OWASP 中国
Open Worldwide Application Security Project

OWASP 十大移动风险 2024

OWASP Mobile Top 10 2024

前言

1 OWASP **十大移动风险** 2024

2 2024 对比 2016 版本变化

OWASP 2016	OWASP 2024	版本变化
M1: Improper Platform Usage	M1:凭据使用不当 Improper Credential Usage	新增
M2: Insecure Data Storage	M2:供应链安全不足 Inadequate Supply Chain Security	新增
M3: Insecure Communication	M3:不安全的身份认证/授权 Insecure Authentication/Authorization	将 M4 和 M6 合并到 M3
M4: Insecure Authentication	M4:输入/输出验证不足 Insufficient Input/Output Validation	新增
M5: Insufficient Cryptography	M5:通信不安全 Insecure Communication	M3 调至 M5
M6: Insecure Authorization	M6:隐私控制不足 Inadequate Privacy Controls	新增
M7: Client Code Quality	M7:二进制保护不足 Insufficient Binary Protections	将 M8 和 M9 合并到 M7
M8: Code Tampering	M8:安全配置错误 Security Misconfiguration	改写 M10
M9: Reverse Engineering	M9:数据存储不安全 Insecure Data Storage	M2 调至 M9
M10: Extraneous Functionality	M10:加密措施不足 Insufficient Cryptography	M5 调至 M10

3 其他漏洞

在初次发布的列表中虽未提及，但未来可能会将以下漏洞纳入考虑范围：

- 数据泄露
- 硬编码的秘密
- 不安全的访问控制
- 路径覆盖和路径遍历
- 未受保护的端点（深度链接、活动、服务等……）
- 不安全的共享

4 项目贡献者

4.1 Project Leads

- Milan Singh Thakur
- Alaeddine MESBAHI
- Kunwar Atul
- Mohamed Benchikh

4.2 中文项目组

中文项目组长：张坤（破天）

中文项目组成员：刘畅（玄道）、许天翔、王斌

审核：张坤（破天）、王斌

M1：凭据使用不当

1.1 威胁主体

特定应用程序

移动应用程序中，那些利用硬编码凭据和不当凭据使用的威胁主体可能包括使用公开或定制工具进行的自动化攻击。这些威胁主体可能会定位并利用硬编码凭据，或利用因凭据不当使用而产生的弱点。

1.2 攻击向量

可利用性：容易

攻击者可以利用硬编码凭据和不当凭据使用带来的漏洞。一旦这些漏洞被攻击者识别，就可以利用硬编码凭据来获得对移动应用中敏感功能的未授权访问。此外，他们还可能滥用凭据，例如通过不当验证或不当存储的凭据获取访问权限，从而绕过合法的身份验证。

1.3 安全弱点

常见程度：常见　　可检测性：容易

实施不当的凭据管理，例如使用硬编码凭据或处理凭据不当，可能导致严重的安全弱点。全面的安全测试过程应致力于识别这些问题。例如，安全测试人员应仔细查找移动应用程序的源代码或任何配置文件中是否存在硬编码凭据。

1.4 技术影响

影响：严重

不当的凭据管理会带来一系列严重的技术影响。未经授权的用户可能获得对移动应用程序或其后端系统内敏感信息或功能的访问权限。这种情况可能导致数据泄露、用户隐私损失、欺诈活动以及允许对管理功能的潜在访问。

1.5 业务影响

影响：严重

由于硬编码凭据和不当凭据使用而导致的不当凭据管理会对业务产生重大影响，包括：

- 声誉损害；
- 信息盗窃；
- 欺诈；
- 数据未经授权访问。

1.6 我是否易受“凭据使用不当”的影响?

当移动应用程序使用硬编码凭据或凭据被滥用时，就可能存在不安全的凭据管理。以下情况可能表明您的移动应用程序存在漏洞：

- **硬编码凭据**：如果移动应用程序在其源代码或任何配置文件中包含硬编码凭据，这表明存在明显的安全风险。
- **不安全的凭据传输**：如果凭据在传输过程中未加密或通过不安全的通信通道传输，则可能存在漏洞。
- **不安全的凭证存储**：如果移动应用程序以不安全的方式在设备上存储用户凭证，则可能存在安全风险。
- **弱用户身份验证**：如果用户身份验证依赖于弱协议，或容易被绕过，也可能导致安全漏洞。

1.7 如何防止“凭据使用不当”?

避免不安全的凭据管理需要避免使用硬编码凭据，并妥善处理用户凭据。

1.7.1 避免使用硬编码凭据

硬编码凭据很容易被攻击者发现，并为未经授权的用户提供便利的访问入口。因此，务必避免在移动应用程序的代码或配置文件中使用硬编码凭据。

1.7.2 妥善处理用户凭证

用户凭证应始终以安全的方式存储、传输和验证：

- 在传输过程中加密凭证。
- 不要在设备上存储用户凭证，而应考虑使用安全的、可撤销的访问令牌。
- 实施强验证用户身份协议。
- 定期更新和轮换 API 密钥或令牌。

1.8 典型攻击场景

以下场景展示了移动应用程序中凭据使用不当的安全风险：

场景#1：硬编码凭据	场景#2：不安全的凭证传输	场景#3：不安全的凭据存储
攻击者在移动应用程序的源代码中发现了硬编码凭据，并利用这些凭据未经授权地访问应用程序的敏感功能或后端系统。	攻击者拦截移动应用程序与后端系统之间不安全传输的凭证，并使用这些截获的凭证冒充合法用户，从而进行未授权访问。	攻击者获取了用户设备的物理访问权限，并从移动应用程序中提取存储的凭据，随后利用这些凭据获取用户账户的未授权访问权限。

M2:供应链安全不足

2.1 威胁主体

特定应用程序

攻击者可以利用移动应用程序供应链中存在的漏洞来操纵应用功能。例如，攻击者可以在移动应用程序的代码库中插入恶意代码，或在构建过程中篡改代码以植入后门、间谍软件或其他恶意代码。这种行为可以使攻击者窃取数据、监视用户甚至控制移动设备。此外，攻击者还可能利用第三方软件库、SDK、供应商或硬编码凭据中的漏洞，获取对移动应用程序或后端服务器的非法访问权限。这可能导致未授权的数据访问或篡改、拒绝服务，甚至完全接管移动应用程序或设备。

2.2 攻击向量

可利用性：中等

针对供应链安全不足，存在多种利用方式，例如内部威胁代理或攻击者可以在应用程序开发阶段注入恶意代码，然后他们可以破坏应用的签名密钥或证书，将恶意代码伪装成可信代码进行签名。

另一种方式是，攻击者可以利用应用程序中使用的第三方库或组件的漏洞进行攻击。

2.3 安全弱点

常见程度：常见　　可检测性：困难

供应链安全不足的漏洞通常是由于缺乏安全编码实践、代码审查和测试不足导致的，从而在应用程序中引入漏洞。

其他可能导致供应链安全漏洞的原因包括不充分或不安全的应用程序签名和分发流程、第三方软件组件或库中的弱点、数据、加密、存储的安全控制不足或敏感数据暴露于未经授权的访问。

2.4 技术影响

影响：严重

如果攻击者成功利用供应链安全漏洞，其技术影响可能十分严重。具体影响取决于漏洞性质，但可能包括：

- **数据泄露**：攻击者可能窃取敏感数据，如登录凭据、个人信息或财务信息。数据泄露可能给受害者带来长期影响，例如身份盗用或金融欺诈。
- **恶意软件感染**：攻击者可能将恶意软件引入移动应用程序，使用户设备受到感染，从而窃取数据或执行恶意操作。此类恶意软件可能难以检测和清除，并可能对用户设备和数据造成重大损害。
- **未经授权的访问**：攻击者可能获取移动应用程序的服务器或用户设备的访问权限，并执行未授权的操作，例如修改或删除数据。这可能导致数据丢失、服务中断或其他技术问题。
- **系统被攻陷**：攻击者可能完全控制移动应用程序的系统，导致系统完全失控。这可能引发应用程序关闭、大量数据丢失，并对应用程序开发者的声誉造成长期损害。

2.5 业务影响

影响：严重

如果攻击者成功利用供应链安全漏洞，其对业务的影响可能十分重大。具体影响取决于攻击的方式，以及组织的规模、行业和整体安全状况，但可能包括：

- **经济损失**：组织可能因攻击而遭受经济损失，例如调查漏洞的成本、通知受影响用户的成本，或法律和解费用。此外，如果用户对移动应用程序失去信任并停止使用，组织可能会失去收入。
- **声誉损害**：组织可能因攻击而遭受声誉损害，导致品牌形象受损和客户信任度下降。这可能导致收入减少和难以吸引新客户。
- **法律和监管后果**：组织可能因攻击面临法律和监管后果，例如罚款、诉讼或政府调查。这些后果可能给组织带来严重的经济和声誉损害。
- **供应链中断**：攻击可能会破坏组织的供应链，导致商品或服务的交付延迟或中断，从而造成经济损失和声誉损害。

2.6 我是否易受“供应链安全不足”的影响？

如果您使用由第三方开发的移动应用程序，或依赖第三方库和组件，您可能容易受到供应链安全漏洞影响。此类漏洞可能由多种因素引发，例如：

- **第三方组件缺乏安全性**：第三方组件（如库或框架）可能包含攻击者可以利用的漏洞。如果移动应用程序开发者未对第三方组件进行充分审查或未及时更新，则应用程序可能容易受到攻击。
- **恶意内部威胁**：恶意的内部人员（如不受信任的开发者或供应商）可能故意在移动应用程序中引入漏洞。如果开发者未实施足够的供应链过程安全控制和监控，此类攻击可能发生。
- **测试和验证不足**：如果移动应用程序开发者未对应用程序进行充分测试，应用程序可能容易受到攻击。此外，开发者也可能无法验证供应链过程的安全性，导致应用程序中存在漏洞。
- **缺乏安全意识**：如果移动应用程序开发者没有足够的安全意识，他们可能不会实施必要的安全控制措施来防范供应链攻击。

