移动应用安全控制和设计原则

TOP 10

版本（原版最后修改时间 22 MAY 2013, AT 09:40）

译者：卢彬良 沈明星

目录

1. 确认并保护设备上的敏感数据

风险：废弃的移动设备（手机）会无意导致敏感信息泄露，保护设备上的敏感数据是把风险降到最低的方法。

1.1. 在设计阶段，按照敏感程度和应用的访问策略对数据存储区域分类(如密码、用户数据、位置信息、错误日志等等)，并检查调用这些敏感数据的 API 是否安全。

1.2. 敏感数据应安全保存在服务端，而不应存储在客户端。保存的前提是确保网络通信与服务端数据存储机制的安全。 客户端的安全是相对的，它与服务端的安全相比还需具体问题具体分析（参考 ENISA 云风险评估（3）与 OWASP Cloud top 10（4））。

1.3. 数据存储时应采用操作系统提供的文件加密函数或其它权威的加密函数。有的平台提供一种文件加密 API，它的密钥可以通过设备上的解锁码和远程删除功能来保护。如果具有这种可行方法，我们应当广泛使用，加密不在终端进行从而既减小了负担而增加了安全性。然而我们应当注意，密钥是由设备解锁码保护的，数据存储在设备上的安全将仅仅取决于解锁码和远程删除功能。

1.4. 敏感信息（包括密钥）的存储或缓存必须经过加密处理，并尽可能存储在防篡改区域（参考控制二）。

1.5. 基于上下文信息，限制对地理位置等敏感信息的访问（例如，GPS 显示位置在国外时手机钱包无法使用，汽车钥匙在距离汽车 100 米外无法使用……）。

1.6. 在应用程序所需要的有效时间外，不能保存 GPS 历史记录及其它敏感信息（参考 1.7、1.8）。

1.7. 假设共享的存储区是不可信的，数据很容易以各种方式泄露。尤其是：

- 当与其它应用共享缓存和临时存储区时 ；
- 公共的共享存储区——如地址簿，媒体库，音频文件等可能泄露信息的通道。例如，在媒体库中存储包含位置信息的照片，可能会被其它恶意的应用程序访问。
- 不要在全局可读的目录中存放临时文件或缓存。

1.8. 个人敏感信息必须设置最大保存时间，超过时间必须删除（以防止数据还在无限期缓存）。

1.9. 由于没有针对闪存的标准安全删除程序（不可擦写的存储介质除外），所以数据的加密和安全密钥的管理尤为重要。

1.10. 在编写应用程序时应考虑所有数据的生命周期安全（如网络连接信息、临时文件、缓存、备份、删除文件等）。

1.11. 遵循最小化原则，应用程序只能收集和公开业务功能相关所须的数据。在设计阶段定义出必须数据，包括它的敏感度，是否能被收集、存储和使用。

1.12. 运用非持久的标识符，它们必须在任何地方都不与其它应用共享。例如，不能用设备 ID 号作为标识符，应采用随机数生成器（参考 4.3），对应用程序的 session 和 http 的 session/cookie 遵循数据最小化原则。

1.13. 在管理设备上的应用程序必须使用可以远程删除的“删除开关”API，以防止设备丢失造成损失（“删除开关”这种功能是系统级的、专门设计用于可以远程删除程序或数据）。

1.14. 开发者应当考虑设计一个内置的“数据删除开关”功能，允许用户在必要的时候能够远程删除该应用的相关数据（为了保护这个功能被不当使用应设置强认证）。

2. 确保密码等身份认证信息的安全

风险：间谍程序、监控程序、金融恶意程序。用户的认证信息如果被盗，不仅会造成未授权的移动后端服务访问，甚至可能对别的服务或账户带来潜在风险，这个风险是由于用户在不同地方使用了相同密码。

2.1. 可以考虑在设备上使用授权认证 token 的方法来取代密码，并在传输层进行加密（SSL/TLS），后端服务进行验证。

- 智能手机安全开发指南最初为应用程序开发者提供参考标准，某个特定服务的 token 应该在一段时间后失效（服务器端认证），这样可以减少设备丢失带来的损失。使用最新版本的认证标准（如 OAuth2.0），尽可能确保 token 可用性。

2.2. 如果需要在设备上存储密码或 token，应使用手机操作系统所提供的加密方法和密钥存储机制。

2.3. 一些设备和插件允许开发者使用诸如 SD 卡之类的安全元素（5）（6）去存储敏感信息（越来越多的设备提供这种功能）。开发者用这种功能来保存密钥、证书或其它敏感信息，这些安全元素使用了标准的 SD 卡加密认证方法，具有更高的安全性，并遵守了 FIPS 140-2 等级 3 的规定。然而在一般情况下 SD 卡已经作为设备不可分离的一部分，因此不推荐使用 SD 卡来进行二次认证的存储媒介。

