，所以越来越多的应用已经对他们的 API 接口采取了多种保护措施。例如，一部分应用使用 HTTPS 代替不安全的 HTTP，使用签名技术来保证它们传递的参数不受篡改。从实验结果观察发现：TextNow、Flipagram、Smule 和 lionmobile: battery 使用了签名技术保证它们的参数不被篡改。它们中的一些程序同时也在其所有的 API 上使用了 HTTPS。Dubsmash 甚至使用了 HTTPS+MAC 来保护他们的接口和信息。

然而，本文也找到一些程序并没有成功保护他们的 API，从而导致了 API 滥用。本文总结了发现的问题并且按照它们的功能进行了分类说明。具体如下。

1) 注册接口

僵尸账号一直以来都是威胁网络安全的问题。因此它们通常采用验证机制来阻止机器注册。但是一些企业为了让用户在手机上使用便利，在手机应用中移除验证机制，如验证码或授权码。此外还有一些手机应用需要电子邮箱地址作为验证，但是他们并没有阻止用户在单击验证邮箱所有权信息之前在该应用上进行其他操作或者请求。Wallapop 和 Flipagram 这 2 个应用在保护它们注册接口存在的弱点。虽然它们都采用了 HTTPS 连接来保护网络数据传输过程中的数据安全，但是没有对参数进行签名，这导致该接口能够被重放攻击，并且他们也没有对邮箱地址的归属权进行验证。在 Wallapop 的注册接口，仅修改邮箱地址的参数键值就能够成功注册一个僵尸账号。而在 Flipagram 应用中，需要同时修改邮箱地址和用户名 2 个参数来完成一个僵尸账号的注册。

由于移动端应用和浏览器端应用有不同的接口，所以企业在升级应用的时候会出现接口限制不匹配的情况。2.2 节提到优酷的移动端程序没有提供验证码作为验证，进一步发现其浏览器端程序已

经禁止了用户使用邮箱注册新用户，阻止僵尸账号问题（如图 1(a)所示）。但是其移动端应用仍然允许用户使用邮箱进行注册（如图 1(b)所示）。

优酷手机客户端中，注册的 API 接口中网址和参数是在“getRegistURL”方法中被构造的。该方法在 com.youku.phone/smali/com/youku/http/URL Container.smali 文件中并被命名为“com.youku.ui.activity.HomePageActivity”的 Activity 调用。通过对比参数和浏览器端应用接口的参数发现验证码参数被从接口移除（如图 6 所示）。基于这些 API 接口的信息写了一个 Python 的脚本，通过提供虚假的邮箱地址和密码成功注册了一些账号。

passport:"15529853210"
Password:"234456"
cptcha:"kskxh"
smscode:"134512"
Callback:"signupcallback_1 437 204 144 651"
from:"http%3A%2F%2Flogin.youku.com%2"
wintype:"page"

图 6 Web 端接口注册需要的参数

2) 登录接口

对于登录接口，通常软件为了安全，都会通过限制用户在一定时间内尝试错误登录的次数来防止暴力破解密码。但是一些软件并没有做到这一点。从恢复出来的优酷登录 API 接口发现（如图 7 所示），在登录的过程中并没有要求验证码保护。通过网络调用树中查找登录接口相关的函数调用，很容易就获取了登录接口中每个参数的含义。在随后的测试中，发现该接口并没有对同一 IP 地址的登录失败次数做任何的限制。这样让使用包含<用户名，密码>的密码本（从其他网站泄露的用户密码本数据库）进行撞库攻击，使暴力破解账户的攻击成为可能。

POST request to /user/passport/login		
Type	Name	
URL	pid	0865e0628a79dfbb
URL	guid	d9d425ab4f240e3a60e39a769 107a818
URL	mac	08:00:27:3f:cf:3d
URL	imei	000000000000000
URL	ver	4.6
URL	_t	1437491615
URL	e	md5
URL	_s	7d9cd6bacb1f72 871 636fb726 478b695
URL	operator	Android_310260
URL	network	Wi-Fi
URL	uname	abc:@xyz.com
URL	pwd	39d8f4781b6 146 372 038cb956e2 360b8

图 7 优酷移动端登录接口

3) 发送短信验证码接口

在实验中，发现很多应用都采用了短信验证码的机制验证用户身份。但是他们没有很好地对该接口做安全防护，导致滥用该接口给用户发送大量验证码。其中一个例子是，利用卷皮（一个电商应用）暴露的发送验证码短信的接口写了一个 Python 脚本去发送短信。他们的接口限制了一个用户 60 s 内只能发送一条验证码。本文设置了一个从 182****2200 到 182****2300 的 100 个号码的号码池，脚本自动按照顺序从号码池中选择号码，将其填入该接口手机号的参数中，并给他们的服务器发送请求。图 8 展示了成功发送短信的信息。