2.7 如何防止“供应链安全不足”漏洞？

- 在移动应用程序开发生命周期内实施安全编码实践、代码审查和测试，以识别和消除漏洞。
- 确保应用程序签名和分发流程安全，防止攻击者伪造签名并分发恶意代码。
- 仅使用受信任和验证的第三方库或组件，以降低漏洞风险。
- 建立应用更新、补丁和版本管理的安全控制措施，防止攻击者利用应用程序中的漏洞。
- 通过安全测试、扫描和监控供应链安全事件，及时检测和响应安全事件。

2.8 示例攻击场景

场景#1：恶意软件注入

攻击者在应用程序开发阶段向某款热门移动应用注入恶意软件。攻击者随后使用有效证书签署该应用，并成功将其分发至应用商店，从而绕过了应用商店的安全检查。用户下载并安装了受感染的应用程序，该应用程序窃取用户的登录凭据和其他敏感数据。随后，攻击者利用窃取的数据进行欺诈或身份盗用，给受害者造成严重的经济损失，并对应用程序提供商造成声誉损害。

M3:不安全的身份认证/授权

3.1 威胁主体

特定应用程序

利用身份认证和授权漏洞的威胁主体通常通过使用现有或定制的工具来实施自动化攻击。

3.2 攻击向量

可利用性：容易

一旦攻击者了解了身份认证或授权方案中的漏洞，他们可以通过以下两种方式来利用这些弱点：一种是伪造或绕过身份认证，通过直接向移动应用的后端服务器提交服务请求，绕过与移动应用的直接交互；另一种是在成功通过身份认证控制后，作为合法用户登录应用，然后强制浏览到一个易受攻击的端口，执行管理功能。这两种利用方式通常都是通过攻击者拥有的设备或僵尸网络中的移动恶意软件来实现的。

3.3 安全弱点

常见程度：常见　　可检测性：中等

为了测试移动应用中授权和身份认证方案是否完善，测试人员可以采取多种策略。对于授权，测试人员可以对移动应用进行二进制攻击，尝试执行本应只有更高权限用户才能执行的特权功能，尤其是在移动应用处于“离线”模式时。测试人员还应尝试使用低权限会话令牌在对应的 POST/GET 请求中执行任何敏感功能。

不当或缺失的授权方案可能允许攻击者利用已认证但权限较低的用户执行他们本不应执行的功能。当授权决策在移动设备中做出而不是通过远程服务器时，这种特权提升攻击的风险会增加，这种情况通常因移动设备的离线可用性需求所导致。

在身份认证方案不足的情况下，测试人员可以在移动应用处于“离线”模式时进行二进制攻击，旨在绕过离线身份认证，然后执行本应需要离线认证才能进行的功能。测试人员还应尝试通过移除任何 POST/GET 请求中的会话令牌，匿名执行任何后端服务器功能。

不当或缺失的身份认证方案可能允许攻击者匿名执行移动应用或其后端服务器中的功能。由于移动设备的输入特性通常鼓励使用简短的密码或 4 位 PIN 码，这些身份认证漏洞在移动应用中相当常见。

移动应用面临独特的身份认证需求，与传统的 Web 身份认证方案不同，主要是由于其不同的可用性要求。与传统的 Web 应用程序（用户需要在线并实时与后端服务器认证）不同，移动应用可能需要满足因移动互联网连接不稳定或不可预测而要求的离线认证。这一需求显著影响了开发人员在实现移动认证时必须考虑的因素。

3.4 技术影响

影响：严重

身份认证和授权不足对系统的技术影响可能是广泛且显著的，具体取决于执行了何种过度特权功能。例如，对于不当授权，当远程或本地管理功能过度特权执行时，可能导致系统损坏或敏感信息泄露。

当解决方案无法识别执行操作请求的用户时，就会出现身份验证不佳的连锁技术反应。因为无法确认用户的身份，这会立即导致无法记录或审计用户活动。缺乏身份验证会导致无法检测攻击源，无法理解任何潜在漏洞的性质，也无法制定防止未来攻击的策略。

此外，身份认证失败还可能暴露底层授权失败。当身份认证控制失败时，解决方案无法验证用户的身份，而身份通常与用户的角色和相关权限密切相关。如果攻击者能够匿名执行敏感功能，这表明底层代码没有验证发出操作请求的用户权限。因此，代码的匿名执行突显了身份认证和授权控制的失败。

3.5 业务影响

影响：严重

身份认证和授权不足对业务的影响通常至少会导致以下几项后果：

- 声誉损害；
- 信息窃取；
- 欺诈；
- 未经授权的数据访问。

3.6 我是否易受“不安全的身份认证/授权”的影响？

了解身份认证和授权之间的区别对于评估移动应用的安全性至关重要。身份认证识别个体，而授权则验证已识别个体是否具有执行特定操作所需的权限。这两个方面密切相关，因为授权检查应该紧随移动设备请求认证之后。

不安全的授权可能发生在组织未在执行来自移动设备的 API 端口请求之前验证个体身份的情况下，因为在没有确定请求者身份的情况下，几乎不可能进行授权检查。

以下是一些不安全授权的明显指标：

- **存在不安全的直接对象引用（IDOR）漏洞**：发现 IDOR 漏洞可能表明代码没有进行适当的授权检查；
- **隐藏端点**：开发人员可能忽略了后端隐藏功能的授权检查，认为这些隐藏功能只会被具有适当角色的用户访问；
- **用户角色或权限传输**：如果移动应用在请求中将用户角色或权限传输给后端系统，则可能是不安全的授权。
- 同样，移动应用程序也可能表现出各种不安全认证的迹象：
- **匿名后端 API 执行**：应用程序能够在没有提供访问令牌的情况下执行后端 API 服务请求，可能表明身份认证不安全；
- **本地存储密码或共享密钥**：如果应用程序将任何密码或共享密钥本地存储在设备上，这可能是身份认证不安全的标志；
- **弱密码策略**：使用简化的密码输入过程可能意味着不安全的认证；
- **使用 FaceID 和 TouchID 等功能**：使用 FaceID 或 TouchID 等功能可能表明身份验证不安全。

3.7 如何防止“不安全的身份认证和授权”？

要防止不安全的认证和授权，至关重要的是避免弱模式并加强安全措施。

3.7.1. 避免弱模式

应避免以下不安全的移动应用身份认证设计模式：

- 如果将 Web 应用程序移植到移动等效应用程序，请确保移动应用程序的身份认证要求与 Web 应用程序的组件匹配。移动认证不能使用比 Web 浏览器更少的身份认证因素；
- 本地用户身份认证可能导致客户端绕过漏洞。如果应用程序将数据本地存储，身份认证过程可以通过运行时操作或二进制修改绕过。如果离线身份认证是强有力的业务需求，请咨询有关防止针对移动应用程序的二进制攻击的其他指南；
- 尽可能在服务器端执行所有身份认证请求。在身份认证成功后，应用程序数据将加载到移动设备上，确保仅在成功认证后才提供应用程序数据；
- 如果必须使用客户端数据存储，请使用从用户登录凭证安全派生的加密密钥加密数据。但是，数据还存在通过二进制攻击解密的额外风险；
- “记住我”功能不应将用户密码存储在设备上；
- 移动应用程序理想情况下应该使用特定于设备的身份验证令牌，用户可以在移动应用程序中撤销该令牌，从而减轻设备被盗/丢失造成的未经授权的访问风险；
- 避免使用可伪造值（如设备标识符或地理位置）进行用户认证；
- 在移动应用程序中持久身份认证应作为可选项实现，而非默认启用；
- 尽可能避免允许用户提供 4 位数字 PIN 作为认证密码。

3.7.2. 强化身份认证

- 开发人员应假设所有客户端授权和身份认证控制都可能被恶意用户绕过。服务器端强化这些控制至关重要；
- 由于离线使用要求，移动应用可能需要执行本地身份认证或授权检查。在这种情况下，开发人员应在本地执行完整性检查，检测任何未经授权的代码更改。请参阅有关检测和应对二进制攻击的其他指南；
- 使用 FaceID 和 TouchID 解锁生物特征保护的秘密，并安全地保护敏感的身份认证材料，如会话令牌。

3.7.3. 防止不安全授权

要避免不安全授权：

- 后端系统应独立验证已认证用户的角色和权限。不要依赖来自移动设备的任何角色或权限信息；
- 假设所有客户端授权都可能被绕过，因此尽可能强化服务器端授权控制；
- 如果移动应用程序代码中需要离线授权检查，开发人员应执行本地完整性检查以检测未经授权的代码更改。

3.8 攻击场景示例

以下场景展示了移动应用程序中弱身份认证或授权控制的情况：

场景#1：隐藏的服务请求：	场景#2：接口依赖：	场景#3：可用性要求：
开发人员假设只有经过认证的用户才能生成移动应用程序提交给后端处理的服务请求。在处理请求期间，服务器代码不会验证传入请求是否与已知用户相关联。因此，攻击者向后端服务提交服务请求并匿名执行影响解决方案合法用户的功能。	开发人员假设只有授权用户才能在其移动应用程序上看到某个特定功能的存在。因此，他们期望只有合法授权的用户才能从他们的移动设备发出服务请求。处理请求的后端代码没有验证请求所关联的身份是否有权执行该服务。因此，攻击者能够使用权限较低的用户账户执行远程管理功能。	由于可用性要求，移动应用程序允许使用 4 位数字密码。服务器代码正确存储了密码的哈希版本。然而，由于密码长度极短，攻击者可以使用彩虹哈希表快速推断原始密码。如果服务器上的密码文件（或数据存储）被泄露，对手可以快速推断用户的密码。

