基于图像相似性的 Android 钓鱼恶意应用检测方法①

刘永明[1], 杨 婧[2]

[1](中兴软创科技股份有限公司, 南京 211153)
[2](中国科学院信息工程研究所 信息安全国家重点实验室, 北京 100093)

摘 要: 在移动互联网日益兴盛的今天, 攻击者已开始通过移动应用的形式来实施网络钓鱼, 而现有的网络钓鱼检测方法主要针对网页钓鱼, 无法应对这一新的安全威胁. 钓鱼恶意应用的一个显著特点是通过构造与目标应用相似的界面来诱骗用户输入敏感信息. 基于这种视觉相似性, 提出了一种面向 Android 平台的钓鱼恶意应用检测方法. 该方法通过动态技术截取被检测应用的人机交互界面, 利用图像哈希感知算法计算其与目标应用界面的图像相似度. 如果相似度超过阈值, 则识别被检测应用程序为钓鱼恶意应用. 实验表明, 该方法可以有效检测 Android 平台上的恶意钓鱼应用程序.

关键词: 网络钓鱼; 移动安全; 恶意应用; 图像感知哈希; Android

Detection of Android Phishing Malwares Based on Image Similarity

LIU Yong-Ming[1], YANG Jing[2]

[1](Ztesoft Technology CO., LTD., Nanjing 211153, China)
[2](Institute of Information Engineering CAS, State Key Laboratory of Information Security, Beijing 100093, China)

Abstract: As mobile payment has become increasingly popular recently, some phishing malwares on mobile devices try to steal users' sensitive information through forging a user interface similar to the targeted application, which cannot be detected by existing anti-phishing techniques. We propose a new similarity-based anti-phishing approach to detect phishing malware on Android. With the perceptual image hash technology, it compares the similarity of captured screen images between the suspicious application and the targeted application. If the similarity exceeds a threshold, the suspicious application is determined to be malicious. The experiment proves that the approach is effective to detect phishing malware on Android.

Key words: phishing attack; mobile security; malwares; perceptual image hash; Android

随着互联网的兴起, 电子商务成为现代商业不可小觑的一个重要市场. 而网络支付技术的发展是推动电子商务发展的一个重要因素. 对于传统的网络支付, 用户往往使用个人终端电脑通过网页进行支付. 而近年来移动互联网的火爆, 使得移动支付逐渐受到人们的重视. 在移动支付领域, 支付行为除了可以同 PC 上一样通过浏览器实现外, 还可以通过银行或第三方支付平台的移动应用来完成.

网络支付最大的安全威胁来自于网络钓鱼. 传统的网络钓鱼, 一般是利用社会工程的手段, 比如邮件、短信等, 诱骗受害者访问精心设计的与目标组织非常相似的网站, 来获取受害者在目标组织中的账号、密码等敏感信息[1]. 而在移动终端上, 攻击者还可以通过钓鱼恶意应用实现用户信息窃取.

钓鱼恶意应用是一种安装在用户移动终端上的恶意应用程序, 其通过模拟银行或第三方支付应用的界面, 诱骗用户输入用户名、密码等个人敏感信息并发送至远端的服务器, 盗取用户账户信息.

目前网络钓鱼攻击检测研究主要集中在钓鱼网站检测, 常用的技术一般分为基于 URL、基于页面文本

① 基金项目:工信部电子信息产业发展基金项目([2011]295 号)
收稿时间:2014-04-08;收到修改稿时间:2014-05-20

特征和基于视觉相似三类检测手段[2]. 基于视觉相似的检测方法主要是利用钓鱼网站与真实网站在视觉上存在高度相似性这一特点来实施[3,4]. 但是这些方法不适用于移动终端上的应用程序

