利用 Hover 窃取 Android 用户输入隐私

——侯勤胜

Contributions：

1) 基于 Hover 技术提出了一种新的系统级 Android 用户输入信息推理攻击；
2) 实现了 Hoover，一个概念验证恶意软件；
3) 进行了用户调研，验证了 Hoover 的准确性；
4) 讨论了可能的应对方法，发现这种攻击很难防范。

1. Android Hover 技术

Hover 技术能够在非物理接触屏幕的情况下，完成用户与移动设备的交互。主要的应用机型有三星 Galaxy S4、S5 和 Galaxy Note，总共超过 100 万台设备，Hover 技术的示例图如 Fig 1 所示。

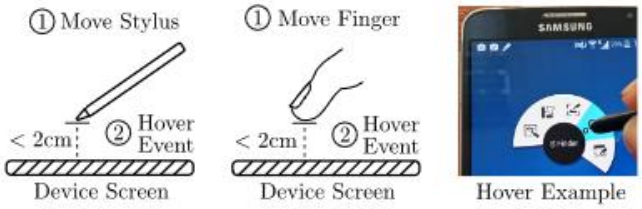


Fig 1. Hover 技术

Hover 的处理流程如下：

1) 当输入设备和屏幕的距离小于 20mm 时，系统会开始生成一个对应(x, y)坐标的 hover 事件序列；
2) 当 touch down 事件发生时，即屏幕被点击，hover exit 事件就会被触发；
3) 当一个 touch up 事件发生时，即屏幕点击结束，系统会生成另一个 hover 事件序列；
4) 最后，当输入设备和屏幕的间距大于 20mm 时，会触发 hover exit 事件。

2. Android View 对象

Android 通过 WindowsManager 接口处理系统虚拟化和屏幕上的 app UI 组件。在本文中利用了 Alert 窗口和 Toast 窗口获取 hover 事件和 touch 事件，通过设置 FLAG_NOT_FOCUSABLE 和 FLAG_NOT_TOUCHABLE 可以让位于顶层的窗口不会影响设备的正常使用；此外，通过设置窗口的 FLAG_WATCH_OUTSIDE_TOUCH 属性还可以获取窗口外的点击事件。Alert 窗口虽然需要申请 SYSTEM_ALERT_WINDOW 权限，但是该权限被普遍使用，不属于敏感权限；Toast 窗口则不需要任何权限就可以调用。

3. 攻击方法

本文的攻击目标是能够高准确度和高粒度（键位级别）地追踪用户的每一次点击。攻击的场景需要用户在搭载 hover 的设备上使用手指或触控笔进行输入，此外，攻击不能够被用户察觉，即不能影响用户与设备的正常交互。

（1）攻击概述

恶意 APP 会生成两个 View：一个是完全透明的 Alert 窗口（或 Toast 窗口），该窗口会位于所有窗口的顶端收集 hover 事件；另一个是 0px 大小的监听窗口，用来监听点击事件的发生。获取 hover 事件的流程如 Fig 2 所示。

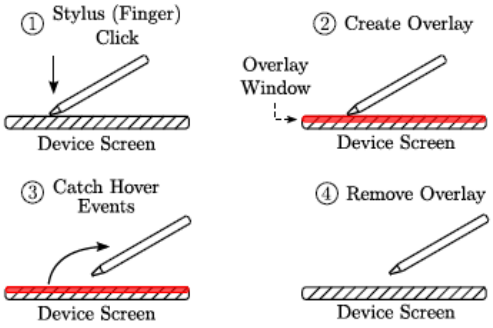


Fig 2. 获取 hover 事件

1) 通过监听窗口（设置 FLAG_WATCH_OUTSIDE_TOUCH）监听点击事件的发生，恶意 APP 得到点击事件发生的时间戳；
2) 点击事件发生后激活 hover 事件获取窗口（全透明的 Alert 窗口），在点击事件发生后激活 Alert 窗口不会影响到用户与设备的正常交互；
3) 输入设备在点击事件后开始离开屏幕（<=20mm），Alert 窗口开始捕捉所产生的 hover 事件序列，系统生成 hover 事件的时间间隔为 19ms；
4) 在经过 70ms 后，Alert 窗口移除，经过测试 70ms 的时间能够收集到足够的 hover 事件而且不会影响用户的交互行为。

