Android 系统点击劫持攻防技术研究

钱正旸[1,2],施 勇[1,2],薛 质[1,2]
(1. 上海交通大学 信息安全工程学院,上海 200240;
2. 上海市信息安全综合管理技术研究重点实验室,上海 200240)

摘 要:点击劫持攻击是近年来出现的一种新型 Web 攻击手段,使用多层透明或不透明的界面欺骗用户点击实现攻击。随着移动互联网的发展和普及,此类攻击逐渐在移动平台中出现,并具有更强的隐蔽性和危害性。文中在总结传统 Web 点击劫持攻击方法的基础上,深入研究了 Android 系统中点击劫持攻击的原理,重点分析了基于通知视图(Toast)的点击劫持攻击(Tapjacking)与基于网页视图(WebView)的点击劫持攻击两种攻击方式的实现方法。由于 X-FRAME-OPTIONS 与 Frame Busting 代码等传统 Web 点击劫持防御方法存在一定局限性,无法有效地防御 Android 系统点击劫持攻击,文中研究了几种针对 Android 系统点击劫持攻击的防御手段,能在一定程度上减缓该类攻击的危害。

关键词:Android 安全;点击劫持;Tapjacking;WebView
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2015)10-0135-05
doi:10.3969/j.issn.1673-629X.2015.10.030

Study of Clickjacking Technology on Android

QIAN Zheng-yang[1,2], SHI Yong[1,2], XUE Zhi[1,2]
(1. College of Information Security and Engineering, Shanghai Jiaotong University, Shanghai 200240, China;
2. Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, Shanghai 200240, China)

Abstract: Clickjacking is a new type of Web attack in recent years. It uses transparent or overlapping interfaces spoofing user clicks. With the development and popularization of Mobile Internet, such attack appears on the mobile platforms, and is more harmful and indetectable. In this paper, based on summarizing the traditional Clickjacking attack on the web, research the theories in depth on Android, mainly analyze Tapjacking and WebView-based Clickjacking. Because the traditional Clickjacking has certain limitations such as X-FRAME-OPTIONS and Frame Busting code, cannot effectively defense Android Clickjacking attack, in this paper study several defense way against Clickjacking, which can slow down the dangers of this kind of attack to a certain extent.

Key words: Android security; Clickjacking; Tapjacking; WebView

0 引 言

近年来,随着移动互联网的普及,移动端网络流量大大增加,已占到全球网络流量的35.3%[1],各类传统 Web 安全漏洞逐渐在移动平台上出现。Android 系统作为市场占有率最高的移动端操作系统,其恶意代码数量出现了指数级的增长。点击劫持漏洞是一个影响范围广、危害大的传统 Web 安全漏洞,也逐渐在移动平台特别是 Android 系统中出现,造成了较大影响。文中研究了几种目前在 Android 系统中较为常见的点击劫持攻击方法,以及可行的防御手段。

1 传统点击劫持漏洞

点击劫持(Clickjacking),又称为“用户界面伪装攻击”(UI Redress Attack),OWASP 对其定义为:攻击者使用多层透明或不透明的界面欺骗用户点击最上层界面的按钮或链接,实际劫持用户访问另一个按钮或链接[2]。使用同样的技术还可以实现键盘输入劫持(Keystrokes Hijack)。

收稿日期:2014-12-10 **修回日期**:2015-03-17 **网络出版时间**:2015-08-26
基金项目:国家自然科学基金资助项目(61332010)
作者简介:钱正旸(1991-),男,硕士研究生,研究方向为 Android 安全;施 勇,讲师,研究方向为网络与信息安全;薛 质,教授,研究方向为信息安全。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20150826.1604.086.html

最早对内嵌框架(iframe)的恶意利用出现在 2002 年[3],2008 年 Robert Hansen 和 Jeremiah Grossman 首次提出点击劫持攻击(Clickjacking)并引起了广泛关注[4]。Paul Stone 在 2010 年 Blackhat 大会上提出了下一代点击劫持攻击方式—拖拽攻击(Drag and Drop)[5],浏览器支持用户拖拽一段文字或一个链接,通常不受同源策略限制,攻击者使用透明的 iframe 诱使用户拖拽出攻击者想要获得的内容,放到另一个攻击者所控制的页面中。通过拖拽攻击技术,攻击者可以绕过针对传统攻击的防御措施,大大增加了点击劫持攻击的危害性。此外,传统的点击劫持攻击还包括:Strokejacking、Likejacking、利用 Flash 与 AJAX 的界面伪装攻击、利用 JavaScript 实现鼠标跟随攻击等[6]。

