OWASP移动应用程序安全检查表

MAS Checklist

OWASP MASTG v1.7.0

OWASP MASVS v2.0.0

前 言

关于OWASP Mobile Application Security项目

- 由Carlos Holguera和Sven Schleier领导的OWASP移动应用安全（MAS）旗舰项目为移动应用提供了安全标准（OWASP MASVS）和全面的测试指南（OWASP MASTG）。
- OWASP MASVS(移动应用程序安全验证标准)是一个为移动应用程序安全性建立安全要求的标准。
- OWASP MASTG(移动应用程序安全测试指南)是一个全面的移动应用程序安全测试和逆向工程手册。它描述了验证MASVS中列出的控制的技术过程。
- OWASP MAS Checklist （Mobile Application Security Checklist，移动应用程序安全检查表），包含每个MASVS控件的MASTG测试用例的链接。

MAS Checklist

每个MASVS控件的MASTG测试用例的链接。该清单有助于：

- 安全评估/渗透测试：确保至少覆盖了标准的攻击面，并开始探索。
- 标准合规性：包括MASVS和MASTG版本和提交ID。
- 学习并练习移动安全技能。
- Bug赏金：逐步覆盖移动攻击面。

中文版项目组

感谢所有的翻译人员，慷慨地自愿贡献自己的时间和专业知识，使OWASP中国社区能够快速访问MASVS。我们衷心感谢您的贡献，并希望在未来继续合作。
MASVS 2.0 CHECK LIST 中文翻译人员：张坤、孙龙

反馈

- 如果您有任何意见或建议，请在我们的GitHub讨论中提出。
- https://github.com/OWASP/owasp-mastg/discussions/categories/ideas

版权

MASVS-STORAGE: 存储

MASVS-ID	平台	描述	L1	L2	R	状态
MASVS-STORAGE-1		**应用程序安全地存储敏感数据。**				
	android	测试本地储存对于敏感数据的存储机制				
	android	测试设备访问的安全策略				
	ios	测试本地数据存储情况				
MASVS-STORAGE-2		**应用程序可防止敏感数据泄漏。**				
	android	确定敏感数据是否通过嵌入服务与第三方共享				
	android	是否通过通知与第三方共享敏感数据				
	android	测试备份中的敏感数据				
	android	测试内存中的敏感数据				
	android	确定是否禁用键盘缓存文字输入位置				
	android	检测日志中的敏感数据				
	ios	检测日志中的敏感数据				
	ios	测试内存中的敏感数据				
	ios	测试备份中的敏感数据				
	ios	检测键盘缓存中的敏感数据				
	ios	确定是否与第三方共享敏感数据				

MASVS-CRYPTO: 密码学

MASVS-ID	平台	描述	L1	L2	R	状态
MASVS-CRYPTO-1		应用程序采用了当前强大的加密技术，可以根据行业的最佳实践来使用它。				
	android	对称密码检测				
	android	测试配置的标准算法加密的情况				
	android	测试随机生成器				
	ios	验证密码标准算法的配置				
	ios	测试随机生成器				
MASVS-CRYPTO-2		应用程序根据行业最佳实践执行密钥管理。				
	android	测试秘钥的用途				
	ios	测试密钥管理				

MASVS-AUTH: 认证和授权

MASVS-ID	平台	描述	L1	L2	R	状态
MASVS-AUTH-1		应用程序使用安全身份验证和授权协议,并遵循相关的最佳实践。				
MASVS-AUTH-2		应用程序根据平台最佳实践安全执行本地身份验证。				
	android	测试生物识别认证				
	android	测试证书情况				
	ios	测试本地身份验证				
MASVS-AUTH-3		应用程序通过额外的身份验证保护敏感操作。				

MASVS-NETWORK: 网络通讯

MASVS-ID	平台	描述	L1	L2	R	状态
MASVS-NETWORK-1		**应用程序根据当前最佳实践保护所有网络流量。**				
	android	测试终端身份验证				
	android	测试安全提供程序				
	android	测试在网络通讯时数据加密情况				
	android	测试TLS设置				
	ios	测试TLS设置				
	ios	测试终端身份验证				
	ios	测试在网络通讯时数据加密情况				
MASVS-NETWORK-2		**应用程序在开发人员的控制下为所有远程终端执行身份固定。**				
	android	测试自定义证书存储和证书固定				
	ios	测试自定义证书存储和证书固定				

