CCS'18 - Aonzo - Phishing Attacks on Modern Android

优点：① 介绍了Android应用间通信的三种机制，且后续分析的细致.
② 提出抵抗针对移动端PM的钓鱼攻击来社区的推动

问题：① 可以进一步探索更多移动端的PM，以及其它移动平台如iOS的PM.
② 可探索桌面端打开应用的链接是否可能被攻击利用
③ Dashlane的第2种映射应属于many-to-many的映射.

▲研究背景(为什么进行这个研究)

1> 当前移动设备产生的网络流量在网络中占比增多，且仍在增多.
→ 一方面，促使开发人员设计新的技术和机制减少用户交互过程产生的问题
另一方面，频繁的用户认证，使得攻击者加大钓鱼攻击的投入.
▲新的功能简化了用户和开发者的工作，也使得钓鱼攻击更简单更实用

2> 本文重点讨论PM和Instant App会导致安卓设备遭受钓鱼攻击
①移动端PM：从技术角度，安卓应用的沙箱机制阻止它们以编程的方式与外部应用交互
因此PM需拥有安卓框架的支持或修改应用的客户端.(目前安卓提供了3种机制)
另外，PM需将app与对应的web域名对应，才能保证正确填充.(app的后台实际是web服务器)
②Instant App：由Google实现，允许用户在未安装app时尝试应用.
→工作模式：开发者上传app的一部分，与app相关的URL关联，开发者需证明URL对应域名的控制权
通过名为App Link Verification的多步骤过程执行，依赖于Digital Asset Links
→使得攻击者在未安装应用的情况下有可能得到控制设备界面的能力.

3> 移动端PM结合Instant App使得移动端钓鱼攻击更加实际.

▲研究问题
1> 研究PM存在的设计和实现中存在的问题，并结合Instant App说明当前潜在的钓鱼攻击
2> 针对问题，指出存在的安全问题的本质，提出安全的解决方案.
3> 评估当前生态系统对解决方案支持的情况.

▲组件和既有知识

1> 安卓App：①通常从官方市场下载，例如Google Play Store；也可从第三方市场下载，不够多安全，用户需手动开启安全选项side-loading
②app需定义元数据，最重要的是包名(package name)，在同一Play store或同一设备上唯一 ↳注意与web中域名的区别!
③安装后，app在有许多安全机制的沙箱中执行，每个app有正常工作时所需的权限列表.

2> autofill需要PM可以访问其应用的界面，绕过沙箱的限制，安卓提供了3种机制：
①Accessibility Service (ally)：让残疾人访问第三方应用的框架. app请求BIND_ACCESSIBILITY_SERVICE权限并实现了一个组件，在Accessibility Events发生时，可接受系统的回调(事件包括焦点改变或点击按钮. 服务需确定用户在与哪个app交互(通过app的包名)
→PM依赖于ally确定用户交互的app以及是否存在凭证相应的文本域(近期工作表明ally可能被滥用)
②Autofill Framework：在Android Oreo引入，允许PM确定用户交互的app，并可通过编程填充
→app需实现Autofill Service，app需请求BIND_AUTOFILL_SERVICE权限
→Android Oreo引入了新的XML属性帮助PM：(i) importantForAutofill，标识此页面是否可自动填充
(ii) autofillHints：提示填充什么数据的一组字符串.(iii) autofillType：提示Autofill Service希望收到数据类型
若想兼容Autofill Framework，必须使用XML属性.
③OpenYOLO (Google和Dashlane共同开发)，需更改支持PM的app. 包含两个组件：(i) client (ii) credential provider (server).
→client嵌入在app中，server在PM中告知请求app的身份，client和server通过Intent机制交互

▲Web和app关联存在的问题.(PM如何正确将web和app关联. 目前使用包名用于映射)

1> 两者的区别
①包名很像URL (com.aaa v.s. aaa.com)，而域名有DNS服务和HTTPS保护，是安全的
②包名无认证机制，任何人都可以创建任意包名，与包的拥有者(com.aaa)及好包(com.aaa.b)无控制权
③凭证通常与网站关联，而非与app关联. 获取包名后，PM需确定包名与哪个网站对应

2> 讨论攻击的实用性
① 映射是否易受攻击？若易受攻击，A可以欺骗应用欺骗PM自动提交给定网站的凭证？
② 恶意和合法的app是否能共存？当两者包名相同时，A需诱导用户卸载原有app？是否可能绕过这个限制，并完成对PM的欺骗.
③ 恶意app是否可上传至Play Store？通常由于包名不能重复，不可以上传，只能诱导用户从第三方下载
④ A可以选择想要的凭证吗？是否可以细粒度地控制哪个或哪些凭证显示出来？