点击劫持攻击还可以与其他攻击技术相结合,如与 CSRF 结合(2009 年 Twitter 点击劫持攻击)、与 XSS 结合(使反射型 XSS 攻击更容易实施)、与 CSS 结合(利用 CSS 的"pointer-events"属性进行点击劫持攻击)。点击劫持漏洞还可以用于绕过浏览器弹出窗口屏蔽与可缩放矢量图形屏蔽(SVG Masking)等[7]。

2 Android 系统点击劫持攻击

近年来,随着移动平台的普及,特别是 Android 系统的迅猛发展,以往通常出现在 PC 端的点击劫持攻击逐渐在移动平台上出现。移动平台上的点击劫持漏洞最早由 Gustav Rydstedt 等在 2010 年提出并称之为 Tapjacking[8],其主要利用 Android 系统的活动(Activity)与视图(View)进行攻击。此后还出现了基于 WebView 的点击劫持攻击,如 Touchjacking 等[9]。

2.1 Tapjacking

Tapjacking 的原理与传统点击劫持攻击类似,通过多层透明或不透明的界面欺骗用户点击,将用户劫持至恶意链接。Android 浏览器支持实现传统点击劫持攻击所需功能,如 iframe、透明度设置、尺寸缩放等。

Android 系统显示的用户界面元素以活动为单位,每个活动容纳不同的视图,视图是屏幕上的一块矩形区域。通常情况下应用程序显示一个活动并占据整个屏幕,但应用程序也可以只显示视图,如对话框视图(Dialog View)和通知信息视图(Toast View)。对话框视图的作用是使应用程序与用户进行双向互动,如输入文字或点击按键,通常对话框视图尺寸较小,用户可以看到其背后的内容,但不能与其遮挡的内容互动。通知信息视图用于显示一段信息,大小可以由开发者决定,用户可以与其背后的内容互动。

Tapjacking 利用了通知信息视图进行攻击,攻击者在恶意链接或按键上覆盖一层通知信息视图,并在通知信息视图中使用吸引人的图片或按钮引诱用户进行点击,用户实际与隐藏在通知信息视图下的恶意活动进行互动。此外,Tapjacking 也可以通过透明的 iframe 实现,将恶意链接或按键放置在一个透明的 iframe 中,并叠加在一个正常界面之上。在传统点击劫持攻击中,攻击者为保证用户会点击到透明 iframe,通常会在监测到 MouseMove 事件时将 iframe 移动到鼠标位置,但在移动平台中较难实现此方法。因此,在 Android 系统点击劫持攻击中,攻击者通常采用如下代码对透明 iframe 进行缩放以增大用户点击恶意链接或按键的概率。其中"initial-scale"将初始大小设置为最大,使透明 iframe 占满整个屏幕,"user-scalable"设置为 0 阻止用户对界面进行缩放,确保用户能点击到恶意链接或按键。

<meta name="viewpoint" content="width=device-width, initial-scale=10,user-scalable=0">

传统点击劫持攻击通常利用各类浏览器实施,由于 Tapjacking 利用 Android 系统的活动与视图进行攻击,因此不仅可以在 Android 浏览器中实现攻击,也可以在其他应用程序甚至 Android 系统桌面实现攻击,具有更大的危害性。

2.2 基于 WebView 的点击劫持攻击

2.2.1 WebView 类

在移动平台中,浏览器不再是浏览网页内容的唯一途径,Android 系统为应用程序提供了 android.webkit 模块,使应用程序可以直接浏览网页而不需要使用外部浏览器。Android 系统的 WebKit 模块底层使用 WebKit 核心库(WebCore 和 JSCore),上层由 Java 语言封装,其中 Java 层负责与 Android 应用程序进行通信[10]。Java 层一共由 41 个文件组成,其中最重要的是 WebView 类。WebView 类是一个视图类(View Class),主要用于在应用程序中渲染展示 Web 页面,所有需要使用 Web 浏览器功能的 Android 应用程序都需要创建该类以显示和处理网络资源。