2.4. 为用户在设备上提供密码修改功能。

2.5. 以加密或散列的形式对密码和证书进行常规备份。

2.6. 智能手机提供了密码可视化的功能，因为明文密码会比“*”容易记住（7）。

2.7. 滑动解锁码功能很容易被破解（把润滑膏涂抹在屏幕上读手指痕迹），如果没有限制错误次数的话就可以短时间内猜解出来（8）。

2.8. 检查所有密码的熵，包括可视化密码在内（参考 4.1）。

2.9. 确保密码和密钥没有在缓存和日志中输出。

2.10. 在应用程序的二进制代码中不语序存储任何密码和敏感信息，后端也不能存储通用的敏感信息（如硬编码中的密钥），因为应用程序在下载后很容易被反编译。

3. 数据传输安全

风险：网络嗅探、监控。大部分智能手机支持各种网络机制，如 Wi-Fi、运营商网络（3G、GSM、CDMA 等）、蓝牙等。不安全的传输方式会导致敏感信息被截取。

3.1. 假设运营商网络层是不安全的，现有的技术已经可以解密运营商网络，并且也不能保证 Wi-Fi 网络环境都进行了适当的加密。

3.2. 当应用程序需要传输敏感信息时，应该强制使用加密的（如 SSL/TLS）点对点传输方式（例如使用 STS 头（11））保证机密性和完整性保护。这里敏感信息包括用户证书或类似的认证信息。

3.3. 采用权威的加密算法（如 AES，DES），并且选择适当的密钥长度（参考（12）53 页，查看推荐算法）。

3.4. 服务端证书签名必须由权威 CA 提供，不要采用自签名的方式生成证书。

3.5. 敏感数据的传输应采取一定方式降低中间人攻击的风险（如 SSL proxy，SSL strip）。在后端（服务器）进行身份验证后再建立连接，服务器必须使用可信的 SSL 证书。

3.6. 用户界面尽可能地设计简洁，让用户可以很方便地查看证书有效性。

3.7. 不要在手机终端（如短信、彩信、推送通知）发送敏感信息。

参考：不安全 Wi-Fi 环境下 Google 账户登录的脆弱性:
http://www.google.com/url?q=http%3A%2F%2Fwww.uni-ulm.de%2Fin%2Fmi%2Fmitarbeiter%2Fkoenings%2Fcatching-authtokens.html&sa=D&sntz=1&usg=AFQjCNGO-Yp1KHqO8USuLOzxL1Lpwq1Usw

4. 正确的身份认证、授权与会话管理

风险：可能存在通过认证系统绕过或 token、cookies 重放等方法进行非授权认证。（13）

4.1. 应用程序需要适当的认证强度来对用户进行认证，设置密码时应提供密码强度策略。身份认证机制的强度取决于业务是否需要读取或操作敏感信息（如付款操作）。

4.2. 认证通过以后的会话管理也尤为重要，需要安全的协议来实现。例如，后续的请求必须提供身份鉴别信息或 Token 才可以通过（尤其是那些需要授权访问或修改的操作）。

4.3. 用高熵值的不可预测的 session 标志。然而，随机数生成器生成的随机数实际是一种数列，根据 seed 可以算出之后所有的随机数（并且在一定时间后会出现重复），因此提供一个不可预测的 seed 十分重要。通常以日期和时间来生成随机数的方法是不安全的，我们可以通过加入温度传感器、重力感应器的值来进行混合再生成。经过算法测试，以上这种熵最大化的方法是可靠的（或重复使用 SHA1 结合一个随机变量来保持最大的熵——假设固定一个最大长度的 seed）。

4.4. 利用上下文信息增加身份认证的安全性。如 IP 地址位置。

4.5. 应用程序请求访问敏感数据或接口时，尽可能地增加其他身份鉴别因素。如，声音，指纹等任何可以认证个人身份的内容。

4.6. 身份认证应该绑定终端用户的身份（而非设备本身）。

5. 保障后端 API 服务和平台的安全

风险：后端系统攻击及云存储数据丢失。大部分应用程序是通过 REST/Web Services 或协议架构提供的 API 来与服务端进行交互。攻击者可以通过构造恶意数据或第三方应用程序，来向不安全的 API 接口和服务（没有进行加固、或没有及时打补丁）进行攻击。（14）

5.1. 对敏感数据传输、客户端与服务端数据传输、应用程序与外部接口数据传输（如 GPS 位置信息、或包含其它文件元数据）的相关代码进行检查。

5.2. 与应用程序相关的所有服务端程序（Web Services/REST）进行周期性安全测试，如使用静态源代码审计工具或 fuzz 工具进行测试发现安全隐患。