这类攻击不仅会骚扰用户（短信号码 182****2200 到 182****2300 用户会收到验证码短信），同时也会给企业带来额外的短信发送平台的费用开销。

4) 检查账户是否存在接口

当创建一个新账户的时候，移动应用为防止重复注册，通常会检查该邮箱地址是否已经注册过账号。当用户使用忘记密码的功能时，

```
18253162286 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162286"}}
18253162287 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162287"}}
18253162288 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162288"}}
18253162289 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162289"}}
18253162290 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162290"}}
18253162291 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162291"}}
18253162292 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162292"}}
18253162293 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162293"}}
18253162294 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162294"}}
18253162295 {"code":1000,"info":"\u9a8c\u8bc1\u7801\u53d1\u9001\u6210\u529f","data":{"mobile":"18253162295"}}
```

图 8 通过短信验证码接口成功地发送信息

通常需要输入邮箱地址进行查找, 移动应用就需要知道用户填写的邮箱地址是否存在已经注册的账号。调查显示, 很多的程序对检查账号是否存在接口没有做安全防护。例如, 在 Weather Channel 应用中的忘记密码 API 接口和 Flipagram 应用中检查邮箱接口能够被用来测试一个给定的账号是否在该应用中存在。广告商能通过这些接口检查邮件地址列表的用户是否活跃或可以根据他们注册的应用推送特定的广告邮件。

5) 获取内容接口

许多应用提供有用的信息来获取巨大的用户数量, 从而从付费用户或广告中获取收益。例如, The Weather Channel 应用提供了精确的天气信息来吸引用户, 从而他们能够从 Google AD 上获取收益。但是从本文的分析来看, 获取天气数据的核心 API 接口是没有加以任何保护的。笔者通过其接口能够获取任意地方的天气信息而不需要打开他们的应用, 只需要修改“geocode”的值, 该数值是由 Google Map 定义的, 可以直接被查询。另一个例子是 Zedge 应用, 其提供不同手机的壁纸下载服务。但是从他的私有 API 接口中, 能够轻易获取壁纸的详细下载地址并且自动下载。同样的问题也存在于 Cute Emoji Keyboard 应用的列出动态图片接口。该接口提供详细动态图片资源信息的 JSON 格式, 数据能够直接被其他程序解析, 并且下载服务器上存在的动态图片资源。

4.2 会话劫持

为了更好地理解应用程序在用户登录后的行为, 在系统测试应用之前手动地使用正确的用户名和密码登录应用。从实验中发现, 移动应用也存储他们的会话信息在 cookie 文件中或在网络请求的参数中。

当分析优酷手机应用的时候, 发现许多 API 都带有一个特殊的名为“ytkk”的 cookies 参数。因此, 检查了优酷浏览器端程序并且确定了该字段是用于保持 session 连接的。进一步找到了保存接口参数“ytkk”值的类和该类所在的位置。根据这些信息设计了 2 种会话劫持攻击。一是将移动端接口传递的“ytkk”值复制到浏览器应用的对应 cookies 文件中, 成功地在网页端登录。

二是把这个值插入到移动端的 Sharedpreferences 文件中 (位置是/data/data/com.youku.phone/shared_prefs/com.youku.phone.preferences.xml)。当再次启动移动应用时, 则发现应用成功登录了该账号。同样的攻击也在百度应用中重现, cookies 存储的键值是“BDUSS”。

许多应用保存会话信息在其请求参数中作为另一种保持会话信息的方法。一些应用使用签名保护会话信息不被篡改, 如 TextNow 程序使用 HTTPS 和参数签名保护会话信息。然而有些应用, 如 Smule:Sing!, 仅仅在登录和注册使用了 HTTPS, 其他接口依然使用的是 HTTP。在这种情况下, 会话信息能够通过存储在被攻击手机的“/data/data/com.smule.singandroid/shared_prefs/network.xml”值被劫持。另一个应用 Flipagram, 存储会话信息在 HTTP 的请求头部, 但是没有对会话信息做签名, 导致会话被劫持。