2.2.2 WebView API

WebView 类提供了各种应用程序编程接口(API),Android 应用程序通过这些接口使用 WebKit 提供的服务,实现网页浏览等功能。WebView 是专门的 UI 组件,与其他 UI 组件如按键和文本域类似,是更一般的 UI 组件如 View 类的子类,WebView 继承了其父类的 API,因此可以将 WebView API 分为继承 API 与非继承 API。

(1)继承 API:layout、setX、setMinimumHeight、getLeft、getTop、scrollBy、scrollTo 等。Android 应用程序可以使用 layout、setX、setMinimumHeight 等确定位置,也可以调用 getLeft、getTop 检索图案位置,使用 scrollBy、scrollTo 函数为需要滚动条的图案提供支持。se-

tOnKeyListener 函数 Android 应用程序可以注册一个事件处理程序回调函数在有密码输入时启用。requestFocus( )函数可以将焦点移动至用户输入的位置。setBackgroundColor 与 setAlpha 可以调整外观有关的属性,如背景颜色与透明度。

(2)非继承 API:addJavascriptInterface、loadURL、PageDown、PageUp 等。addJavascriptInterface 是 WebView 提供的一种机制,使网页的 JavaScript 代码能够调用 Android 应用程序的 Java 代码,loadURL 则支持 Java 代码调用网页的 JavaScript 代码,PageDown、PageUp 可以实现翻页。

2.2.3　Touchjacking

Touchjacking 基于继承类 API 进行攻击,主要有 WebView 伪装攻击、透明 WebView 攻击、键盘输入劫持攻击三种方式。

(1)WebView 伪装攻击。

该攻击是把两个或多个 WebView 重叠,使其看上去像一个页面。当用户点击页面中的按键或链接时,由于该按键或链接可能属于另一个 WebView 中的网页,用户可能被欺骗访问其他网页,甚至该网页的内容可以被隐藏,用户无法发现已经访问了恶意网页。

以使用两个 WebView 页面的攻击为例,其中一个称为外部 WebView,另一个称为内部 WebView。内部 WebView 中为恶意网页且页面比较小,只占到用户所看到页面的一小部分,这样可以使用户看不到恶意网页的全部内容,减小被用户发现的可能性。恶意应用程序可以只将内部 WebView 中网页的特殊部分(如按键)展示给用户。外部 WebView 页面具有更大的尺寸,使用户能够浏览正常的网页内容。攻击者将内部 WebView 覆盖在外部 WebView 上,由于内部 WebView 很小且没有明显的边界,所以看上去很像外部 WebView 页面的一部分。如果用户浏览了外部 WebView 的内容,点击某个按键或链接,但该按键或链接是属于内部 WebView 的,那么用户可能已访问了恶意内容。

(2)透明 WebView 攻击。

Android 系统允许应用程序将 WebView 中的对象设置为透明。高透明度使 WebView 中的页面很难被看清,甚至完全不可见。在 Android3.0 以上版本中,可以使用 setAlpha 设置 WebView 对象的透明度,每一个本地 Android UI 对象都具有透明度属性,可以由应用程序设置。由于 WebView 类是由 View 类继承的,因此也继承了 View 类的属性。当一个 WebView 对象被设置为完全透明(alpha 属性值为 0),它在视觉上不可见,但实际是存在的,用户仍然可以点击到透明的页面。透明度属性是 UI 组件的通用属性,本身不具有危险性,但被 WebView 继承后,给 WebView 中的网页带来了很大的潜在危险。设置 WebView 透明属性代码如下:

```
WebView mWebView = (WebView) findViewById(R.Id.WebView);
mWebView.setAlpha(0);
```

同样以使用两个 WebView 页面的攻击为例,一个为可见 WebView,另一个为透明 WebView。可见的 WebView 可以载入一个热门网页,吸引用户在页面上进行点击操作。另一个透明的 WebView 载入目标恶意网页,并将透明 WebView 置于可见 WebView 之上。因此,当用户想要点击可见网页上的按键或链接时,实际点击的是透明网页上的恶意按键或链接。