场景#4：不安全的直接对象引用：	场景#5：LDAP 角色传输：
用户向后端 REST API 发出包含行为者 ID 和 OAuth 承载令牌的 API 端口请求。用户在传入 URL 中包含其 ID，并在请求中将访问令牌作为标准标头包含。后端验证承载令牌的存在，但未能验证与承载令牌关联的行为者 ID。结果，用户可以调整 ID 并获取其他用户的账户信息。	用户向后端 REST API 发出包含标准 oAuth 承载令牌以及包含用户所属 LDAP 组列表的标头的 API 端口请求。后端请求验证承载令牌，然后检查传入的 LDAP 组以获得正确的组成员资格，然后继续执行敏感功能。然而，后端系统未独立验证 LDAP 组成员资格，而是依赖于来自用户的传入 LDAP 信息。用户可以调整传入头部，随意声称自己是任意 LDAP 组的成员，从而执行管理功能。

M4：输入/输出验证不足

4.1 威胁主体

特定应用程序

在移动应用中，来自外部来源（如用户输入或网络数据）的数据未经过充分验证和清理，可能会引入严重的安全漏洞。未能正确验证和清理此类数据的移动应用，容易受到特定于移动环境的攻击，例如 SQL 注入、命令注入和跨站脚本（XSS）攻击。

这些漏洞可能导致严重后果，包括未经授权访问敏感数据、篡改应用功能以及可能危及整个移动系统的安全。

输出验证不足可能导致数据损坏或呈现漏洞，使恶意行为者能够注入恶意代码或篡改展示给用户的敏感信息。

4.2 攻击向量

可利用性：困难

输入/输出验证不足会使我们的应用程序面临严重的攻击，包括 SQL 注入、XSS、命令注入和路径遍历等。这些漏洞可能导致未经授权的访问、数据篡改、代码执行以及整个后端系统的入侵。

4.3 安全弱点

常见程度：常见　　可检测性：容易

输入/输出验证不足的漏洞发生在应用程序未能正确检查和清理用户输入，或未能验证和清理输出数据时。该漏洞可以通过以下方式被利用：

- **输入验证不足**：当用户输入未经过彻底检查时，攻击者可以通过输入意外或恶意数据来操纵它。这可以绕过安全措施并导致代码执行漏洞或未经授权的系统访问。
- **输出验证不足**：如果输出数据未经过适当的验证和清理，攻击者可以注入恶意脚本，这些脚本会被用户浏览器执行。这可能导致跨站脚本（XSS）攻击，从而导致数据窃取、会话劫持或篡改展示内容。
- **缺乏上下文验证**：未能考虑特定的上下文或预期的数据格式可能导致漏洞，如 SQL 注入或格式化字符串漏洞。这些情况发生在未验证的用户输入直接纳入数据库查询或在格式化字符串函数中处理不当的情况下，允许攻击者操纵查询或执行任意代码。
- **未能验证数据完整性**：未验证数据完整性使应用程序容易受到数据损坏或错误处理的影响。攻击者可能篡改关键系统变量或引入格式错误的数据，破坏应用程序的功能。

这些漏洞通常源于应用程序逻辑中的错误、验证检查的不完整实现、缺乏安全意识或测试和代码审查不足。

4.4 技术影响

影响：严重

输入/输出验证不足的漏洞可能对受影响的应用程序产生多方面的技术影响：

- **代码执行**：恶意行为者可以利用此漏洞在应用程序环境中执行未经授权的代码，绕过安全措施。

- **数据泄露**：验证不足可以使攻击者操控输入，从而导致未经授权的访问和敏感数据的提取。
- **系统攻破**：攻击者可以未经授权访问底层系统，从而危及系统，甚至可能控制系统。
- **应用程序中断**：恶意输入可能导致应用程序中断、崩溃或数据损坏，影响应用程序的可靠性和功能。
- **声誉损害**：成功利用此漏洞可能导致因数据泄露和客户信任丧失而造成的声誉损害。
- **法律和合规问题**：验证不足可能导致法律责任、监管处罚及违反数据保护法规。

4.5 业务影响

影响：严重

输入/输出验证不足的漏洞具有显著的技术和业务影响。从应用程序的角度来看，影响包括：

- **代码执行**：攻击者可以利用此漏洞执行未经授权的代码，可能导致系统攻破和未经授权的访问。
- **数据泄露**：验证不足允许攻击者操控输入，导致数据泄露和敏感信息的未经授权访问。
- **系统中断**：漏洞的利用可能导致应用程序崩溃、不稳定或数据损坏，从而导致服务中断和运营效率低下。
- **数据完整性问题**：验证不足可能导致数据损坏、错误处理或不准确地输出，危及系统的可靠性和完整性。

在业务方面，影响包括：

- **声誉损害**：漏洞被成功利用可能导致数据泄露、系统中断和客户不信任，损害组织的声誉和品牌形象。
- **法律和合规后果**：由于验证不足，未遵守数据保护法规可能导致法律责任、监管处罚和潜在的财务损失。
- **财务影响**：数据泄露或系统中断可能导致财务损失，包括事件响应、修复费用、法律费用以及潜在的收入损失。

4.6 我是否易受“输入/输出验证不足”的影响?

应用程序可能因以下原因容易受到输入/输出验证不足的影响：

- **缺乏输入验证**：未能正确验证用户输入可能使应用程序暴露于 SQL 注入、命令注入或 XSS 等注入攻击。
- **输出清理不充分**：输出数据清理不足可能导致 XSS 漏洞，使得攻击者注入和执行恶意脚本。
- **忽视上下文特定验证**：未能根据数据上下文考虑特定验证需求可能产生漏洞，例如路径遍历攻击或未经授权访问文件。
- **数据完整性检查不充分**：未执行适当的数据完整性检查可能导致数据损坏或未经授权的修改，危及可靠性和安全性。
- **不良安全编码实践**：忽视安全编码实践，如使用参数化查询或转义/编码数据，会导致输入/输出验证漏洞。

4.7 如何防止“输入/输出验证不足”?

为了防止“不足的输入/输出验证”漏洞：

1. **输入验证：**
   - 使用严格的验证技术验证和清理用户输入。
   - 实施输入长度限制并拒绝意外或恶意数据。
2. **输出清理：**
   - 正确清理输出数据以防止跨站脚本（XSS）攻击。
   - 在显示或传输数据时使用输出编码技术。
3. **上下文特定验证：**
   - 根据数据上下文（例如文件上传、数据库查询）进行特定验证，以防止路径遍历或注入攻击。
4. **数据完整性检查：**
   - 实施数据完整性检查以检测和防止数据损坏或未经授权修改。
5. **安全编码实践：**
   - 遵循安全编码实践，如使用参数化查询和预编译语句，防止 SQL 注入。
6. **定期安全测试：**
   - 进行定期的安全评估，包括渗透测试和代码审查，以识别和修复漏洞。

4.8 攻击场景示例

场景#1： 通过恶意输入执行远程代码	**场景#2： 通过不足的输出验证进行注入攻击**	**场景#3： 通过格式错误的输出执行远程代码**
攻击者发现一个缺乏适当输入验证和清理的移动应用程序。通过精心构造包含意外字符的恶意输入，他们利用应用程序的行为。由于验证不足，应用程序错误处理输入，导致漏洞。攻击者成功执行任意代码，获得对设备资源和敏感数据的未经授权访问。	攻击者发现一个输出验证和清理不足的移动应用程序。他们利用一个处理用户生成内容或不可信数据的入口点。通过精心构造包含代码或脚本（如 HTML、JavaScript、SQL）的恶意输入，攻击者利用缺乏输出验证的缺失。通过用户交互提交精心构造的输入，应用程序未能验证或清理它，允许执行注入代码或未预期的操作。攻击者成功执行基于注入的攻击，如跨站脚本（XSS）或 SQL 注入，危及应用程序的完整性并获得对敏感信息的访问。	攻击者发现一个处理用户提供的数据并生成动态输出的移动应用程序。攻击者精心构造特定格式的数据，利用应用程序的输出验证不足。攻击者通过直接交互或利用暴露的 API 提交格式错误的数据。应用程序未能正确验证或清理生成的输出，允许攻击者精心构造数据执行代码或触发未预期的操作。通过利用此漏洞，攻击者实现远程代码执行，获得对移动设备、其资源或敏感数据的控制。

M5：通信不安全

5.1 威胁主体

特定应用程序

大多数现代移动应用都会与一个或多个远程服务器交换数据。在数据传输过程中，它通常会通过移动设备的运营商网络和互联网进行，如果数据以明文形式传输或使用过时的加密协议，监听网络的威胁主体可以拦截并修改这些数据。威胁主体的动机各不相同，包括窃取敏感信息、进行间谍活动、身份盗窃等。存在以下威胁主体：

- 与您共享本地网络的攻击者（被攻破或监控的 Wi-Fi）；
- 恶意运营商或网络设备（路由器、手机信号塔、代理服务器等）；
- 您移动设备上的恶意软件。

5.2 攻击向量

可利用性：容易

尽管现代应用程序依赖于加密协议如 SSL/TLS，但它们有时在其实现中可能存在缺陷，例如：

使用过时的协议和/或不良配置设置；

接受不良的 SSL 证书（自签名证书、已撤销的证书、过期证书、错误的主机等）；

不一致性（仅在某些工作流如身份验证中使用 SSL/TLS）。

5.3 安全弱点

常见程度：常见　　可检测性：一般

尽管现代移动应用旨在保护网络流量，但它们的实现中经常存在不一致之处。这些不一致可能导致暴露数据和会话 ID 的漏洞。仅仅因为应用使用了传输安全协议，并不意味着它的实现是正确的。要识别基本缺陷，您可以观察手机上的网络流量。然而，要检测更微妙的缺陷，则需要仔细查看应用的设计和配置。

5.4 技术影响

影响：严重

此漏洞可能暴露用户数据，导致账户接管、用户冒充、个人身份信息（PII）泄露等问题。例如，攻击者可能会拦截用户凭证、会话令牌、二次身份验证（2FA）令牌，这可能为更复杂的攻击打开大门。

