Mobile Security Risks

Introduction

Security testing will be conducted according to the OWASP Mobile Top 10 methodology. OWASP is the de facto security standard for web applications. It is constantly being developed and updated, thus allowing for timely response to new types of vulnerabilities. Today it covers 95% of critical vulnerabilities and their detection methods, which is described in details on the official site of the project https://www.owasp.org.

Mobile Security Risks

1.	Insecure Data Storage	This category insecure data storage and unintended data leakage. Data stored insecurely includes, but is not limited to, the following: - SQL databases; - Log files; - XML data stores ou manifest files; - Binary data stores; - Cookie stores; - SD card; - Cloud synced.
2.	Insecure Communication	This covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.
3.	Insecure Authentication	This category captures notions of authenticating the end user or bad session management. This can include: - Failing to identify the user at all when that should be required - Failure to maintain the user's identity when it is required - Weaknesses in session management
4.	Insufficient Cryptography	Insecure use of cryptography is common in most mobile apps that leverage encryption. There are two fundamental ways that broken cryptography is manifested within mobile apps. First, the mobile app may use a process behind the encryption / decryption that is fundamentally flawed and can be exploited by the adversary to decrypt sensitive data. Second, the mobile app may implement or leverage an encryption / decryption algorithm that is weak in nature and can be directly decrypted

		by the adversary.
5.	Insecure Authorization	This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.). If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.
6.	Client Side Injection	The best way to find out if an application is vulnerable to injection is to identify the sources of input and validate that user/application supplied data is being subject to input validation, disallowing code injection. Manual penetration testers can confirm these issues by crafting exploits that confirm the vulnerability.
7.	Cross Site Request Forgery	Cross-Site Request Forgery (CSRF) is a type of attack that occurs when a malicious web site, email, blog, instant message, or program causes a user's app to perform an unwanted action when the user is authenticated.
8.	Unintended Data Leakage	Unintended data leakage (formerly side-channel data leakage) includes vulnerabilities from the OS, frameworks, compiler environment, new hardware, etc. without a developers knowledge.