为了实现该攻击,攻击者需要事先计算用户可能进行点击的按键或链接所处的位置。由于可见网页是攻击者选定的,所以攻击者可以较容易地确定按键或链接的位置,并将透明 WebView 中的目标恶意按键或链接放置在对应的位置。

(3)键盘输入劫持攻击。

在前述的两类攻击中,攻击者劫持用户的操作至与用户所见不同的另一个 WebView 中的网页。本攻击方法中,攻击者劫持用户的操作至由恶意应用程序控制的本地 UI 对象(如文本输入框)。例如用户在网页中输入用户名、密码,则可以被攻击者截获。

该攻击基于 WebView 中的 HTML UI 对象与 Android 本地 UI 对象基于同一种 GDI(GraphicsDeviceInterface),且 HTML UI 对象与对应的 Android 本地 UI 对象外观相似。例如,HTML 输入框外观类似于 Android 文本编辑 EditText。因此,若将本地 UI 对象覆盖在对应的 HTML UI 对象之上,用户很难发现其差别。如果用户想对 HTML UI 对象进行操作(如输入文本),实际上用户在对属于攻击者的本地 UI 对象操作。

为了实现该攻击,攻击者需要精确地将本地 UI 对象覆盖在对应的 HTML UI 对象上,具有相同的位置与大小。由于目标网页在很多情况下基本不变(如登入界面),攻击者可以较容易地计算网页中目标 UI 对象的位置和大小。

3　Android 系统点击劫持漏洞防御

3.1　Android 浏览器点击劫持防御

由于可以使用传统点击劫持方法对 Android 系统浏览器进行点击劫持攻击,因此可以使用传统点击劫持攻击的防御方法进行防御。传统点击劫持攻击的主要防御方法有 X-FRAME-OPTIONS[11] 与使用 Frame Busting 代码[9] 等。

(1)X-FRAME-OPTIONS 最先由微软提出并在 IE8 中实施,Firefox3.6、Chrome4 以上版本均支持该功

能。X-FRAME-OPTIONS 是一个 HTTP 头,用以防御利用 iframe 嵌套的点击劫持攻击,可以有三个值:DENY、SAMEORIGIN、ALLOW-FROM。其中若值为 DENY,浏览器拒绝当前页面加载任何 iframe 页面;若值为 SAMEORIGIN,浏览器仅允许加载同源的 iframe 页面;若值为 ALLOW-FROM,可以定义允许加载的 iframe 页面地址。

(2)Frame Busting 是添加一段 JavaScript 代码阻止载入恶意网页,根据文献[12]的研究,传统的 Frame Busting 代码容易被绕过,改进后的 Frame Busting 代码能更好地保障网页的安全。改进后的 Frame Busting 首先判断浏览器是否支持 JavaScript,若不支持则不显示任何内容,若支持则执行 Frame Busting 代码并判断网页是否安全。

传统 Frame Busting 代码:

```
<script>
if (top.location! =location){
top.location=self.location;
}
</script>
```

改进后的 Frame Busting:

```
<head>
<style> body {display:none;} </style>
</head>
<body>
<script>
if(self==top) {
var theBody=document.getElementsByTagName('body')[0];
theBody.style.display="block";
}else{
top.location=self.location;
}
</script>
```

3.2 Tapjacking 防御

Tapjacking 攻击的实质是欺骗用户与其不可见的视图进行交互,因此只要阻止这类交互,就可以在一定程度上减缓 Tapjacking 攻击的危害。Android2.3 以上版本提供了一个特性使 Android 视图只有在可见情况下才能够与用户交互(设置 filterTouchesWhenObscured 的值为 True)。若 Android 应用程序启用该设置,当用户启动另一个视图时会锁定原先的视图,用户就不能与原先的视图进行交互。例如当警告在原视图上弹出时,用户只有在关闭警告后才能继续在原视图上进行操作。但启用该设置可能限制了某些应用程序正常使用视图的需求,在一定程度上影响了应用程序的易用性,也可能给用户使用带来不便。