5.5 业务影响

影响：中等

至少，通过通信通道拦截敏感数据会导致隐私违规。

用户机密性的泄露可能导致以下后果：

- 身份盗窃；

- 欺诈；
- 名誉损害。

5.6 我是否易受“通信不安全”的影响？

这个风险涵盖了从 A 点到 B 点的数据传输，但以不安全的方式进行。它包括移动设备之间的通信、应用程序到服务器之间的通信，或移动设备到其他设备之间的通信。此风险涵盖了移动设备可能使用的所有通信技术：TCP/IP、Wi-Fi、蓝牙/蓝牙低能耗（Bluetooth-LE）、NFC、音频、红外、GSM、3G、SMS 等。

所有 TLS 通信问题都属于此类。所有 NFC、蓝牙和 Wi-Fi 问题也都属于此类。

突出的特点包括打包某种敏感数据并将其传输进出设备。敏感数据的示例包括加密密钥、密码、私人用户信息、账户详情、会话令牌、文档、元数据和二进制文件。这些敏感数据可以来自服务器，也可以是从应用到服务器，或者可能在设备与其他本地设备（例如 NFC 终端或 NFC 卡）之间传输。此风险的定义特征是存在两个设备及其之间传输的数据。

如果数据存储在设备本地，则属于不安全的数据。如果会话详细信息通过安全的方式（例如，通过强 TLS 连接）传输，但会话标识符本身存在问题（例如，可预测、低熵等），那么这属于不安全的身份认证问题，而不是通信问题。

不安全通信的常见风险包括数据完整性、数据机密性和来源完整性。如果数据在传输过程中可以被修改，而这一变化无法被检测到（例如，通过中间人攻击），那么这是该风险的一个典型例子。如果机密数据可以在传输过程中被泄露、获取或推测（例如，通过窃听）或通过录制对话并在后续攻击（离线攻击）中进行攻击，也属于不安全通信问题。如果未能正确设置和验证 TLS 连接（例如，证书检查、弱密码套件、其他 TLS 配置问题），这些问题也都属于不安全通信范畴。

5.7 如何防止“通信不安全”？

6.7.1. 通用最佳实践

1) 假设网络层不安全且容易被窃听。
2) 对于移动应用使用的传输通道，应用 SSL/TLS 保护，确保数据安全地传输到后端 API 或 Web 服务。
3) 考虑外部实体，如第三方分析公司、社交网络等，在应用通过浏览器/Webkit 运行时使用它们的 SSL 版本。避免混合 SSL 会话，因为它们可能暴露用户的会话 ID。
4) 使用强大的行业标准密码套件，并选择合适的密钥长度。
5) 使用受信任的 CA 提供商签发的证书。
6) 永远不允许不良证书（自签名证书、过期证书、不受信的根证书、已撤销证书、错误的主机等）。
7) 考虑证书锁定。
8) 始终要求 SSL 链验证。

9) 在验证端点服务器的身份并使用受信任证书时，才能建立安全连接。

10)如果移动应用检测到无效证书，则通过 UI 警告用户。

11)不通过替代通道（例如 SMS、MMS 或通知）发送敏感数据。

12)如果可能，在数据传输到 SSL 通道之前对任何敏感数据应用独立的加密层。如果 SSL 实现中发现未来的漏洞，加密数据将为防止机密性泄露提供二次防护。

13)在开发周期中，避免覆盖 SSL 验证方法以允许不受信任的证书，而是尝试使用自签名证书或本地开发证书颁发机构（CA）。

14)在进行安全评估时，建议分析应用流量，查看是否有任何流量通过明文通道。

6.7.2. iOS 特定最佳实践

最新版本的 iOS 中默认类处理 SSL 密码强度协商非常出色。问题出现在开发人员临时添加代码绕过这些默认设置以克服开发困难时。除了上述通用实践外：

- 确保证书有效，并在失败时关闭连接。
- 使用 `CFNetwork` 时，考虑使用 Secure Transport API 来指定受信任的客户端证书。在几乎所有情况下，应使用 `NSStreamSocketSecurityLevelTLSv1` 以确保更高标准的密码强度。
- 开发完成后，确保所有 `NSURL` 调用（或 NSURL 包装器）不允许自签名或无效证书，例如 NSURL 类方法 `setAllowsAnyHTTPSCertificate`。
- 考虑通过执行以下操作实现证书锁定：导出您的证书，将其包含在应用程序包中，并将其锚定到您的信任对象。使用 NSURL 方法 `connection:willSendRequestForAuthenticationChallenge`：现在将接受您的证书。

6.7.3. Android 特定最佳实践

- 在开发周期结束后，删除所有可能允许应用程序接受所有证书的代码，如 org.apache.http.conn.ssl.AllowAllHostnameVerifier 或 SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER。这些相当于信任所有证书。
- 如果使用扩展 SSLSocketFactory 的类，请确保正确实现 checkServerTrusted 方法，以便正确检查服务器证书。
- 避免覆盖 `onReceivedSslError` 方法以允许无效的 SSL 证书。

5.8 攻击场景示例

以下是渗透测试人员在检查移动应用程序的通信安全性时常发现的一些常见场景：

场景#1： 缺乏证书检查	场景#2： 弱握手协商	场景#3： 隐私信息泄漏
移动应用程序和端点成功连接并执行 TLS 握手以建立安全通道。然而，移动应用程序未能检查服务器提供的证书，并无条件地接受服务器提供的任何证书。这破坏了移动应用程序和端口之间的相互身份验证能力。移动应用程序容易受到通过 TLS 代理的中间人攻击。	移动应用程序和端点成功连接并在连接握手过程中协商了密码套件。客户端成功与服务器协商使用弱密码套件，导致加密强度较弱，攻击者可以轻松解密。这危及了移动应用程序和端点之间的通道的机密性。	移动应用程序通过非安全通道而不是通过 SSL/TLS 向端点传输个人身份信息。这危及了移动应用程序和端点之间任何与隐私相关数据的机密性。

场景#4： 凭证信息泄漏	场景#5： 双重身份验证绕过
移动应用程序通过非安全通道而不是通过 SSL/TLS 向端点传输用户凭据。这使得攻击者以明文形式拦截这些凭据。	移动应用程序通过非安全通道而不是通过 SSL/TLS 从端点接收会话标识符。这使得攻击者可以通过使用拦截的会话标识符绕过双重身份验证。

M6：隐私控制不足

6.1 威胁主体

特定应用程序

隐私控制涉及保护个人身份信息（PII），例如姓名、地址、信用卡信息、电子邮件和 IP 地址、健康信息、宗教信仰、性取向和政治观点等。

这些信息对攻击者有很高的价值，原因有多种。例如，攻击者可以：

- 冒充受害者进行欺诈；
- 滥用受害者的支付数据；
- 用敏感信息勒索受害者；
- 通过破坏或篡改受害者的关键数据来伤害受害者。

一般来说，PII 可能被泄露（违反机密性）、篡改（违反完整性）或被销毁/阻塞（违反可用性）。

6.2 攻击向量

可利用性：一般

PII 的典型来源通常都受到较好保护，例如应用程序的沙箱、与服务器的网络通信、应用程序的日志和备份等。某些源虽然保护较少，但仍然难以访问，如 URL 查询参数和剪贴板内容。

因此，获取 PII 需要攻击者先突破其他层的安全性。攻击者可能通过窃听网络通信、访问文件系统、剪贴板或日志，或者通过特洛伊木马获取移动设备并创建备份进行分析。由于 PII 只是可以通过移动设备存储、处理和传输的数据，因此提取或篡改它的方式多种多样。

6.3 安全弱点

常见程度：常见　　可检测性：容易

几乎所有应用程序都会处理某种形式的 PII。许多应用甚至收集和处理超出其功能需求的 PII，这使得它们在没有业务需求的情况下更具吸引力，成为攻击目标。

隐私违规的风险因开发人员对 PII 的不当处理而增加。PII 应始终在考虑到攻击者可能访问通信和存储介质的情况下进行处理。

因此，如果某些个人数据的收集使攻击者有动机通过不充分保护的存储或传输介质操控或滥用这些数据，那么该应用程序就容易受到隐私侵犯。

6.4 技术影响

影响：低

隐私违规通常对系统整体的技术影响较小。只有当 PII 包括身份认证数据时，可能会影响某些全局安全属性，例如可追溯性。

如果用户数据被篡改，可能会导致该用户无法使用系统。通过不规范的数据，也可能导致后端系统出现问题，尤其是当后端未做适当的清理和异常处理时。

6.5 业务影响

影响：严重

隐私违规对业务的影响范围和严重程度通常取决于受影响用户的数量、受影响数据的关键性，以及违规发生地适用的数据保护法规。隐私违规的业务影响通常至少会导致以下后果：

- **违反法律法规**：隐私控制方面的法律法规是最大的问题。GDPR（欧洲）、CCPA（美国加利福尼亚州）、PDPA（新加坡）、PIPEDA（加拿大）、LGPD（巴西）、2018 年数据保护法（英国）、POPIA（南非）、PIPL（中国）等都规定了公司未能保护用户数据所面临的制裁。
- **因受害者诉讼造成的财务损失**：任何受隐私违规影响的个人都可能起诉让违规发生的应用提供商。这些诉讼的成功与否具体取决于适用的法律法规以及提供商是否能够证明他们具备足够的最新保护机制。
- **声誉损害**：如果隐私侵犯影响到大量用户，很可能会被媒体曝光，从而为应用程序提供商带来负面宣传。结果可能是应用程序甚至其提供商的其他无关产品的销售和使用下降。
- **PII 的丢失或盗窃**：被盗的实际信息可能会被滥用，甚至用于攻击应用提供商。例如，特定的用户数据可能被用来冒充受害者对提供商进行社会工程攻击。

6.6 我是否易受“隐私控制不足”的影响?