而在移动恶意应用检测研究方面, 大多数成果主要关注恶意行为的识别, 通过静态的逆向代码分析[5,6]、动态的监控模拟拦截[7,8]以及静态与动态相结合[9,10]的方式, 分析应用的动作, 识别非正常的行为模式. 但是, 这些工作所针对的恶意行为主要是窃取终端存储的用户敏感信息或越权使用终端的短信发送、摄像头等功能或设备. 而钓鱼恶意应用往往是通过与用户的交互完成敏感信息的窃取, 而非从存储空间中获取, 同时这些信息通过普通的 HTTP 协议即可上传, 也不需要获取短信、摄像头等功能权限. 因此, 一般的移动恶意应用检测方法不一定适用于钓鱼恶意应用. 另外, 也有一些学者从应用相似性的角度入手,通过文本或代码的比较[11,12]检测恶意应用, 如识别经过重打包的应用程序. 但是, 钓鱼恶意应用仿造的往往只是一个登陆的界面, 单纯依靠文本或代码相似性很难识别.

与其他恶意应用程序相比, 钓鱼恶意应用的一个显著特点是构造和目标应用非常相似的登录、支付等关键界面, 诱骗用户输入敏感信息. 本文基于这种相似性, 提出了一种面向 Android 平台的钓鱼恶意应用检测方法. 该方法通过动态调试截取被检测应用的人机交互界面, 利用图像哈希感知算法计算其与目标应用界面的图像相似度. 当相似度超过指定阈值时, 识别被检测应用为针对目标应用的钓鱼恶意应用. 本文所提出的方案能够满足现阶段市场的需要、丰富了移动互联网中钓鱼应用检测技术, 并且实验表明本方法可以有效地识别流行的恶意钓鱼应用.

1 Android钓鱼恶意应用实现

在 Android 系统中, Activity 是一个应用程序的重要组成部分. 一个 Activity 对应于应用程序呈现给用户的一个交互界面. 每当一个新的 Activity 启动时, 前一个 Activity 就会停止, 这些 Activity 都保留在系统中的一个 Activity 历史栈中. 所有的 Activity 遵循后进先出的原则, 当用户按下返回键, 在最顶层 Activity 就出栈并被销毁, 之前的 Activity 就会重新显示在屏幕上.

一种典型的 Android 平台上的钓鱼恶意应用正是利用了 Android 系统的这一特性, 通过 Activity 劫持来实现[13]. 攻击者在钓鱼恶意应用中启动一个后台的服务, 这个服务不断地扫描当前运行的进程, 当发现目标进程启动时, 就启动一个伪装的 Activity. 如果这个 Activity 是登录界面, 那么就可以从中获取用户的账号密码, 发送给远程服务器.

2 图像感知哈希技术

对于相似图像检测, 业界使用广泛的一种技术是图像感知哈希技术. 其基本思路是从图像中抽取特定特征然后计算哈希值, 根据指定的相似度计算方法计算两个图像的哈希值相似度, 如果相似度超过指定阈值, 则判定两个图像内容是相似的.

密码学中常用的哈希函数对输入数据的变化非常敏感, 输入源的相似性无法从哈希值中体现出来. 因此, 无法直接应用于图像检测. 图像哈希函数应该具有感知鲁棒性, 即对于感官上相似的两幅图像, 计算得到的哈希值也应该以很大的概率是接近的.

感知哈希值的计算关键在对图像内容的处理变换抽取特征, 实现方式有多种, 包括基于 DCT 变换、基于图像边缘检测、基于 Radon 变换以及利用块均值等方法. 哈希值相似度计算函数的实现可以采用错误码比特校验、汉明距离比较、相关系数峰值检测等[14].