4.3 WebView 网址重用

在静态分析代码的时候发现一些程序将 WebView 的入口地址固定在代码中。而这样的代码很容易导致程序内部信息的泄露。很多程序或多或少地使用了 WebView 来加载动态页面以获取更好的展示效果, 但是其中有些程序, 如 com.kingkr.kykyfkb, 没有限制访问网站的 User-Agent 或者其他限制从非其程序的 WebView 访问该网址的方法, 以至于人们可以直接通过计算机端浏览器访问该页面, 从而获取整个程序的所有网页部分的逻辑。并且通过分析其网页, 发现了网页调用 Android 的 JavaScript 接口的通信信息, 从而获取会话的 cookies 信息保存位置, 并成功地进行会话劫持。另一方面, 如果 WebView 的网页没有采用 HTTPS 的加密方式, 就会导致传输的 JavaScript 文件代码被修改, 这样中间人能够加入特定的 JavaScript 脚本来获取应用上的一些信息。

5 系统局限

本文系统采用了联合静态分析和动态分析的混合策略。在静态分析部分, 通过查找图 3 的 Smali 代码来过滤目标 Activity。因此不能处理使用不是基于 HttpClient 或者 Http URLConnection 的自定义网络框架。除此之外, 不是所有的网络

连接都是使用 HTTP。例如, 推送服务通常采用 TCP 连接来保持和服务器的长连接, 从而保证能够及时地收到服务器的推送信息。一些游戏或者聊天服务也是使用 TCP 连接来代替 HTTP 以保证低延迟。本文系统只关注 HTTP 请求所以不能处理这些情况。

在动态分析部分, 为了保证目标 Activity 能够被启动, 修改了应用的 Manifest 文件, 使这些 Activities 的属性变为“export”而能被启动。然而在动态运行时, 有些 Activities 需要被其他的 Activities 传递正确的 Intent 参数才能被正常启动。除此之外, 一些 Activities 需要提供正确的用户名和密码才能被启动, 本文的系统并不能处理这些情况。

6 相关工作

6.1 移动应用漏洞的发现

在过去的几年, 有很多工作都关注于在移动应用中发现不同种类的漏洞。例如, TaintDroid^[4]通过跟踪信息流来探测隐私泄露的漏洞, PiOS^[13]使用静态分析在 iOS 平台上探测同样的问题。Chex^[3]通过使用静态分析技术来探测安卓应用中的组件劫持。FlowDroid^[14]提供了一个高精度的对安卓应用生命周期的分析。在这些工作^[15~18]中讨论了 WebView 的攻击, 包括 JavaScript 注入、事件监听和劫持。他们说明了 Webview hook 方法和“addJavascriptInterface”接口的广泛使用。这些都是用于攻击 WebView。SMV-HUNTER^[2]通过混合的分析技术来探测 SSL/TSL 的中间人攻击漏洞, 然而很少有人关注于应用服务器端的漏洞。文献[19]给出一个严重的安全漏洞, 如果开发者没有很好地处理安全检查而导致密码被暴力破解等问题。

6.2 自动化的 UI 交互

自动化的 UI 交互是自动测试 Android 应用的重要步骤。文献[11]给出了一个系统化的自动测试输入生成方法, 并且在代码覆盖、失败检查、多平台可用性和简易使用 4 个维度进行了评价。

事件(包括 UI 事件和系统事件)是按照随机或者系统探测策略来生成的。随机的测试输入生成法^[20,21]能够有效地生成事件, 因此他们适用于

压力测试。但是他们通常生成许多冗余的测试用例, 因为他们不能感知需要多少行为才能被覆盖。许多工作都采用了系统策略来生成。基于 Activity 的状态和事件的有限状态机模型来作为切换^[8,9]; 基于探索策略的模型会有更有效率的结果。但是这些工具可能丢失一些事件, 如 GUI 改变的时候通常会认为这是一个新的状态而无法准确判定。另一些工具采用了更复杂的技术, 如符号执行和进化算法来达到一个更好的覆盖率^[22,23]。

7 结束语

为了实现对于给定的移动应用, 检查其服务器端的 API 和其之前公开的网页服务是否有相同的安全防护措施的需要。本文设计并实现了一个系统, 该系统能够自动发现并处理私有的 Web API。并尝试利用该系统分析从 Google Play 中最畅销的 120 个程序中选择了 76 个应用程序及国内安卓市场上畅销的程序, 成功发现了许多 API。实验证明了被发现的服务器端的私有 API 的漏洞是可以被利用的。

参考文献:

- [1] BOSOMWORTH D. Mobile marketing statistics compilation. [EB/OL]. <http://www.smartinsights.com/mobile-marketing/mobile-marketinganalytics/mobile-marketing-statistics/>.
- [2] SOUNTHIRARAJ D, SAHS J, GREENWOOD, et al. SMV-HUNTER: large scale, automated detection of SSL/TLS man-in-the-middle vulnerabilities in Android App[C]//The 21st Annual Network and Distributed System Security Symposium.
- [3] LU L, LI Z, WU Z, et al. Chex: statically vetting android apps for component hijacking vulnerabilities[C]//The 2012 ACM conference on Computer and communications security. 2012: 229-240.
- [4] ENCK W, GILBERT P, CHUN B G, et al. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones[C]//The 9th Usenix Conference on Operating Systems Design and Implementation. 2016: 1-6.
- [5] FAHL S, HARBACH M, MUDERS T, et al. Why eve and mallory love Android: an analysis of Android SSL (in) security[C]//The 2012 ACM Conference on Computer and Communications Security. 2012: 50-61.
- [6] CAI F, CHEN H, WU Y, et al. Appcracker: widespread vulnerabilities in user and session authentication in mobile Apps[C]//Mobile Security Technologies Workshop. 2015.
- [7] DEVELOPERS A. Transmitting network data using volley[EB/OL]. <http://developer.android.com/training/volley/index.html>.
- [8] YANG W, PRASAD M R, XIE T. A grey-box approach for auto-

- mated GUI-model generation of mobile applications[C]//In Fundamental Approaches to Software Engineering. 2013: 250-265.
- [9] AMALFITANO D, FASOLINO A R, TRAMONTANA P, et al. Using GUI ripping for automated testing of Android applications[C]//The 27th IEEE/ACM International Conference on Automated Software Engineering. 2012: 258-261.
- [10] HAO S, LIU B, NATH S, et al. PUMA: programmable UI-automation for large-scale dynamic analysis of mobile Apps[C]//The 12th Annual International Conference on Mobile Systems, Applications, and Services. 2014: 204-217.
- [11] CHOUDHARY S R, GORLA A, ORSO A. Automated test input generation for Android: are we there yet?[C]//ArXiv Preprint ArXiv: 1503. 2015.
- [12] WEB SECURITY P. Burp suite [EB/OL]. [https:// portswigger.net/burp/](https://portswigger.net/burp/).
- [13] EGELE M, KRUEGEL C, KIRDA E, et al. PiOS: detecting privacy leaks in iOS applications[C]//Network and Distributed System Security Symposium (NDSS). 2011.
- [14] ARZT S, RASTHOFFER S, FRITZ C, et al. Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android Apps [C]//ACM SIGPLAN Notices. 2014: 259-269.
- [15] WEI F, ROY S, OU X, et al. Amandroid: a precise and general inter-component data flow analysis framework for security vetting of android Apps[C]//The 2014 ACM SigSAC Conference on Computer and Communications Security. 2014: 1329-1341.
- [16] CUI X, WANG J, HUI L C, et al. Wechecker: efficient and precise detection of privilege escalation vulnerabilities in android Apps[C]//The 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks. 2015: 25.
- [17] CUI X, YU D, CHAN P, et al. Cochecker: detecting capability and sensitive data leaks from component chains in android[C]//Information Security and Privacy. 2014: 446-453.
- [18] LUO T, HAO H, DU W, et al. Attacks on WebView in the Android system[C]//The 27th Annual Computer Security Applications Conference. 2011: 343-352.
- [19] CHAOSHUN Z, WUBING W, RUI W, et al. Automatic forgery of cryptographically consistent messages to identify security vulnerabilities in mobile services[C]//The 23rd ISOC Network and Distributed System Security Symposium (NDSS) 2016.
- [20] MACHIRY A, TAHILIANI R, NAIK M Dynodroid: an input generation system for Android Apps[C]//The 9th Joint Meeting on Foundations of Software Engineering. 2013: 224-234.
- [21] [EB/OL] [http://developer.android.com/tools/help/ monkey.html](http://developer.android.com/tools/help/monkey.html).
- [22] ANAND S, NAIK M, HARROLD M J, et al. Automated concolic testing of smartphone Apps[C]//The ACM Sigsoft 20th International Symposium on the Foundations of Software Engineering. 2012: 59.
- [23] MAHMOOD R, MIRZAEI N, MALEK S. EvoDroid: segmented evolutionary testing of android Apps[C]//The 22nd ACM Sigsoft International Symposium on Foundations of Software. 2014: 599-609.

作者简介:



陈佳 (1993-), 男, 福建福州人, 山东大学硕士生, 主要研究方向为移动安全。



郭山清 (1976-), 男, 山东滨州人, 山东大学副教授、博士生导师, 主要研究方向为应用安全、网络安全。