只有当应用处理某种形式的个人身份信息时，它才容易受到隐私控制不足的影响。几乎所有应用都存在这种情况：客户端应用程序的 IP 地址对服务器可见，应用程序使用日志和与崩溃报告或分析一起发送的元数据是适用于大多数应用程序的 PII。通常，应用程序会从用户那里收集和处理额外的、更敏感的 PII，如账户、支付数据、位置等。

对于使用 PII 的应用，它可能像处理任何其他敏感数据一样暴露 PII。最常见的情况通过以下方式发生：

- 不安全的数据存储和通信（参见 M5, M9），
- 带有不安全身份验证和授权进行数据访问（参见 M3, M1）
- 针对应用程序沙盒的内部攻击（参见 M2, M4, M8）。

其他 OWASP Mobile Top 10 风险提供了关于应用如何可能受到不同攻击途径的深入洞察。

6.7 如何防止“隐私控制不足”?

不存在的东西无法被攻击，因此防止隐私违规的最安全方法是最小化处理的 PII 的数量和种类。这要求全面了解应用中所有 PII 资产。在此基础上，以下问题应被评估：

- 是否所有处理的 PII 都是真正必要的，例如姓名和地址、性别、年龄？
- 是否可以用较不关键的信息替代一些 PII，例如用粗略的位置代替精确的位置？
- 是否可以减少某些 PII 的处理频率，例如每小时更新位置而不是每分钟更新？

- 是否可以对某些 PII 进行匿名化或模糊处理，例如通过哈希、分桶或添加噪声？
- 是否可以在某个过期周期后删除某些 PII，例如只保留最近一周的健康数据？
- 用户是否可以同意选择性使用 PII，例如为了获得更好的服务，但同时了解额外的风险？

剩余的 PII 除非绝对必要，否则不应存储或传输。如果必须存储或传输，则必须通过适当的认证和可能的授权来保护访问。对于特别关键的数据，还应考虑纵深防御。例如，健康数据除了存储在应用程序的沙盒中之外，还可以使用密封在设备 TPM 中的密钥进行加密。因此，如果攻击者设法绕过沙箱限制，数据仍然不可读。其他 OWASP Mobile Top 10 风险建议了安全存储、传输、访问和处理敏感数据的措施。

威胁建模可以用来确定在特定应用中最可能出现隐私侵犯的方式。然后，可以将保护 PII 的努力集中在这些方面。

静态和动态安全检查工具可能揭示常见的错误，如敏感数据的日志记录或泄漏到剪贴板或 URL 查询参数。

6.8 攻击场景示例

以下场景展示了移动应用程序中的隐私控制不足：

场景#1：日志和错误消息未充分清理	**场景#2：在 URL 查询参数中使用 PII**	**场景#3：在备份中排除个人数据/未设置 hasFragileUserData**
报告日志和异常对于保证应用的质量至关重要。崩溃报告和其他使用数据帮助开发人员修复 BUG，并了解应用的使用情况。然而，如果开发人员在日志或错误信息中包含 PII，这些日志和错误信息可能会暴露 PII。此外，第三方库也可能在其错误信息和日志中包含 PII。一个常见的问题是数据库异常，它会泄露部分查询或结果。这很可能对用于收集和评估崩溃报告的任何平台提供商可见。它还可能出现在用户界面上，或者对可以读取设备日志的攻击者可见。开发人员在记录内容时应特别小心，并确保在显示给用户或报告给服务器之前清理异常信息。	URL 查询参数常用于向服务器传输请求参数。然而，URL 查询参数至少在服务器日志中可见，通常也会出现在网站分析中，甚至可能出现在本地浏览器历史记录中。因此，敏感信息绝不能作为查询参数传输。而应作为头部或请求体的一部分发送。	大多数应用程序处理的 PII 存储在其沙盒中。应用程序应明确配置要包含在设备备份中的数据。攻击者可能获取设备并创建备份或从其他来源获取备份，从中可以提取沙盒内容。

另外，在 Android 中通过将 hasFragileUserData 设置为'true'，应用程序可以在卸载时保留其数据。稍后成功安装具有相同包 ID 的恶意应用程序的攻击者可以访问这些数据。

因此，这两个设置都应明确设置，以使开发者的意图透明，并控制通过备份或应用程序后续安装之间的信息流动。

M7：二进制保护不足

7.1 威胁主体

特定应用程序

针对应用程序二进制文件的攻击者动机各异。

二进制文件可能包含有价值的机密信息，例如商业 API 密钥或硬编码的加密密钥，攻击者可能会滥用这些信息。此外，二进制文件中的代码本身也可能很有价值，例如包含了关键的业务逻辑或预训练的人工智能模型。有些攻击者可能不会直接针对应用本身，而是通过应用来探索相应后端的潜在弱点，为后续攻击做准备。

除了收集信息外，攻击者还可能操控应用的二进制文件，以免费访问付费功能或绕过其他安全检查。在最坏的情况下，流行的应用程序可能会被修改以包含恶意代码，并通过第三方应用商店或以新名称分发，以利用毫无防备的用户。一种常见的攻击方式是重新配置应用中的支付标识符，重新打包应用并通过应用商店分发。当用户从应用商店下载这种未经授权的副本时，攻击者将获得支付款项，而非原应用提供商。

7.2 攻击向量

可利用性：容易

应用程序二进制文件通常可以从应用商店下载或从移动设备复制，因此二进制攻击很容易设置。

应用程序二进制文件可能会受到两种类型的攻击：

- **逆向工程：**应用程序二进制文件被反编译并扫描以获取有价值的信息，如秘密密钥、算法或漏洞。
- **代码篡改：**应用程序二进制文件被操纵，例如删除许可证检查、绕过付费墙或获得其他用户利益。或者，应用程序可以被篡改以包含恶意代码。

7.3 安全弱点

常见程度：常见　　可检测性：容易

几乎所有应用都容易受到二进制攻击，许多应用最终会成为某种攻击的目标。那些将敏感数据或算法硬编码在二进制中的应用特别容易受到二进制攻击。这些应用应该采取对策，以抵挡潜在攻击者足够长的时间，使攻击者因突破保护的成本高于从成功中获得的收益而放弃。通常情况下，例如在防拷贝保护的情况下，只需要延长破解过程，直到应用销售的目标收入达到。

一般而言，完全编译的应用（如 iOS 应用）比 Android 应用中的高级字节码更不容易遭受逆向工程和代码篡改（但注意，这对于使用跨平台技术开发的应用如 PWA 或 Flutter 可能不适用）。

尤其是流行的应用更有可能被篡改并通过应用商店重新分发。检测和移除这些篡改的副本可以由专业公司提供支持，但也可以通过应用本身的某些检测和报告机制实现。

需要注意的是，目前没有完全可靠的机制可以防止二进制攻击。防范二进制攻击是开发者投入反制措施与攻击者突破这些措施之间的军备竞赛。因此，对于每个应用来说，问题是：应该投入多少精力来防止二进制攻击？

7.4 技术影响

影响：中等

如前所述，二进制攻击可能发生为逆向工程并泄露应用二进制文件中的信息，或发生为代码篡改并改变应用的工作方式。

如果秘密信息泄露，必须迅速替换系统中的所有密钥，这在密钥硬编码在应用中的情况下很困难。二进制文件中的信息泄露也可能暴露后端中的安全漏洞。

然而，篡改对系统技术健全性的影响更大。通过篡改二进制文件，攻击者可以任意改变应用的工作方式，例如为自己的利益服务，或者扰乱后端系统，尤其是当后端系统没有针对这种恶意请求进行足够的加固时。

如前所述，二进制攻击可能以逆向工程的形式发生并泄露来自应用程序二进制文件的信息，或者以代码篡改的形式改变应用程序的工作方式。

如果秘密泄露，必须在整个系统中快速替换它们，这在秘密硬编码在应用程序中时很难做到。从二进制文件中泄露的信息还可能揭示后端的安全漏洞。

然而，篡改对系统的技术完整性有更大的影响。通过篡改二进制文件，攻击者可以任意更改应用程序的工作方式，例如为了自己的利益或干扰后端（如果后端不足以防范此类恶意请求）。

7.5 业务影响

影响：中等

如果大规模滥用商业 API 密钥或类似内容，可能会造成重大成本。同样，如果应用程序被篡改以删除许可检查或将功能与竞争应用程序一起发布，也会产生类似情况。在这两种情况下，个人破解应用程序或窃取 API 密钥用于个人使用很可能不会被注意到。然而，在规模上，例如当 API 密钥甚至功能被系统性地与其他应用一起使用时，恶意竞争对手可能会因成本显著降低而获得显著优势。

如果通过巨大努力开发的知识产权，如算法或人工智能模型变得公开或被恶意竞争对手窃取，应用程序开发者的商业模式可能会受到更大威胁。

特别是对于被重新分发带有恶意代码的流行应用程序，可能会造成巨大的声誉损害。尽管应用程序提供商很难防止篡改版本的重新分发，负面宣传很可能会指向原始提供商。因此，应尽可能让攻击者难以重新分发未经授权的副本，以降低这种风险的概率。

7.6 我是否易受“二进制保护不足”的影响？

所有应用都容易受到二进制攻击。如果应用将敏感数据或算法硬编码在其二进制文件中，或如果该应用非常流行，二进制攻击可能尤其有害。如果有额外的保护措施，例如混淆、在本地代码中编码秘密（对于 Android）或类似的措施，成功的攻击会变得更加困难，但永远不可能完全防止。

应用是否足够安全取决于不同二进制攻击可能带来的业务影响。攻击者的动机越大，潜在影响越大，应该投入更多精力进行保护。因此，应用对二进制攻击的“脆弱性”是高度特定于该应用的。

开发人员可以通过使用类似攻击者使用的工具来快速检查自己的应用程序二进制文件。现在有许多免费的或价格合理的工具，例如 MobSF、otool、apktool 和 Ghidra，它们都很容易使用且文档齐全。

7.7 如何防止“二进制保护不足”？

对于每个应用，应评估二进制文件中是否包含任何关键内容，或者其流行性判断是否需要加强二进制保护。如果确实存在此类需求，可以通过威胁建模分析来帮助识别最高风险及其发生时可能带来的财务影响。对于已识别的最相关的风险，应采取相对应的反制措施。