3.3 基于 WebView 点击劫持攻击的防御

基于 WebView 的点击劫持攻击利用了 WebView 类的缺陷,滥用了继承于视图类的属性,使用透明或叠加的 WebView 页面欺骗用户,因此如 Frame Busting 或 X-FRAME-OPTIONS 等常规点击劫持防御方法不能阻止该攻击。根据文献[13]等的研究总结,可以使用以下方法在一定程度上减缓该类攻击的危害:

(1)使用手机传感器提醒用户:如当用户点击透明或被覆盖的 WebView 时,使用手机震动或蜂鸣提醒用户可能受到了攻击。该方法的缺陷是可能会误报正常使用 WebView 的应用程序,给用户带来困扰。

(2)使用系统状态栏提醒用户:如当检测到有透明或叠加的 WebView 被使用,在系统状态栏中弹出消息提示用户可能受到了攻击。但该提示消息可能很容易地被攻击者屏蔽,若未被屏蔽,一些没有安全意识的用户也可能忽视提示的警告。

(3)动态状态绑定:预先定义一些用户可能遇见的场景,对每一种场景分别限制相应的权限[14],以阻止点击劫持攻击的发生。例如,针对应用程序使用 WebView 的情况进行分类,如图 1 所示,若未使用 WebView,则允许点击、附加 Cookie 等;若使用了重叠的 WebView,则仅允许点击操作;若使用了透明的 WebView,则不允许任何操作。

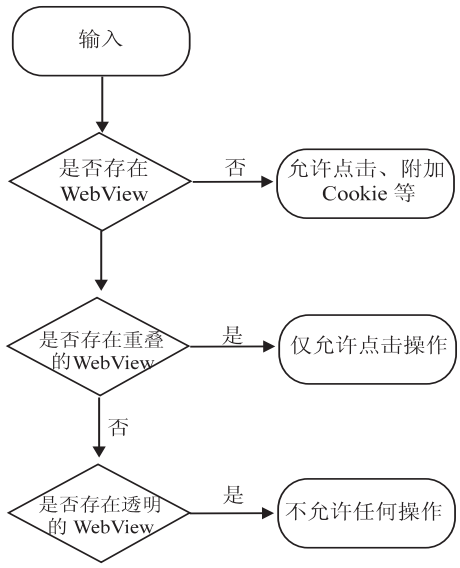

图 1 基于 WebView 的点击权限制流程图

4 结束语

点击劫持攻击的根本原因是攻击者使用虚假页面欺骗用户进行交互,随着移动互联网的普及,这类攻击逐渐向移动端转移,而移动平台的功能越来越完善,不仅可以浏览网页,也可以实现网上支付、交易等金融功能,而通常移动平台用户安全意识普遍不高,因此,移动端点击劫持攻击具有隐蔽性强、影响范围广、危害大的特点。目前,针对 Android 系统点击劫持攻击的几种防御手段都能在一定程度上减缓攻击的危害,但同

时用户的易用性也受到影响,如何在安全性和易用性上实现统一将是一个未来值得研究的方向。

参考文献:

[1] Top 8operating systems from Nov 2013 to Nov 2014[EB/OL]. [2013]. http://gs. statcounter. com/#all - os - ww - monthly-201311-201411.

[2] Clickjacking[EB/OL]. [2014]. https://www. owasp. org/index. php/Clickjacking.

[3] Iframe content background defaults to transparent[EB/OL]. [2013]. https://bugzilla. mozilla. org/show_bug. cgi? id = 154957.

[4] SecTheory. Clickjacking[EB/OL]. [2013]. http://www. sectheory. com/clickjacking. htm.

[5] Stone P. Next generation clickjacking[R/OL]. 2010. http://www. contextis. com/documents/5/Context - Clickjacking _ white_paper. pdf.

[6] Niemietz M. Ui redressing:attacks and countermeasures revisited[R/OL]. 2011. http://ui-redressing. mniemietz. de/uiRedressing. pdf.

[7] Huang L S, Moshchuk A, Wang H J, et al. Clickjacking: attacks and defenses[C]//Proc of USENIX security symposium. [s. l.]:USENIX Association,2012:413-428.