本文利用 pHash[15]库来进行图像的相似度检测. pHash 是一个实现了多种感知哈希算法的开源软件库, 其中包括 DCT-HASH、MH-HASH 等算法. DCT-HASH 比较简单, 哈希值采用图像的 DCT 变换系数生成, 长度为 8 个字节, 相似性采用汉明距离方法来计算. MH-HASH 方法利用墨西哥帽小波变换生成的图像边缘信息, 然后压缩成 72 字节的哈希值. 哈希相似性同样采用汉明距离计算, 但是由于该方法的哈希值较长, 采用实际汉明距离与哈希值的长度比值作为归一化后的距离. pHahs 库的作者在大量实验数据的基础上认为, DCT-HASH 算法的阈值选择为 22, MH-HASH 的阈值为 0.4, 可以有效区分两张不同图片. 为了比较 DCT-HASH 与 MH-HASH 哪种算法更适用于本文所提出的检测目标, 本文以该作者提出的阈值作为两个算法的检测阈值, 以实验结果中两种算法的检测率和误报率作为算法适用性的评价指标.

3 Android钓鱼恶意应用检测方法

钓鱼攻击的核心特征在于模仿诱骗用户, 比如,

钓鱼短信通过文字和发送号码模仿可信机构, 钓鱼网站通过 URL 与网页内容布局模仿目标网站. 而钓鱼恶意应用, 则是通过构造和目标移动应用非常相似的登录或支付界面, 来诱骗用户输入敏感信息. 其相似性的核心在于应用程序的人机交互界面的视觉相似. 因此, 本文针对这种相似性特点, 提出了一种面向 Android 平台的动态检测技术, 通过截取待检测应用的人机交互界面, 利用图像感知哈希技术来与目标应用程序的特征界面进行比对, 如果二者的界面图像相似度超过阈值, 则识别待检测应用为钓鱼恶意应用.

在 Android 系统中, Activity 的用户交互界面是由 View 来控制的. 每个 View 控制着一片矩形区域并会对用户的操作进行反馈. 例如, 一个 View 可能是一个按钮, 当用户按下时, 它会触发一个事件. 所有 Activity 都必须在 AndroidManifest.xml 文件中声明, 否则无法在系统中运行. 当一个 Activity 在 AndroidManifest.xml 文件中的 Action 属性标记为 Main 后, 则表明该 Activity 是主 Activity, 应用程序启动时由 Launcher 启动.

通过自动化测试方法, 调用 Launcher 启动应用中的主 Activity, 然后对每个 Activity 上的按钮元素发出点击指令, 可以遍历一个应用程序的所有 Activity, 并得到每个 Activity 的界面截图. Google 公司在 Android SDK 开发工具包中提供几款 Android 应用自动化测试工具, 包括 Monkey、Monkeyrunner、HierarchyViewer 等. 其中 Monkeyrunner 工具提供了一组 API, 允许测试人员通过编写 Python 程序黑盒地控制 Android 设置和模拟器, 实现应用程序的安装卸载、发送动作指令、截取保存界面图片等.

3.1 术语定义

表 1 为本文所使用术语定义.

表 1 术语定义

符号	定义
APP_T	目标应用程序
APP_D	待检测的应用程序
$Act_T=\{Act_{Ti}\}, i\in\{1,n\}$	目标应用程序 APP_T 待比较的 Activity 集合, 其中 Act_{Ti} 为第 i 个 Activity, $i\in\{1,n\}$
$I_T=\{I_{Ti}\}, i\in\{1,n\}$	目标应用程序 APP_T 的待比较的 Activity 截图集合, 其中 I_{Ti} 为第 i 个截图, $i\in\{1,n\}$
$Act_D=\{Act_{Dj}\}, j\in\{1,m\}$	待检测的应用程序 APP_D 待比较的 Activity 集合, 其中 Act_{Dj} 为第 j 个 Activity, $i\in\{1, m\}$
$I_D=\{I_{Dj}\}, j\in\{1,m\}$	待检测的应用程序 D 的所有 Activity 截图集合, 其中 ID_j 为第 j 个 Activity 的截图, $j\in\{1,m\}$
$M=\{s_{ij}\}$, $i\in\{1,n\}$, $j\in\{1,m\}$	目标程序 APP_T 与待检测程序 APP_D 的 Activity 截图相似度矩阵, s_{ij} 为 APP_T 的第 i 个截图与 APP_D 的第 j 个截图的相似指数
Phash-Test	图像感知哈希检测函数, 输出相似指数
S-Threshold	图片相似度阈值