（2）推断点击位置

接下来，攻击者会利用得到 hover 事件序列来推断出用户的点击位置。

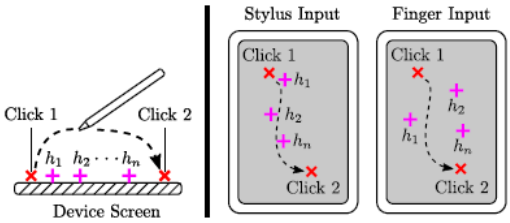


Fig 3. Hover 事件获取示例

从 Fig 3 中可以看出，在使用触控笔的情况下，得到的 hover 事件序列很接近实际路径；但是在使用手指的情况下，得到的 hover 事件序列会分散得多。因此，单纯地考虑第一个 hover 事件的位置是不可行的，为了提高推断的准确度，本文中使用了机器学习工具对 70ms 内所有的 hover 事件进行处理。针对键位推断，本文使用了分类器，分类器的输出最可能是用户的点击位置；在高层次上，本文对收集到的 hover 事件使用回归模型来判断用户点击的位置。本文的实验选择使用了数个基于 **scikit-learn** 框架的回归模型和分类器。由于不同用户会有不同的 hover 事件模式，所以为了得到更高的准确度和健壮性，本文中通过不同的用户来对回归模型和分类器进行训练。

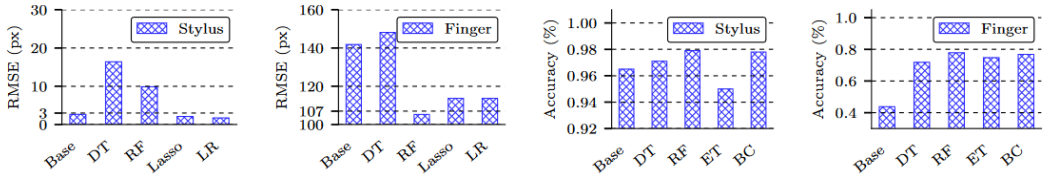
Device Type	Operating System	Input Method
Samsung Galaxy S5	Cyanogenmod 12.1	Finger
Samsung Galaxy Note 3 Neo	Android 4.4.2	Stylus

Table 1. 实验设备

4. 评估实验

实验招募了 20 个志愿者，分为两种攻击场景：一种是用户在屏幕上随机点击，另一种是使

用键盘输入一段文字。每种场景会重复 3 次，分别让志愿者使用**食指、中指和触控笔**。
攻击过程中上传收集的 hover 事件到服务端不需要占用多大的带宽，在 4 小时的时间内，恶意 APP 收集了大约 3800 次用户点击的 hover 事件，这些数据一共有 150KB，不过这是为了得到最大数据量收集的，在用户的日常使用中会明显小很多。
在实验中总共收集了大约 24K 次的用户点击，恶意 APP 在每次点击后会收集 70ms 内的 hover 事件，在用户点击中，有 17K 是键盘输入，7K 是游戏随机输入，在实验过程中，用户并没有察觉任何延迟或恶意软件的存在。



(a) Click position RMSE in pixels. (b) Click position RMSE in pixels. (c) Key prediction accuracy. Standard dev ≤ 1% with all models. (d) Key prediction accuracy. Standard dev ≤ 1% with all models

Fig 4. 场景 1 和场景 2 的评估结果

（1）推断用户随机点击的坐标

Fig.4(a)和 Fig.4(b)分别表示使用触控笔和手指得到的评估结果，Base 是基准值，即点击后输出的第一个 hover 事件，DT 是决策树模型，RF 是随机森林模型，Lasso 是 Lasso 线性模型，LR 是线性回归模型。从图中可以看出不同回归模型的标准误差也不一样，首先可以看出在所有的回归模型中，基于手指的推断准确性<基于触控笔的准确性，这是因为触控笔的 hover 检测准确度高于手指。对于触控笔而言，线性回归模型的准确度最高；对于手指而言，随机森林模型的准确度最高。