由于应用程序总是在不受信的执行环境中运行，因此只应获取运行所需的最少信息，以降低信息泄露或被篡改的风险。但若某些秘密、算法、安全检查等必须包含在应用的二进制文件中，可以通过不同的手段来抵御不同类型的攻击：

- **逆向工程**：为了防止应用程序二进制文件被逆向工程，可以利用多种技术手段来增加其理解难度。许多免费和商业的混淆工具都能为此提供帮助。部分应用程序通过原生编译（iOS 和 Android）或使用解释器或嵌套虚拟机加大逆向工程的难度，因为许多反编译工具只支持一种语言和二进制格式。这种混淆需要在代码复杂性和防逆向工程健壮性之间找到平衡，因为许多依赖代码中某些字符串或符号的库在完全混淆的情况下无法工作。因此，开发人员可以使用前面提到的工具检查混淆的质量。
- **破坏安全机制**：混淆也有助于防止篡改，因为攻击者必须理解控制流才能跳过安全检查。此外，后端也应强制执行本地安全检查。例如，受保护功能所需资源只有在本地和后端检查成功时才应下载。最后，完整性检查可以检测代码篡改并使应用程序安装不可用，例如通过删除某些资源。然而，这样的完整性检查也可能被发现并禁用，就像任何其他本地安全检查一样。
- **重新分发（带恶意代码）**：启动时的完整性检查也可以检测应用程序二进制文件的重新分发和修改。这些违规行为可以自动报告，以便在应用商店变得广泛传播之前找到并删除未经授权的副本。也有专业公司提供这种用例的支持。

7.8 攻击场景示例

场景#1 硬编码 API 密钥	场景#2 禁用支付和许可检查	场景#3 硬编码 AI 模型
假设一个应用使用商业 API，每次调用均需少量付费。这些费用通常由用户支付的订阅费来支付。然而，用于访问和计费的 API 密钥硬编码在应用的未保护二进制代码中。攻击者可以使用免费工具对应用进行逆向工程，从而获取秘密字符串。由于 API 访问仅受 API 密钥保护，缺乏额外的用户身份验证，攻击者可以无限制地使用 API，甚至出售 API 密钥牟利。在最坏的情况下，API 密钥可能被大量滥用，给应用提供商造成巨大的财务损失，或在 API 访问被限速的情况下影响应用的合法用户。	一款移动游戏可能会发布其应用及前几个关卡供免费使用。如果用户喜欢游戏，他们支付费用以获得完整访问权限。所有后续关卡的资源都包含在应用中，它们仅受许可检查保护，许可证在用户支付时下载。攻击者可以逆向工程应用并试图了解支付验证是如何进行的。如果应用二进制文件未充分保护，攻击者很容易找到许可检查并将其替换为静态的成功声明。攻击者随后可以重新编译应用并免费玩游戏，甚至将其以其他名称在应用商店中销售。	假设一款医疗应用包含一个 AI 模型，用于回答用户的语音或自由文本输入需求。该应用将其专门的、经过质量保障的 AI 模型包含在源代码中，以便离线访问，并避免自行托管下载服务器。这个 AI 模型是该应用最有价值的资产，开发过程中耗费了大量人力。攻击者可能尝试从源代码中提取这个模型并将其出售给竞争对手。如果应用二进制文件保护不足，攻击者不仅可以访问 AI 模型，还能了解它是如何使用的，并将这些信息连同 AI 训练参数一起出售。

M8：安全配置错误

8.1 威胁主体

特定应用程序

移动应用中的安全配置错误是指安全设置、权限和控制配置不当，可能导致漏洞和未经授权的访问。能够利用安全配置错误的威胁主体通常是旨在获取未经授权的敏感数据或执行恶意操作的攻击者。这些威胁主体可能是具有物理访问权限的攻击者，或是设备上的恶意应用，利用安全配置错误在目标应用上下文中执行未经授权的操作。

8.2 攻击向量

可利用性：困难

移动应用程序中的安全配置错误可以通过各种攻击向量被利用，包括：

- **不安全的默认设置**：移动应用程序通常带有默认配置，这些配置可能存在弱安全设置或启用了不必要的权限，使其容易受到攻击。
- **不当的访问控制**：配置不当的访问控制可能会允许未经授权的用户访问敏感数据或执行特权操作。
- **弱加密或哈希算法**：实施不当或弱加密和哈希算法可能会被利用以获取对敏感信息的访问。
- **缺乏安全通信**：未能使用安全通信协议（如 SSL/TLS）可能暴露敏感数据，受到窃听或中间人攻击。
- **未受保护的存储**：以不安全的方式存储敏感数据（如密码或 API 密钥），例如明文或弱加密，可能会导致未经授权的访问。
- **不安全的文件权限**：以全局可读和/或全局可写权限存储应用程序文件。
- **配置不当的会话管理**：不当的会话管理可能会导致会话劫持，使攻击者能够冒充合法用户。

8.3 安全弱点

常见程度：常见　　可检测性：容易

由于开发过程中时间限制、缺乏意识或人为错误等因素，安全配置错误在移动应用程序中很常见。通过手动代码审查、安全测试或自动扫描工具，检测安全配置错误相对容易。

安全配置错误的示例包括：

- 未能在发布版本中禁用调试功能，这可能会暴露敏感信息。
- 允许不安全的通信协议（如 HTTP），而不是强制通过 HTTPS 进行安全通信。
- 未更改默认用户名和密码，为攻击者提供轻松访问的机会。
- 不足的访问控制允许未经授权的用户执行特权操作。

8.4 技术影响

影响：严重

安全配置错误对移动应用程序可能产生重大技术影响，包括：

- **对敏感数据的未经授权访问**：配置错误可能会允许攻击者访问敏感信息，如用户凭据、个人数据或机密业务数据。
- **账户劫持或冒充**：弱或配置错误的身份验证机制可能导致账户接管或冒充合法用户。
- **数据泄露**：不足的安全配置可能导致数据泄露，将敏感数据暴露给未经授权的个人。
- **后端系统的攻破**：移动应用程序中的配置错误可能为攻击者提供突破口，以攻破后端系统或基础设施。

8.5 业务影响

影响：严重

安全配置错误可能产生严重的业务影响，包括：

- **财务损失**：由安全配置错误导致的违规可能导致财务损失，包括法律处罚、监管罚款以及对组织声誉的损害。
- **数据丢失或盗窃**：配置错误可能导致敏感数据的丢失或盗窃，从而导致法律和财务后果。
- **停机和中断**：利用安全配置错误可能导致应用程序停机、服务中断或功能受损，影响用户体验和业务运营。
- **品牌声誉受损**：公开披露的安全事件可能损害组织的声誉，导致客户信任丧失和潜在的业务损失。

8.6 我是否容易受到安全配置错误的影响?

如果移动应用程序没有正确配置以遵循安全最佳实践，则容易受到安全配置错误的影响。常见的安全配置错误指标包括：

- **未审查默认设置**：使用默认配置而不审查安全设置、权限和默认凭据。
- **缺乏安全通信**：使用未加密或弱加密的通信渠道。
- **弱或缺失的访问控制**：允许未经授权访问敏感功能或数据。
- **未能更新或修补**：未应用必要的安全更新或补丁。
- **敏感数据存储不当**：以明文或弱保护格式存储敏感数据。
- **不安全的文件提供者路径设置**：本应仅供内部使用的文件内容提供者暴露给其他应用或用户，可能导致敏感数据泄露或未经授权的资源访问。
- **导出的活动**：本应仅供内部使用的活动被导出和/或可以浏览，暴露了额外的攻击面。

要确定您的应用程序是否容易受到安全配置错误的影响，您应该进行全面的安全评估，包括代码审查、安全测试和配置分析。

8.7 如何防止安全配置错误?

防止移动应用中的安全配置错误需要遵循安全编码和配置最佳实践。以下是一些关键的防范措施：

- **安全的默认配置**：确保默认设置和配置得到妥善保护，不暴露敏感信息或提供不必要的权限。
- **默认凭据**：避免使用硬编码的默认凭据。

- **不安全的权限**：避免以过于宽松的权限（如全局可读和/或全局可写）存储应用程序文件。
- **最小特权原则**：仅请求应用程序正常运行所需的功能权限。
- **安全的网络配置**：禁止明文流量并尽可能使用证书锁定。
- **禁用调试**：在应用程序的发布版本中禁用调试功能。
- **禁用备份模式（Android）**：通过在 Android 设备上禁用备份模式，您可以防止应用程序数据包含在设备的备份中，确保应用程序的敏感数据不会存储在设备备份中。
- **限制应用程序攻击面**：仅导出必需的活动、内容提供者和服务。

8.8 攻击场景示例

以下场景展示了移动应用程序中的安全配置错误：

场景#1：不安全的默认设置	场景#2：不安全的文件提供者路径设置	场景#3：过于宽松的存储权限
一款移动应用发布时使用了默认设置，这些设置启用了弱安全配置，包括使用不安全的通信协议、未更改默认用户名和密码，并且没有在发布版本中禁用调试功能。攻击者利用这些配置错误获得未经授权的访问权限，访问敏感数据或执行恶意操作。	一款移动应用在导出的文件内容提供者中暴露了其根路径，允许其他应用访问其资源。	一款移动应用将应用共享首选项以全局可读权限存储，允许其他应用读取这些数据。

场景#4：导出的活动	场景#5：不必要的权限
一款移动应用程序导出了一些仅供内部使用的活动，为攻击者提供了额外的攻击面。	一款移动应用程序请求超出其核心功能所需地过多权限。例如，一个简单的手电筒应用程序请求访问用户的联系人、位置和相机。这使用户数据面临不必要的风险，因为应用程序可能会滥用授予的权限或无意中泄露敏感信息。