3.2 方案描述

本方案的总体流程图如图 1 所示. 对于目标应用 APP_T 与待检测应用 APP_D 分别通过预处理阶段 1 和 2 获得截图集合 I_T 与截图集合 I_D. 在预处理阶段 2 中, 需要识别待检测应用 APP_D 是否与目标应用 APP_T 为同一应用的不同版本, 因为如果二者属于同一应用的不同版本, 则存在相似甚至相同界面的概率很大. 对于集合 I_T 与 I_D, 在检测阶段利用图像哈希感知算法计算元素两两相似指数, 从而检测 APP_D 是否为恶意应用.

具体的过程如下:

预处理阶段 1: 反编译目标应用 APP_T, 获得待截图的 Activity 集合 Act_T, 在模拟器中获取 Act_T 对应的截图集合 I_T.

预处理阶段 2: 反编译目标应用 APP_D, 比较 APP_D 与 APP_T 的程序签名证书中的主体是否一致, 如果否, 获得全部的 Activity 集合 Act_D, 在模拟器中获取 Act_T 对应的截图集合 I_D.

检测阶段: 利用 Phash-Test 计算 I_T 与 I_D 中元素两两相似指数, 组成相似矩阵 M, 如果 M 中存在元素超过 S-Threshold, 则判定 APP_D 为恶意应用.

其中, 从目标应用 APP_T 的 Activity 集合 Act_T, 获取对应的界面截图集合 I_T 的过程如下:

①从 APP_T 的主 Activity 开始, 记录当前 Activity 名称, 遍历当前 Activity 上的按钮控件, 并记录控件 ID;

②模拟用户点击操作, 依次触发按钮控件的单击动作, 并记录已被触发的控件 ID;

③监控 Android 系统中的 Activity 栈中最顶层的 Activity 名称, 如果最顶层的 Activity 名称属于集合 Act_T 且未被截图, 则调用截屏操作接口, 截取当前活动组件对应的界面图像;

④如果集合 Act_T 中的所有元素对应的界面图像都已获取, 或所有记录的控件 ID 都已被触发, 则停止, 并输出集合 Act_T 中的 Activity 所对应的界面图像存储位置组成的集合 I_T, 对于没有获得界面图像的 Activity, 其在 I_T 集合中对应的值为字符串“NULL”.

获取待检测应用 APP_D 的所有界面截图集合 I_D 的过程分为两步: 第一步同上述过程一致, 即从 APP_D 的主 Activity 开始依次遍历 APP_D 的所有 Activity, 获取对应的界面截图; 第二步是再次从 APP_T 的主 Activity 开始依次遍历 APP_T 的所有 Activity, 监控 Android 系统中的 Activity 栈中最顶层的 Activity 名称是否属于集合 Act_D, 如果是, 则截取当前活动组件对应的界面图像. 集合 I_D 由这两步过程中的界面图像存储位置组成.