（2）推断键盘输入

推断键盘输入的方法：

1) 利用先前的方法推断点击坐标；
2) 观察坐标值在键盘上的位置；
3) 输出相应的键值作为推断结果。

但是这种推断方法适用于触控笔而不适用于手指，所以选择了几种分类模型来提高推断准确性，即决策树、Extra trees、Bagging Classifier 和随机森林，与回归问题一样，选择了相同的基准模型作为比较标准。从图中可以看出，随机森林模型在两种场景下的准确率都是最高，手指是 79%（基准是 40%），触控笔是 98%（基准是 97%）。
结果显示，Hoover 能够足够准确地窃取用户的键盘输入，此外，可以预见如果应用了基于字典的纠正方法，准确度会进一步提高。

（3）区分键盘输入和其他点击事件

排除侧信道分析方法，主要方法有两种：

1) 定位屏幕下方的点击，因为键盘位于屏幕下方，消除 false negatives。
2) 过滤短点击序列（<4 个字母），筛选出一定时间间隔后的长点击序列（输入框的载入时间），降低 false positives。

经过实验验证，false negatives 为 0，false positives 为 14.1%（简单版本）和 10.76%（改进版本）。未来可以通过对输入时间和点击时间的判断来进一步降低 false positives。

（4）无权限实现攻击

使用 Toast Windows（不需要权限）作为监听器和透明覆盖窗口，虽然 Toast Windows 有时间限制，不过只需要保证透明窗口出现 70ms 即可（定期调用 Toast.show()），收集到 Hover 事件后移除覆盖窗口并激活监听器。

（5）未来的攻击方法

未来可以应用两种方法来提高攻击准确度：

1) 检测用户输入的语言，在推断键位的同时辅助以语言修正方法，从而提高准确度；

2) Per-user 模型，可以通过对单个用户数据的分析来进行各个用户输入信息的推断。

（6）滑动输入和安全键盘

滑动输入和安全键盘的应用范围都比较窄，依然有大量信息会暴露在攻击范围内。

5. 攻击引发的后果

（1）窃取用户隐私

窃取用户隐私，可以通过研究用户输入习惯提高攻击准确率。

（2）获取用户生物学信息

可以得到用户的**点击时长**、**点击间隔时间**（两次 touch down 的时间间隔）和**点击悬浮时间**（touch up 和下一个 touch down 的时间间隔），这些信息可以用来判断当前用户的身份（是否为设备拥有者），一方面可以用来进行用户识别（设备是否被盗窃），另一方面可以进行身份伪造来绕过一些生物识别系统。

6. 讨论和防御方法

（1）限制 hover event 使用

影响 APP 使用体验、增加开发和权限管理难度、限制 hover 使用。

（2）触摸过滤

对指定 view 外的点击事件模糊处理（忽略掉），对 hover event 的获取没有影响。

（3）禁止 0px 窗口或 FLAG_WATCH_OUTSIDE_TOUCH

可以将 0px 窗口设置为非常小的 view，可以通过其他信息判断点击事件发生，禁用 FLAG_WATCH_OUTSIDE_TOUCH 会造成许多正常应用 crash。

（4）限制透明 view 或限制系统服务

键盘复制 view 取代透明 view。

（5）对 view 进行标识，添加可信路径

用户安全意识薄弱，透明 view 是动态的，而且存在时间短（70ms），安全监听器作用受限。

（6）保护敏感 view

增加了代码的逻辑复杂度和开发难度。

7. 相关工作

钓鱼攻击：影响用户体验，容易被发现，适用性差，每个 APP 都要订制攻击工具；

侧信道攻击（加速器、麦克风等）：信号精确度低，受环境影响大。

8. 展望

per-user 模型+per-finger 模型，提高攻击准确率。