M9：数据存储不安全

9.1 威胁主体

特定应用程序

移动应用中的不安全数据存储可能吸引各种威胁主体，他们旨在利用漏洞并获得未经授权的敏感信息。这些威胁主体包括：针对移动应用提取有价值数据的技术娴熟的攻击者；滥用其权限的恶意内部人员；进行网络间谍活动的国家支持的行为者；通过数据盗窃或勒索谋取经济利益的网络犯罪分子；利用预建工具进行简单攻击的脚本小子；利用不安全存储以出售个人信息的数据经纪人；旨在获取竞争优势的竞争对手和工业间谍；以及具有意识形态动机的黑客。

这些威胁主体利用诸如弱加密、数据保护不足、不安全的数据存储机制以及用户凭证处理不当等漏洞。对于移动应用开发者和组织来说，实施强有力的安全措施至关重要，如强加密、安全的数据存储实践和遵循移动应用安全最佳实践，以降低与不安全数据存储相关的风险。

9.2 攻击向量

可利用性：容易

移动应用中的不安全数据存储暴露了多个攻击途径，攻击者可以利用这些途径进行攻击。攻击途径包括通过物理或远程手段未经授权访问设备的文件系统，利用弱加密或缺乏加密，拦截数据传输，利用安装在设备上的恶意软件或恶意应用。此外，越狱或破解的设备为攻击者提供了绕过安全措施并直接访问敏感数据的机会。其他攻击途径包括社交工程学技术，诱使用户提供对其数据的访问或操纵应用程序的行为。

总的来说，移动应用中不安全的数据存储为从直接数据提取到敏感信息拦截的各种攻击提供了多种途径，突显了强加密、安全传输协议和移动应用开发中全面安全措施的关键需求。

9.3 安全弱点

常见程度：常见　　可检测性：平均

移动应用中的不安全数据存储包括多种安全弱点，可能危及存储信息的机密性和完整性。这些弱点包括使用弱加密或根本没有加密，使攻击者容易访问和解密敏感数据。此外，数据存储在设备文件系统中容易访问的位置，如明文文件或未加密的数据库，也使数据容易被未经授权提取或篡改。缺乏适当的访问控制和用户身份验证机制进一步加剧了问题，使未经授权的人员能够访问敏感数据。

此外，缺乏安全的数据传输协议使数据在移动应用与外部服务器通信时容易受到拦截。总的来说，这些安全弱点在移动应用的数据存储中为数据泄露、未经授权的访问和数据篡改创造了机会，突显了强加密、安全存储实践和严格的访问控制的重要性，以减少这些风险。

9.4 技术影响

影响：严重

移动应用中的不安全数据存储可能对系统的整体安全性和功能产生显著的技术影响。具体影响包括

- **数据泄露：**不安全的数据存储使敏感信息容易受到未经授权的访问和数据泄露。攻击者可以利用漏洞提取或篡改敏感数据，导致潜在的隐私违规和机密信息丧失。

- **用户账户被破坏**：不当的数据存储实践可能导致用户账户被破坏。攻击者可能获得对存储不安全的登录凭据或个人信息的访问权限，导致未经授权的账户访问、身份盗用或冒充用户执行未经授权的操作。
- **数据篡改和完整性问题**：没有适当的数据保护措施，攻击者可以修改或篡改存储的数据。这可能导致数据完整性问题、不准确的信息或将恶意内容注入应用程序的数据存储中。
- **未经授权访问应用程序资源**：不安全的数据存储可能使攻击者能够未经授权访问应用的关键资源。这包括存储在应用程序中的敏感文件、配置文件或加密密钥，它们可以被利用来破坏应用程序的功能或利用其底层系统。
- **声誉和信任损害**：如果发现应用程序存在不安全的数据存储，将严重损害应用程序开发人员或组织的声誉和信任。用户可能会失去对应用程序安全性的信心，导致用户采用率下降和潜在的法律和监管后果。
- **合规违规**：不安全的数据存储可能导致不符合行业规定和数据保护标准。应用开发者如果未能充分保护用户数据并维持安全的数据存储实践，可能面临处罚或法律诉讼。

9.5 业务影响

影响：严重

不安全数据存储对移动应用的业务影响可能是显著且广泛的。以下是一些关键的业务影响：

- **声誉损害**：不安全的数据存储可能导致数据泄露和用户账户劫持，这会严重损害组织的声誉和信任。数据泄露的消息可能迅速传播，导致负面宣传、客户不满以及潜在的业务损失。
- **客户信任丧失**：当敏感的客户数据由于不安全的数据存储而被泄露时，客户可能会失去对组织保护其信息能力的信任。这种信任的丧失可能导致客户忠诚度下降、客户流失增加，并对整体客户满意度产生负面影响。
- **法律和监管后果**：不当的数据存储实践可能导致不符合行业规定和数据保护法律。不充分的数据存储实践可能导致不符合行业规定和数据保护法律。组织可能面临法律后果，包括罚款、处罚或因未能充分保护用户数据而提起的诉讼。合规违规也可能损害组织在客户和业务合作伙伴眼中的声誉和可信度。
- **财务影响**：数据泄露及其后果可能对组织产生重大财务影响。这包括调查泄露事件、通知受影响客户、提供身份盗窃保护服务、潜在的法律和解决费用，以及业务机会的丧失。
- **竞争劣势**：在当今高度竞争的市场环境中，遭遇数据泄露或因不安全数据存储而声誉受损的组织可能面临竞争劣势。客户越来越关心其数据的安全性，他们可能选择那些在保护敏感信息方面有更好记录的竞争对手。

9.6 我是否容易受到“数据存储不安全”的影响?

移动应用程序中的不安全数据存储和意外的数据泄漏可以通过多种方式表现出来，从而导致潜在的隐私泄露和对敏感信息的未经授权访问。以下是这些问题的一些常见表现：

- **缺乏访问控制**：应用程序内的不充分访问控制可能允许未经授权的用户或攻击者访问存储在设备或应用程序数据库中的敏感数据。
- **加密不足**：未正确加密敏感数据，可能导致在攻击者访问存储位置时无意间泄露数据。没有加密，数据容易被读取并被利用。
- **无意的数据暴露**：移动应用可能通过应用日志、错误信息或调试功能无意中暴露敏感数据，允许未经授权的人员查看或捕获敏感信息。

- **会话管理不当**：弱会话管理可能导致无意的数据泄露。如果会话令牌或用户身份验证信息未得到适当保护或管理，可能会被拦截或篡改，从而允许未经授权访问敏感数据。
- **输入验证不足**：输入验证和数据清理不足可能导致无意的数据泄露。攻击者可能利用此弱点，通过操控输入字段注入恶意脚本或检索敏感数据。
- **云存储配置错误**：如果移动应用使用云存储服务存储数据，并且配置管理不当或配置错误，可能导致存储数据的无意暴露或未经授权的访问。
- **第三方库漏洞**：移动应用中使用的不安全第三方库可能存在漏洞，导致无意的数据泄露。攻击者可以利用这些漏洞获取未经授权的敏感信息。
- **意外的数据共享**：应用内的数据共享功能处理不当，可能导致无意的数据泄露。如果敏感数据与不该接收的人共享，或共享过程未得到充分保护，可能导致隐私违规。

9.7 如何防止“数据存储不安全”？

为了防止移动应用中的不安全数据存储并确保敏感数据的保护，应实施以下安全措施：

- **使用强加密**：实施强大的加密算法和实践，以保护静态数据和传输中的数据。使用行业标准的加密算法，并确保加密密钥得到安全存储和管理。
- **安全数据传输**：使用安全的通信协议（例如 HTTPS、SSL/TLS）保护移动应用程序与后端服务器之间传输的数据。避免通过不安全的通道发送敏感数据。
- **实施安全存储机制**：将敏感数据存储在未经授权用户无法访问的安全存储位置。使用移动操作系统提供的平台特定的安全存储机制，例如 iOS 的 Keychain 或 Android 的 Keystore。
- **实施适当的访问控制**：实施强大的访问控制以限制对敏感数据的未经授权访问。安全地验证用户身份，执行基于角色的访问控制，并在授予访问权限之前验证用户权限。
- **验证输入并清理数据**：实施输入验证和数据清理技术，以防止注入攻击，并确保仅存储有效且预期的数据。验证用户输入，以减少恶意代码注入或无意数据泄露的风险。
- **应用安全的会话管理**：实施安全的会话管理技术，如使用随机生成的会话令牌、设置适当的会话超时，并安全地存储客户端和服务器端的会话数据。
- **定期更新和修补依赖项**：保持所有库、框架和第三方依赖项的最新版本，因为它们可能包含导致不安全数据存储的安全漏洞。定期应用供应商提供的安全修补程序和更新。
- **保持信息更新**：保持对移动应用领域最新安全威胁和漏洞的了解。监控安全论坛、安全公告和移动平台更新，确保及时应对新出现的风险。

9.8 攻击场景示例

以下是展示移动应用中不安全数据存储的几个示例场景：

场景#1：以纯文本形式存储密码：	场景#2：不安全的本地存储：	场景#3：不安全的数据缓存：
移动应用将用户密码以明文形式存储在本地数据库或文件中，使得攻击者一旦获得设备的未经授权访问权限，就能轻易地检索并滥用这些凭证。	移动应用将敏感用户数据（如个人身份信息 PII）存储在设备本地，但未使用适当的访问控制或加密。这样，任何具有物理访问设备的人都可以提取和查看这些数据。	移动应用缓存敏感数据（如用户身份验证令牌或会话信息），但未实施适当的安全措施。如果攻击者获得设备缓存的访问权限，他们可以获取这些凭证并冒充用户。
场景#4：不受保护的日志记录：	**场景#5：不安全的云存储配置：**	**场景#6：临时文件处理不当：**
移动应用程序记录敏感数据，包括用户操作、API 响应或错误消息，而没有适当的安全控制。如果攻击者获得对设备的访问权限或拦截日志文件，可能会无意中暴露敏感信息。	移动应用程序使用云存储服务存储用户数据，但配置存储权限错误，允许未经授权访问存储的信息。这可能导致数据泄露或敏感数据的未经授权暴露。	移动应用创建临时文件以处理或存储敏感数据，但未妥善处理和删除这些文件，导致敏感信息暴露并容易受到未经授权的访问。