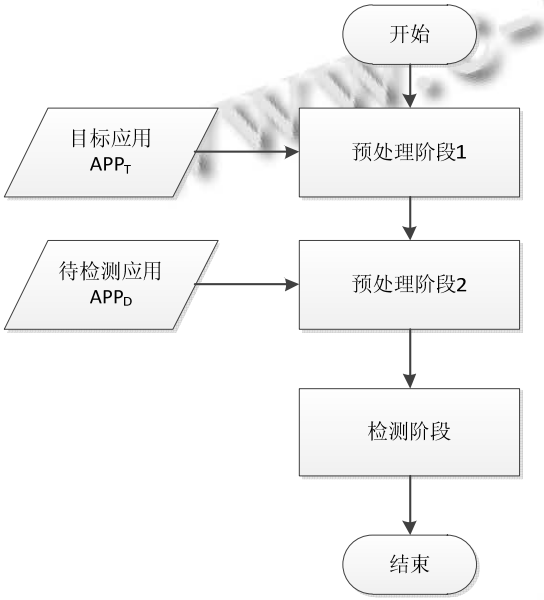

图 1 方案总体流程图

4 实验

本文使用参考文献[13]中的钓鱼恶意应用样本作为待检测应用 APP_D, 目标应用 APP_T 为著名的第三方支付移动应用“支付宝”[16]的 Android 应用程序, 版本号为 3.2.0.1228, 同时从木蚂蚁应用市场[17]选择十五款应用作为参照应用, 构成参照应用集合 $\{APP_{Rk}\}_{k\in\{1,15\}}$. 实验中模拟器目标选择为 Android2.3.1. 对于集合 Act_T, 本实验中选择 APP_T 的登录界面“com.alipay.android.client.Login”作为 Act_T 的唯一元素, 如图 2 所示. 这是因为, 在支付宝应用中, 所有涉及用户账户操作的敏感动作在用户登录前都是无法执行的, 而其他不涉及敏感动作的界面及时被模仿对于攻击者而言也获得不了任何有价值的信息.

图 2 APP_T 的登录界面截图

实验过程中, 对于 APP_D 共截取界面 2 个, 其中一个是该应用的主界面, 如图 3(a)所示, 另一个是拦截支付宝登陆 Activity 后伪造的一个钓鱼界面, 如图 3(b)所示. 对于集合 $\{APP_{Rk}\}_{k\in\{1,15\}}$, 共截取 50 个界面, 其中有些应用限制只有登陆后才能使用, 因此能够拦截到的界面很少; 还有一些应用属于信息类, 虽然一个界面上有很多按钮, 但每个按钮点击后出现的界面对应的 Activity 都是同一个.

图 3 APPT 的界面截图

图 4 为分别采用 phash 库的两种哈希计算方法得到的 APP_D 和 $\{APP_{Rk}\}_{k\in\{1,15\}}$ 所截取的 52 张界面图片的相似指数散点图, 其中图 4(a)为采用 DCT-HASH 计算后的结果, 图 4(b)为采用 MH-HASH 计算后的结果. 图中的横线为两种算法选择的相似阈值, 分别为 22 和

0.4, “*”点为 APP_D 的钓鱼界面截图, “+”点为 APP_D 的登录界面截图, 其他为 $\{APP_{Rk}\}_{k\in\{1,15\}}$ 的界面截图.

很显然, 两种算法都可以准确识别钓鱼界面, 采用 DCT-HASH 算法没有产生误报, 而采用 MH-HASH 在四张图片上产生了误报, 这四张界面截图分别属于三个应用程序, 如图 5 所示. 通过人工分析发现, 其中两张截图在界面的元素布局上与 APP_T 的登录界面有一定的相似性, 而另外两张则布局完全不同, 这说明 MH-HASH 算法利用墨西哥帽小波变换提取图像边缘信息的方法在检测界面截图类图像上灵敏度不够. 因此, DCT-HASH 相比 MH-HASH 算法更适合应用于本文所提出的钓鱼恶意应用检测.

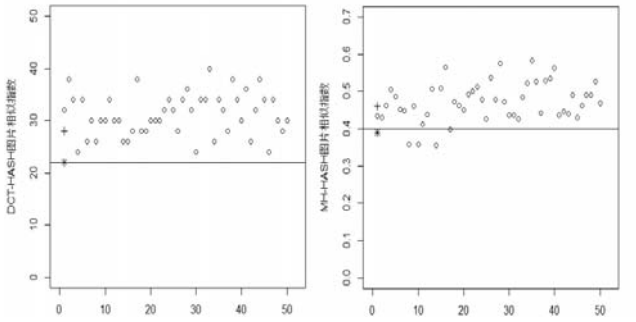

图 4　图片相似指数散点图

值得注意的是, APP_D 是一个演示性质的钓鱼应用, 其钓鱼界面为了突出演示效果在顶端增加了标识以进行识别. 去掉该标识后再使用 DCT-HASH 算法与 APP_T 的登录界面进行对比, 其相似指数降为 10. 这说明实际的钓鱼应用在界面相似度上远高于一般的应用程序.