MASVS-PLATFORM: 联动平台

MASVS-ID	平台	描述	L1	L2	R	状态
MASVS-PLATFORM-1		**应用安全使用IPC机制。**				
	android	应用程序权限检测				
	android	PendingIntent脆弱性测试				
	android	通过IPC进行敏感功能暴露测试				
	android	确定是否有通过IPC机制暴露敏感存储数据				
	android	内部链接检测				
	ios	应用扩展检测				
	ios	自定义URL方案检测				
	ios	测试应用程序权限				
	ios	确定是否有通过IPC机制公开敏感数据				
	ios	用户页面粘贴板测试				
	ios	通过IPC进行敏感功能暴露测试				
	ios	测试UIActivity共享				
	ios	测试通用链接				
MASVS-PLATFORM-2		**应用程序安全使用WebView。**				
	android	测试webview协议处理程序				
	android	测试WebView清理				
	android	测试JavaScript在WebView中的执行				
	android	测试通过WebView公开的Java对象				
	ios	测试Webview协议处理程序				
	ios	测试ios webviews				
	ios	确定原生方法是否有通过webviews公开				
MASVS-PLATFORM-3		**应用程序安全使用用户界面。**				
	android	覆盖攻击测试				
	android	检查通过用户界面披露的敏感数据				
	android	查找自动生成的截图中是否包含敏感信息				
	ios	检查通过用户界面披露的敏感数据				
	ios	测试自动生成的截图是否包含敏感信息				

MASVS-CODE: 代码质量

MASVS-ID	平台	描述	L1	L2	R	状态
MASVS-CODE-1		**应用程序需要最新的平台版本。**				
MASVS-CODE-2		**应用程序有一个机制来强制执行应用程序更新。**				
	android	测试强制更新				
	ios	测试强制更新				
MASVS-CODE-3		**应用程序只使用没有已知漏洞的软件组件。**				
	android	检查第三方库中的风险				
	ios	检查第三方库中的风险				
MASVS-CODE-4		**应用验证和清理所有不受信任的输入。**				
	android	确保免费安全功能已激活				
	android	测试注入风险				
	android	对本地存储进行输入验证测试				
	android	内存损坏错误				
	android	测试对象持久性				
	android	测试隐式Intents				
	android	测试WebView中的URL加载				
	ios	测试对象持久性				
	ios	内存损坏错误				
	ios	确保免费安全功能已激活				

MASV-RESILIENCE:逆向工程对抗和防篡改的能力

MASVS-ID	平台	描述	L1	L2	R	状态
MASVS-RESILIENCE-1		**应用程序验证平台的完整性。**				
	android	ROOT环境检测				
	android	模拟器环境检测				
	ios	越狱环境检测				
	ios	模拟器环境检测				
MASVS-RESILIENCE-2		**应用程序实现防篡改机制。**				
	android	文件的完整性检查				
	android	运行环境完整性检测				
	android	重签名检测				
	ios	文件完整性检测				
	ios	重签名检测				
MASVS-RESILIENCE-3		**应用程序实现反静态分析机制（反编译）。**				
	android	调试符号测试				
	android	调试代码和详细错误日志的测试				
	android	代码混淆测试				
	ios	调试代码和详细错误日志的测试				
	ios	代码混淆测试				
	ios	调试符号测试				
MASVS-RESILIENCE-4		**应用程序实现反动态分析技术（反调试）。**				
	android	测试的应用程序是否可调试				
	android	逆向工程工具检测				
	android	反调试检测				
	ios	反调试检测				
	ios	测试应用程序是否可调试				
	ios	逆向工程工具检测				

列表1

MASVS-STORAGE: 存储	MASVS-STORAGE-1	安全存储敏感数据	应用程序处理来自许多来源的敏感数据，例如用户、后端、系统服务或设备上的其他应用程序。通常需要将这些敏感数据存储在本地。存储位置可能是应用程序私有的（例如其内部存储），也可能是可被用户或其他已安装的应用程序（例如下载等公共文件夹）公开访问的位置。这种控制措施可确保应用程序有意存储的任何敏感数据得到适当的保护，不受目标位置的影响。
	MASVS-STORAGE-2	防止敏感数据泄露	在某些情况下，敏感数据会无意中存储或暴露在可公开访问的位置。通常这种情况是由于使用某些API、备份或日志等系统功能造成的副作用。这种控制措施应涵盖这种无意泄漏。实际上，开发人员有办法防止这种泄漏。
MASVS-CRYPTO: 密码学	MASVS-CRYPTO-1	采用当前强大的加密技术，并根据行业最佳实践使用。	密码学在保护用户数据方面起着特别重要的作用，尤其是在移动环境中，攻击者是有可能物理访问用户设备的。这种控制措施应涵盖在外部典型标准中定义的一般密码学最佳实践。
	MASVS-CRYPTO-2	根据行业最佳实践执行密钥管理。	即使是最强大的密码术也会受到糟糕的密钥管理的危害。这种控制措施应涵盖加密密钥在其整个生命周期内的管理，包括密钥生成、存储和保护。
MASVS-AUTH: 认证和授权	MASVS-AUTH-1	使用安全身份验证和授权协议，并遵循相关的最佳实践。	大多数连接到远程终端的应用程序都需要用户身份验证，并且还强制执行某种授权。虽然这些机制的强制执行必须在远程终端上，但应用程序还必须确保它遵循所有相关的最佳实践，以确保所涉及协议的安全使用。
	MASVS-AUTH-2	根据平台最佳实践执行本地安全身份验证。	许多应用程序允许用户通过生物识别或本地PIN码进行身份验证。这些身份验证机制需要正确实施。此外，一些应用程序可能没有远程终端，完全依赖于本地应用程序身份验证。
	MASVS-AUTH-3	通过额外的身份验证来保护敏感操作。	对于应用程序中的敏感操作，通常需要其他形式的额外的身份验证。这可以通过安全地实现不同的验证方式（生物识别、PIN、MFA代码生成器、电子邮件、深度链接等）完成，
MASVS-NETWORK: 网络通讯	MASVS-NETWORK-1	根据当前最佳实践保护所有网络流量。	对于任何通过网络进行通信的应用程序来说，确保数据隐私和传输中任何数据的完整性至关重要。这通常是通过加密数据和验证远程终端来实现的，就像 TLS 一样。然而，开发人员有很多方法可以禁用平台的安全默认设置，或者通过使用低级 API 或第三方库完全绕过它们。这种控制措施可确保应用程序在任何情况下都能建立安全连接。
	MASVS-NETWORK-2	在开发人员控制下为所有远程终端执行身份固定。	这种控制措施不是信任框架或设备的所有默认根CA，而是确保只信任非常特定的CA。这种做法通常被称为证书固定或公钥固定。