M10:加密措施不足

10.1 威胁主体

特定应用程序

利用移动应用中不安全加密的威胁主体可能会破坏敏感信息的机密性、完整性和真实性。这些威胁主体包括：攻击加密算法或实现以解密敏感数据的攻击者；在组织内或应用开发团队中，滥用其权限或泄露加密密钥的恶意内部人员；进行网络破解以获取情报的国家支持的行为者；利用弱加密窃取有价值数据或进行金融欺诈的网络犯罪分子；利用加密协议或库漏洞的攻击者。

10.2 攻击向量

可利用性：一般

不安全加密的攻击途径涉及利用保护敏感信息的加密机制中的漏洞。攻击者可能会采用多种技术，例如密码攻击、暴力破解攻击或侧信道攻击，利用加密算法、密钥管理或实现缺陷中的弱点。通过针对不安全加密，攻击者旨在解密加密数据、篡改加密过程或未经授权访问敏感信息。这可能导致数据泄露、未经授权访问用户账户、机密性受损或伪造或篡改数据。

10.3 安全弱点

常见程度：常见　　可检测性：一般

移动应用中的不安全加密引入了可能破坏加密措施有效性并危及敏感数据机密性和完整性的安全弱点。这些弱点可能包括使用弱加密算法或不足的密钥长度、不良的密钥管理实践、不当处理加密密钥、不安全的随机数生成、加密协议的错误实现，或加密库或框架中的漏洞。攻击者可以利用这些弱点绕过加密、执行密码攻击、篡改数据或获取加密信息的未经授权访问。不安全的哈希函数和加密算法是移动应用中显著的安全弱点。这些漏洞可能导致严重的数据泄露和未经授权访问敏感信息。当使用过时或弱的哈希函数时，攻击者可以利用漏洞反向工程哈希数据，揭示原始内容。为了保护移动应用免受这些安全风险，必须采用强大且现代的哈希函数和加密算法，并遵循加密和密钥管理的最佳实践，以确保数据的完整性和机密性。定期的安全审计和更新对于维持最高级别的保护以抵御潜在威胁也至关重要。

10.4 技术影响

影响：严重

此漏洞将导致从移动设备未经授权地检索敏感信息。

10.5 业务影响

影响：严重

移动应用中的不充分加密或不安全哈希函数可能对业务产生重大影响。以下是一些潜在后果：

- **数据泄露：**弱加密或加密措施不足使得攻击者更容易破坏敏感数据的机密性，导致数据泄露，暴露敏感的客户信息，如个人身份信息（PII）、财务细节或知识产权。这类泄露可能导致法律责任、监管处罚、客户信任丧失以及声誉损害。

- **知识产权损失：**加密措施不足可能危及嵌入在移动应用中的专有算法、商业秘密或其他知识产权的保护。如果攻击者能够解密并提取这些有价值的信息，可能被竞争对手利用以获得竞争优势，或者在黑市上出售。
- **财务损失：**加密措施不足可能通过多种方式导致财务损失。例如，如果支付交易或财务数据没有正确加密，可能导致客户遭受欺诈和未经授权访问其资金。此外，调查和修复安全漏洞的费用、通知受影响客户的费用、提供身份盗窃保护服务的费用以及潜在的法律和解决费用可能非常高。
- **合规与法律后果：**许多行业有特定的数据保护和隐私法规，要求对敏感信息使用强加密。不足的加密可能导致不合规，从而导致法律后果、罚款或监管机构的制裁。

10.6 我是否易受“加密措施不足”影响?

不安全加密和不安全哈希函数在移动应用中可能以几种方式表现出来：

1) **弱加密算法：**移动应用可能使用已知的弱加密算法或易受到攻击的算法。这些算法可能存在已知的弱点、过时，或缺乏足够的安全性，无法有效保护敏感数据。
2) **密钥长度不足：**不足的密钥长度会削弱加密强度。如果移动应用使用短或容易猜测的加密密钥，则攻击者通过暴力破解或其他密码攻击解密加密数据变得更容易。
3) **密钥管理不当：**不良的密钥管理实践，如不安全地存储加密密钥或以明文形式传输它们，可能使密钥暴露于未经授权的访问。获得密钥访问权限的攻击者可以轻松解密数据。
4) **加密实现缺陷：**加密/解密过程本身可能实现不正确或包含编程错误。这些实现错误可能引入攻击者可利用以绕过或削弱加密保护的漏洞。
5) **数据/加密密钥存储不安全：**如果加密密钥在移动设备上存储不安全，例如以明文形式或存储在易于访问的位置，则具有物理或未经授权访问设备的攻击者可以检索密钥并解密受保护的数据。移动应用可能使用弱加密算法或错误使用加密，例如使用弱密钥或未能正确加密所有敏感数据。如果加密被攻击者轻易绕过或解密，这可能导致数据泄露。
6) **缺乏安全传输层：**在网络上传输加密数据时，使用安全传输层协议（如 HTTPS）至关重要。如果移动应用未能实现安全传输协议，则加密数据可能在传输过程中易受拦截或篡改。
7) **验证和认证不足：**加密过程中涉及的各方验证和认证不足可能削弱整体安全性。如果没有适当的验证，攻击者可以冒充合法实体、拦截加密数据并在不被发现的情况下操纵它。
8) **缺乏加盐：**加盐是指在哈希之前向输入添加随机数据的过程，这对于增强密码的安全性至关重要。不安全的哈希函数可能不支持加盐或使用弱加盐方法，使密码哈希容易受到预计算表或暴力破解攻击。

10.7 如何防止“加密措施不足”？

为了防止移动应用中的“加密措施不足”漏洞，请考虑以下最佳实践：

1) **使用强加密算法：**实现广泛接受且安全的加密算法，如 AES（高级加密标准）、RSA（Rivest-Shamir-Adleman）或 ECC（椭圆曲线加密）。保持对当前加密标准的同步，并避免使用已弃用或弱化的算法。
2) **确保密钥长度足够：**选择适当长度的加密密钥以确保强大的加密强度。遵循针对所使用具体加密算法的行业推荐密钥长度。
3) **遵循安全密钥管理实践：**采用安全的密钥管理技术，如使用密钥库或硬件安全模块（HSM）来安全地存储加密密钥。保护密钥不受未经授权的访问，包括限制仅授权人员访问、加密密钥并确保安全的密钥分发机制。
4) **正确实施加密：**在移动应用中仔细实现加密和解密过程，遵循已建立的加密库和框架。避免使用自定义加密实现，因为它们更容易出错和存在漏洞。
5) **安全存储加密密钥：**确保加密密钥在移动设备上安全存储。避免以明文存储密钥或将其存储在易于访问的位置。考虑使用操作系统提供的安全存储机制或利用硬件加密存储选项。
6) **使用安全的传输层协议：**使用安全的传输层协议，如 HTTPS（安全的 HTTP），通过网络传输加密数据。实施适当的证书验证，确保移动应用与后端系统之间的安全通信通道。
7) **验证和认证：**实施强有力的验证和认证机制，以验证参与加密过程各方的完整性和真实性。对证书、数字签名或其他认证机制进行正确验证。
8) **定期更新安全措施：**关注来自加密库、框架和平台提供商的安全更新、补丁和建议。保持移动应用和底层加密组件的更新以解决任何已识别的漏洞或弱点。
9) **进行安全测试：**执行全面的安全测试，包括密码漏洞评估、渗透测试和代码审查。识别并修复测试过程中发现的任何弱点或漏洞。
10) **遵循行业标准和最佳实践：**关注与加密相关的行业标准和最佳实践。像 NIST（美国国家标准与技术研究院）和 IETF（互联网工程任务组）这样的组织提供了安全加密实践的指南和建议。
11) **使用强哈希函数：**选择广泛认可且加密安全的哈希函数，如 SHA-256 或 bcrypt。这些算法旨在抵御攻击并提供高水平的安全性。
12) **实现加盐：**在哈希密码时始终使用强随机盐。加盐通过使攻击者更难使用预计算表或彩虹表破解密码而增加额外的安全层。
13) **使用密钥派生函数（KDFs）：**用于密码哈希的密钥派生函数（如 PBKDF2、bcrypt 或 scrypt）专门设计用于安全地从密码派生加密密钥，并提供额外的安全特性，如迭代计数，以减缓暴力破解攻击。

10.8 攻击场景示例

场景#1：中间人（MitM）攻击	场景#2：暴力破解攻击	场景#3：密码降级攻击
攻击者拦截移动应用与服务器之间的通信。弱加密使得攻击者能够解密拦截的数据，修改数据，并在转发给预定接收者之前重新加密。这可能导致未经授权的访问、数据篡改或恶意内容注入。	攻击者系统地尝试各种密钥组合，直到找到正确的密钥以解密数据。弱加密会缩短此类攻击所需的时间，可能暴露敏感信息。	移动应用可能支持多种加密协议或算法以建立安全连接。如果允许弱加密作为回退选项，攻击者可以利用此弱点迫使应用使用弱加密。结果是他们可以更容易地解密拦截的数据并发起后续攻击。

场景 #4：密钥管理漏洞	场景 #5：加密实现缺陷
弱的密钥管理实践可能破坏移动应用中使用的加密系统的安全性。例如，如果加密密钥存储不安全或容易猜测，攻击者可以获得未经授权的密钥访问并解密加密数据。这可能导致数据泄露和隐私违规。	弱加密也可能源于移动应用本身的实现缺陷。这些缺陷可能包括加密库的错误使用、不安全的密钥生成、不恰当的随机数生成或加密相关功能的不安全处理。攻击者可以利用这些缺陷绕过或削弱加密保护。