图 5　MH-HASH 算法误报的截图

5　总结

传统的面向网络钓鱼的检测都是针对网页的, 无法有效检测针对移动应用的钓鱼攻击. 与一般的恶意移动应用相比, 钓鱼恶意应用主要通过构造与目标应用非常相似的登录、支付等关键界面, 来诱骗用户输入敏感信息. 本文基于这种相似性, 提出了一种面向 Android 平台的钓鱼恶意应用检测方法, 通过动态调试截取被检测应用的人机交互界面, 利用图像哈希感知算法计算与目标应用界面的图像相似度, 当相似度超过指定阈值时, 识别被检测应用为针对目标应用的钓鱼恶意应用. 实验表明, 基于 DCT-HASH 算法, 该方法可以有效检测 Android 平台上的恶意钓鱼应用.

参考文献

1 维基百科“钓鱼式攻击”.http://zh.wikipedia.org/wiki/钓鱼式攻击,2013-12-31.

2 黄华军,王耀钧,姜丽清.网络钓鱼防御技术研究.信息网安全,2012,(4):30–35,42.

3 Fu AY, Liu WY, Deng XT. Detecting phishing web pages with visual similarity assessment based on earth mover’s distance (EMD). IEEE Trans. on Dependable and Secure Computing, 2006, 3(4): 301–311

4 周国强,田先桃,张卫丰,张迎周.基于图像感知哈希技术的钓鱼网页检测.南京邮电大学学报(自然科学版), 2012,32(4):59–63,69

5 Clint G, Jonathan C, Jeremy E, Hao C. Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale. In Stefan K, ed. Trust and Trustworthy Computing, 2012: 291–307

6 Aubrey-Derrick S, , Rainer B, Hans-Gunther S, Jan C, Osman K, Kamer AY, Seyit AC, Sahin A. Static analysis of executables for collaborative malware detection on Android. IEEE International Conference on Communications. New York. 2009. 1–5.

7 Iker B, Urko Z, Simin NT. Crowdroid: Behavior-based malware detection system for Android. ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York. 2011. 15–26.

8 William E, Peter G, Byung-Gon C, Landon PC, Jaeyeon J, Patrick M, Anmol NS. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. USENIX Conference on Operating Systems Design and Implementatio. Berkeley. USENIX. 2010. 1–6.

9 Zheng C, Zhu SX, Dai SF, Gu GF, Gong XR, Han XH, Zou W. SmartDroid: An automatic system for revealing ui-based

trigger conditions in android applications. ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York. 2012. 93–104.

10 Vaibhav R, Yan C, William E. Appsplayground: Automatic security analysis of smartphone applications. ACM Conference on Data and Application Security and Privacy. New York. 2013. 209–220.

11 Desnos A. Android: Static analysis using similarity distance. Hawaii International Conference on System Sciences. New York. 2012. 5394–5403.

12 Steve H, Ling H, Edward W, Saung L, Charles C, Dawn S. Juxtapp: A scalable system for detecting code reuse among android applications. In Ulrich F, ed. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Heidelberg. Springer. 2012. 62–81.

13 大谈 Android 安全 1—Activity 劫持与用户防范. http://maosidiaoxian.iteye.com/blog/1623016, 2013-12-28

14 Christoph Z. Implementation and Benchmarking of Perceptual Image Hash Functions [Master Thesis]. Hagenberg: University of Applied Sciences Upper Austria, 2010

15 Phash. http://phash.org/, 2013-12-30

16 支付宝. https://www.alipay.com/, 2013-12-30

17 木蚂蚁应用市场. http://www.mumayi.com/, 2013-12-3