3> 不同的映射情况 (6种)
① Secure mapping：安全验证当前app的开发者是否有给定域名的权限. 若是则是安全的.
一种方法：Digital Asset Links：网站拥有者发布一个assets文件，包含与合法关联的app所使用包名和合法签名key的哈希标识，PM可以检查asset文件验证合法性.
② Static one-to-one mapping：域名与包名一对一，可能受到攻击(采用代码签名证书难以实现该攻击不实际，因为包名通常唯一)
③ Static many-to-one mapping：n个包映射到一个域名，此时A不需要诱导卸载合法应用更实用
④ Crowdsourced mapping：采用众包，当用户将凭证输入app(而PM未知该映射)，弹出窗口询问用户是否共享该关联，可以让所有用户获知此信息. 攻击者可以自定义关联信息，更实用.
⑤ Heuristic-based mapping：通过包名或其它元数据启发式地推断包名，容易被A利用
⑥ No Mapping：无映射，PM显示所有凭证，未提供任何保护

▲Case Study (PM + Google Smart Lock)
→研究问题：① PM如何解决上述提到的挑战. ② PM的建议系统是如何工作的？
③ PM如何将app和包名映射到网站 ④ 恶意app是否可以欺骗PM为任意网站提供凭证⑤攻击的难度如何

1> 方法：通过简单的静态分析及动态分析，加上逆向工程.

①Package name as app identifier. 确定PM是否以包名作为自动填充的唯一依据（安装一个只有包名相同的app，查看是否会自动填充）

②Mapping Extraction：若包名是唯一相关的信息，则确定PM完成映射的方式，通过几个黑盒测试得出结论，并进一步通过逆向工程支持

③Exploitation. 利用漏洞进行攻击，app需有一包名和至少一个登录表单，可能需加元数据

2> 结果（所有第三方PM均支持Only和 Autofill Framework）

①keeper：不支持OpenYOLO，下载配置文件（包含知名网站）用于在用户手动填写凭证时的自动建议

采用启发式映射：使用包名在Play Store中推断app webpage的URL，但该URL是不可信的，可以创建特定的包名指向特定的域名.

②Dashlane：三者均支持

两个层面的映射：(i)包名硬编码的 package→domain的条目. (ii)采用启发式映射推断域名（将包名以.分割，每个部分 至少3字符在网站域名中就会自动填充）

可以利用many-to-one的特性进行攻击，例如 com.etrade.mobilepro.activity 和 com.etrade.tabletapp 均指向 www.etrade.com（也可能一个包名对应多个域名的凭证）

⇒在Autofill Framework中，会执行额外的检查，从无法绕过检查（可通过不兼容此机制来绕过）.

③LastPass：三者均支持，但对OpenYOLO的支持不成熟，允许用户选择凭证发送到未标识的app）首先采用启发式映射，给定包名 aaa.bbb.ccc，以分割，逆序前两个得到域名模式 bbb.aaa.

若未找到对应的域名，则采用 crowdsourced mapping，LastPass在首次安装时下载该数据库

两种机制均可以被攻击者利用

④1Password：三者均支持，但三者现象不同，Autofill Framework需输入主口令，only不需要，而OpenYOLO只返回空凭据，未提供任何映射机制.

⑤Google Smart Lock. 采用非自动化的secure Mapping，开发者需手动填充Google Form提供证

---

▲利用Instant App实现完全界面控制.⇒促使全面攻击完成

1> Instant App可能被攻击者用于控制整个UI界面：在浏览器沙箱之外获取代码执行权限，进而控制整个UI界面，进而更容易实现攻击.

2> Instant App的名称与icon是从控制的，且PM不区分实际的App和Instant App.

3> Hidden Password Fields：讨论3+4种情况下是否会自动填充.

①Transparency ②Small Size. ③Same-color Background and Foreground ④Invisible Flag

在不同的机制中攻击现象不同，但从可以组合攻击，使得攻击更有效.

---

▲ 问题的关键在于使用包名作为映射的依据，且将映射的实现留给开发人员.

1> 本文提出 getVerifiedDomainNames() API，以域名来做映射，按 OpenYOLO 模式执行.

①Client 发送 Intent 向 PM 请求凭证. ②PM 调用 getVerifiedDomainNames()，传递 Intent 作为参数

③API从Intent中找到发送者的包名，用于找到客户端app的签名密钥

④API从客户端的manifest文件中找出app可访问的域名列表，根据DAL文件验证请求的app是否在其中. 满足条件的域名返回给PM. （每个域名对于一个文件）

2> 应避免侧信道攻击，不让从得知通信的时间

3> 实际部署的难度

从PM存储的域名（88214）查找哪些包含 assetlinks.json 文件，但只有8%有相应的DAL文件，2%根据谷歌文档制定了Android应用程序，采用率很低

→认为PM开发者自己无法解决该问题，需要社区的整体推动.