[8] Rydstedt G, Gourdin B, Bursztein E, et al. Framing attacks on smart phones and dumb routers:tap-jacking and geo-localization attacks[C]//Proceedings of the 4th USENIX conference on offensive technologies. [s. l.]:USENIX Association,2010:1-8.

[9] Luo T, Jin X, Ananthanarayanan A, et al. Touchjacking attacks on web in Android, IOS, and windows phone[M]//Foundations and practice of security. Berlin: Springer, 2013: 227 - 243.

[10] Android. webkit | Android developers[EB/OL]. [2014]. http://developer. android. com/reference/android/webkit/package-summary. html.

[11] IE8 security part VII: ClickJacking defenses [EB/OL]. [2010]. http://blogs. msdn. com/b/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking-defenses. aspx.

[12] Rydstedt G, Bursztein E, Boneh D, et al. Busting frame busting: a study of clickjacking vulnerabilities at popular sites [J]. IEEE Oakland Web,2010,2:6-15.

[13] Luo T, Jin X, Du W. Mediums: visual integrity preserving framework[C]//Proceedings of the third ACM conference on data and application security and privacy. [s. l.]: ACM, 2013:309-316.

[14] Enck W, Ongtang M, McDaniel P. Understanding android security[J]. IEEE Security & Privacy,2009,7(1):50-57.

(上接第 134 页)

实现[D]. 杭州:杭州电子科技大学,2014.

[3] 尹相乐,马　力,关　昕. 软件缺陷分类的研究[J]. 计算机工程与设计,2008,29(19):4910-4913.

[4] Li M, Kang H, Zhou P, et al. Hybrid optimization algorithm based on chaos, cloud and particle swarm optimization algorithm[J]. Journal of Systems Engineering and Electronics, 2013,24(2):324-334.

[5] Zeng G P, Fan H L. Two-subpopulation particle swarm optimization based on pheromone diffusion[J]. Applied Mechanics and Materials,2014,667:300-308.

[6] Vapnik V N. The nature of statistical learning theory[M]. New York:Springer-Verlag,1995.

[7] Basili V, Green S, Laitenberger O, et al. The empirical investigation of perspective-based reading[J]. Empirical Software Engineering,1996,1:133-164.

[8] Možina M, Žabkar J, Bratko I. Argument based machine learning[J]. Artificial Intelligence,2007,171:922-937.

[9] Mundra P A, Rajapakse J C. SVM-RFE with MRMR filter for gene selection[J]. IEEE Trans on NanoBioscience, 2010, 9(1):31-37.

[10] 王　青,伍书剑,李明树. 软件缺陷预测技术[J]. 软件学报,2008,19(7):1565-1580.

[11] Kazman R, Bass L, Abowd G, et al. SAAM: a method for analyzing the properties of software architectures[C]//Proceedings of the 16th international conference on software engineering. Sorrento, Italy: IEEE, 1994: 81-90.

[12] 葛贺贺,金　聪,叶俊民. 基于 PSO 和朴素贝叶斯的软件缺陷预测模型[J]. 计算机工程,2011,37(12):36-37.

[13] Whatling C, McPheat W, Hersloef M. The potential link between atherosclerosis and the 5-lipoxygenase pathway: investigational agents with new implications for the cardiovascular field[J]. Expert Opinion on Investigational Drugs, 2007, 16(12):1879-1893.

[14] Johannes M, Brase J C, Fröhlich H, et al. Integration of pathway knowledge into a reweighted recursive feature elimination approach for risk stratification of cancer patients[J]. Bioinformatics,2010,26(17):2136-2144.

[15] Goulão M, Fonte N, Wermelinger M, et al. Software evolution prediction using seasonal time analysis: a comparative study [C]//Proc of 16th European conference on software maintenance and reengineering. [s. l.]:IEEE,2012:213-222.

[16] Fawcett T. An introduction to ROC analysis[J]. Pattern Recognition Letters,2006,27(8):861-874.

Android系统点击劫持攻防技术研究

作者： 钱正旸， 施勇， 薛质， QIAN Zheng-yang， SHI Yong， XUE Zhi
作者单位： 上海交通大学 信息安全工程学院，上海 200240；上海市信息安全综合管理技术研究重点实验室，上海 200240

刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2015(10)