列表2

MASVS-PLATFORM: 联动平台	MASVS-PLATFORM-1	安全使用IPC机制。	通常，应用程序使用平台提供的 IPC 机制来有意地暴露数据或功能。已安装的应用程序和用户都可以以多种不同的方式与应用程序进行交互。这种控制措施可确保了所有涉及 IPC 机制的交互都是安全的。
	MASVS-PLATFORM-2	安全使用WebView。	WebViews通常用于需要对UI进行更多控制的应用程序。这种控制措施可确保WebViews被安全配置，以防止敏感数据泄露以及敏感功能暴露（例如通过JavaScript桥到本机代码）。
	MASVS-PLATFORM-3	安全地使用用户界面。	在许多情况下，敏感数据必须在UI中显示（例如密码、信用卡详细信息、通知中的OTP代码）。此控制措施可确保这些数据不会因平台机制（如自动生成的屏幕截图）或意外事件（如肩窥或与他人共享设备）而无意中泄露。
MASVS-CODE: 代码质量	MASVS-CODE-1	最新的平台版本。	移动操作系统的每个版本都包含安全补丁和新的安全功能。使用旧版本，应用程序容易受到已知威胁的攻击。这种控制措施可确保应用程序在最新的平台版本上运行，以便用户拥有最新的安全保护。
	MASVS-CODE-2	具有强制应用程序更新的机制。	有时，在应用程序已经投入生产时，会发现关键漏洞。这种控制措施可确保有一种机制可以强制用户在继续使用应用程序之前更新应用程序。
	MASVS-CODE-3	仅使用没有已知漏洞的软件组件。	为了真正安全，应对所有应用程序组件进行全面的白盒评估。然而，就像第三方组件通常会遇到的情况一样，这并不总是可行的，而且通常也不是渗透测试的一部分。这种控制措施涵盖了“低级”情况，例如那些仅通过扫描库中的已知漏洞即可检测到的情况。
	MASVS-CODE-4	验证并清理所有不可信的输入。	应用程序有许多数据输入点，包括UI、IPC、网络、文件系统等。这些传入数据可能已被不可信的行为者无意中篡改，并可能导致绕过关键的安全检查以及经典的注入攻击，如SQL注入、XSS或不安全的反序列化。这种控制措施可确保这些数据被视为不可信的输入，并在使用前得到适当的验证和清理。
MASV-RESILIENCE: 逆向工程对抗和防篡改的能力	MASVS-RESILIENCE-1	验证平台的完整性。	在已被篡改的平台上运行应用程序可能非常危险，因为这可能会禁用某些安全功能，使应用程序的数据面临风险。对于许多依赖平台安全的MASVS控件（例如安全存储、生物识别、沙盒等）来说，信任平台是至关重要的。此控制措施可验证操作系统是否未被破坏，且其安全功能可以信任。
	MASVS-RESILIENCE-2	实现防篡改机制。	应用程序在用户控制的设备上运行，如果没有适当的保护措施，在本地运行篡改后的版本相对容易（例如在游戏中作弊，或在不支付费用的情况下启用高级功能），或将后门版本上传到第三方应用商店。这种控制措施可通过防止对原始代码和资源的篡改来确保应用程序预期功能的完整性。
	MASVS-RESILIENCE-3	实现反静态分析机制。	了解应用程序的内部通常是篡改它的第一步（无论是动态还是静态）。这种控制措施可通过使静态分析尽可能难以弄清楚应用程序的工作原理来阻碍理解。
	MASVS-RESILIENCE-4	实现反动态分析技术。	有时，纯粹的静态分析非常困难且耗时，因此它通常与动态分析相结合。在运行时观察和操纵应用程序使其更容易破译其行为。这种控制措施旨在使执行动态分析尽可能困难，并防止动态插装这些可能允许攻击者在运行时修改代码的行为。


OWASP