5.3. 确保服务端所运行的环境（操作系统、Web Server 或其它应用组件）升级了最新的安全补丁。

5.4. 确保服务端有足够的日志量以供检测及应急响应（根据数据保护规范的要求）。

5.5. 采用限速和限制单用户/IP 登陆（如果用户身份可识别）的方法来防范 DDos 攻击。

5.6. 对应用程序需要集中调用的资源和接口进行 Dos 压力测试。

5.7. 可以用类似于 web 安全测试的方法来对 Web Services, REST 以及 API 进行测试：

- 除了正常的案例测试以外，进行极限测试
- 对服务端的 Web Service, REST 及 API 进行安全测试

6. 与第三方应用程序或服务的安全交互

风险：数据泄露。用户可能在不知情的情况下安装了恶意软件，这些软件会获取用户的个人信息。

6.1. 对应用程序调用的所有第三方代码和库进行验证（如：确保来源是否可靠、是否有支持维护、是否不含后门）。

6.2. 跟踪应用程序中相关第三方框架和 API 的安全补丁，确保它们都升级了最新的补丁。

6.3. 特别注意未经验证的第三方应用的数据接收和发送（例如网络传输）。

7. 设置用户隐私使用条例

风险：个人隐私信息泄露、数据非法处理。欧盟强制规定，在收集可识别的个人信息前，必须获得用户的同意（PII）。（15）（16）

7.1. 为用户提供使用个人数据的隐私条例，用户可以在阅读之后选择“同意”或“不同意”。

7.2. 以上隐私条例可以由下面三种方式提供给用户选择：

- 安装时；
- 运行时、数据发送之前；
- 采用 “opt-out”机制，用户选择“不同意”时退出程序。

7.3. 检查应用程序是否收集 PII（可能不明显），如：你把包含了个人信息的持久性唯一标识符链接到数据存储中心了吗？

7.4. 审计通信机制，检查是否有意外的数据泄漏（如图像元数据）。

7.5. 对允许 PII 的传输做日志记录，这个记录可以允许用户查看（也可以考虑在服务端记录用户数据的存储）。这种记录应尽量最小化保存个人数据（如使用哈希）。

7.6. 检查你的隐私条例是否与其它地方（如 APP-native + webkit HTML ）的相重复或矛盾（如在数据处理的表述方面），并及时修正。

8. 实现对付费资源的授权访问控制

风险：智能手机给应用程序提供了访问拨打电话、短信、数据漫游、NFC 支付等资源的接口。对于调用这些 API 访问权限的应用程序需要特别小心，以防止滥用，设计时应充分考虑攻击者利用这些来盗取用户资金。

8.1. 以特定格式记录应用程序调用付费资源的日志（例如签署用户同意的回执发送到受信任的服务器后端），并提供给最终用户监测。日志应受到保护，避免非授权访问。

8.2. 检查异常的资源使用模式，触发重新认证。例如地理位置、用户语言发生变化等。

8.3. 考虑设置默认的白名单程序来访问付费资源。例如地址簿只提供给电话拨打使用。

8.4. 对所有付费资源 API 的调用进行身份验证。例如验证开发者证书。

8.5. 确保该付费功能的 API 回调不会传输明文账户、定价、计费、条款等信息。

8.6. 针对任何可能影响付费功能的应用程序行为，警示用户后取得用户同意。

8.7. 采用快速休眠（参考 3GPP 文档）、缓存等优化方案，减少基站的信道负荷。

9. 保证移动应用的分布式安全

风险：分布式安全能够减少 OWASP Mobile Top 10 Risks 和 ENISA top 10 risks 中所述的风险。

9.1. 考虑到应用商店对应用程序的审批要求和延迟周期，必须为应用程序设计、配置安全补丁的更新功能。

9.2. 大部分应用商店都会监控不安全的应用程序，在发生安全事件时能在短时间内远程删除程序。因此，万一您发现应用程序存在严重安全漏洞，可以及时通过官方应用商店的分布式应用的安全网机制来进行远程删除。

9.3. 为用户提供安全问题的反馈接口。如提供一个 security@example.com 电子邮件地址。

10. 运行环境中的代码错误检查

风险：源代码执行解释会被攻击者利用，把未经验证的输入作为代码解释。例如，超出游戏等级、脚本、SMS 数据头解释等。这可能导致恶意软件绕过应用程序商店的安全防护机制，会引起注入攻击，从而导致数据泄漏、监控、木马种植、恶意拨号等。

需要注意的是，有时候我们并不知道我们的代码中包含了解释器。通过用户输入的数据和第三方 API 来定位功能访问， 例如 JavaScript 解释器。

10.1. 最小化执行解释器的权限。

10.2. 定义全面的转义语法。

10.3. 对执行解释器进行 Fuzz 测试。

10.4. 用沙箱保护执行解释器。

11. 附录

本文参照 OWASP《Top Ten Mobile Controls》进行翻译：
https://www.owasp.org/index.php/Mobile#section_control_1

OWASP/ENISA 合作

OWASP 与欧洲网络与信息安全局（ENISA）合作建立控制原则。ENISA 已在其官网发表了该合作文档《智能手机安全开发指南》：http://www.enisa.europa.eu/activities/application-security/smartphone-security-1/smartphone-secure-development-guidelines

贡献者

文档为 OWASP 与 ENISA 共同合作编写，感谢以下作者：

- Vinay Bansal, Cisco Systems
- Nader Henein, Research in Motion
- Giles Hogben, ENISA
- Karsten Nohl, Srlabs
- Jack Mannino, nVisium Security
- Christian Papathanasiou, Royal Bank of Scotland
- Stefan Rueping, Infineon
- Beau Woods, Stratigos Security

相关通用编码最佳实践

有一些适用于移动应用的通用编码最佳实践，在此列举出比较重要的几条：

- 除了案例测试之外，执行极限测试。
- 对所有输入进行验证。
- 减小代码量和代码复杂度，可以以圈复杂度（cyclomatic complexity）作为衡量标准。（17）
- 使用安全的开发语言（避免缓冲区溢出）。
- 设置个安全反馈邮箱地址，security@example.com。
- 使用代码审计工具与 fuzz 测试发现安全问题。
- 使用安全的字符函数，避免缓冲区和整数溢出。
- 最小化应用程序的运行权限。尤其要注意某些 API 默认权限较大，应禁用。
- 禁止代码/应用程序以 root 账户或系统管理员权限运行。
- 以特权用户的权限来对应用程序进行测试。
- 避免在客户端设备上开放监听端口，应使用操作系统提供的通信机制。
- 在应用程序发布 release 版本前删除所有测试代码。
- 确保适当的日志记录，但不能过度，尤其是包含用户隐私的内容。

企业指南

- 如果需要在设备中安装含有企业隐私相关的应用程序，应当强制要求设备具有安全功能（如 PIN 码、远程删除/管理功能、程序监控等）。
- 设备的证书应当进行强认证。

参考文献

1. ENISA. Top Ten Smartphone Risks . [在线文档] http://www.enisa.europa.eu/act/application-security/smartphone-security-1/top-ten-risks.
2. OWASP. Top 10 mobile risks. [在线文档] https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_Ten_Mobile_Risks.
3. Cloud Computing: Benefits, Risks and Recommendations for information security. [在线文档] 2009. http://www.enisa.europa.eu/act/rm/files/deliverables/cloud-computing-risk-assessment.
4. OWASP Cloud Top 10. [在线文档] https://www.owasp.org/index.php/Category:OWASP_Cloud_%E2%80%90_10_Project.
5. Blackberry developers documents. [在线文档] http://www.blackberry.com/developers/docs/7.0.0api/net/rim/device/api/io/nfc/se/SecureElement.h tml,.
6. Google Seek For Android. [在线文档] http://code.google.com/p/seek-for-android/.
7. Visualizing Keyboard Pattern Passwords. [在线文档] http://cs.wheatoncollege.edu/~mgousie/comp401/amos.pdf.
8. Smudge Attacks on Smartphone Touch Screens. Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. s.l. : Department of Computer and Information Science – University of Pennsylvania.
9. Google vulnerability of Client Login account credentials on unprotected . [在线文档] http://www.uni- ulm.de/in/mi/mitarbeiter/koenings/catching-authtokens.html.
10. SSLSNIFF. [在线文档] http://blog.thoughtcrime.org/sslsniff-anniversary-edition.
11. [在线文档] http://tools.ietf.org/html/draft-ietf-websec-strict-transport-sec-02.
12. NIST Computer Security. [在线文档] http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_PART3_key-management_Dec2009.pdf.
13. Google's ClientLogin implementation . [在线文档] http://www.uni-ulm.de/in/mi/mitarbeiter/koenings/catching-authtokens.html.
14. [在线文档] https://www.owasp.org/index.php/Web_Services.
15. EU Data Protection Directive 95/46/EC. [在线文档] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML.
16. [在线文档] http://democrats.energycommerce.house.gov/sites/default/files/image_uploads/Testimony_05.04.11_Spafford.pdf.
17. [在线文档] http://www.aivosto.com/project/help/pm-complexity.html.
18. [在线文档] http://code.google.com/apis/accounts/docs/AuthForInstalledApps.html.
19. Google Wallet Security. [在线文档] http://www.google.com/wallet/how-it-works-security.htm.