M8: Security Misconfiguration

Threat Agents

Application Specific

Security misconfiguration in mobile apps refers to the improper configuration of security settings, permissions, and controls that can lead to vulnerabilities and unauthorized access. Threat agents who can exploit security misconfigurations are attackers aiming to gain unauthorized access to sensitive data or perform malicious actions. Threat agents can be an attacker with physical access to the device, a malicious app on the device that exploits security misconfiguration to execute unauthorized actions on the target vulnerable application context.

Attack Vectors

Exploitability DIFFICULT

Security misconfigurations in mobile apps can be exploited through various attack vectors, including:

- Insecure default settings: Mobile apps often come with default configurations that may have weak security settings or unnecessary

permissions enabled, making them vulnerable to attacks.
- Improper access controls: Misconfigured access controls can allow unauthorized users to access sensitive data or perform privileged actions.
- Weak encryption or hashing: Improperly implemented or weak encryption and hashing algorithms can be exploited to gain access to sensitive information.
- Lack of secure communication: Failure to use secure communication protocols, such as SSL/TLS, can expose sensitive data to eavesdropping and man-in-the-middle attacks.
- Unprotected storage: Storing sensitive data, such as passwords or API keys, in an insecure manner, such as plain text or weakly encrypted, can lead to unauthorized access.
- Insecure file permissions: Storing application files with world-readable and/or world-writable permissions.
- Misconfigured session management: Improper session management can result in session hijacking, allowing attackers to impersonate legitimate users.

Security Weakness

Prevalence COMMON

Detectability EASY

Security misconfigurations are common in mobile apps due to factors such as time constraints, lack of awareness, or human error during development. Detecting security misconfigurations is relatively easy through manual code review, security testing, or automated scanning tools.

Examples of security misconfigurations include:

- Failure to disable debugging features in release builds, which can expose sensitive information.
- Allowing insecure communication protocols, such as HTTP, instead of enforcing secure communication over HTTPS.
- Leaving default usernames and passwords unchanged, providing easy access to attackers.
- Inadequate access controls that allow unauthorized users to perform privileged actions.

Technical Impacts

Impact SEVERE

Security misconfigurations can have significant technical impacts on mobile apps, including:

- Unauthorized access to sensitive data: Misconfigurations may allow attackers to access sensitive information, such as user credentials, personal data, or confidential business data.
- Account hijacking or impersonation: Weak or misconfigured authentication mechanisms can lead to account takeover or impersonation of legitimate users.
- Data breaches: Inadequate security configurations may result in data breaches, exposing sensitive data to unauthorized individuals.
- Compromise of backend systems: Misconfigurations in the mobile app can provide attackers with a foothold to compromise the backend systems or infrastructure.

Business Impacts

Impact SEVERE

Security misconfigurations can have severe business impacts, including:

- Financial loss: Breaches resulting from security misconfigurations can lead to financial losses, including legal penalties, regulatory fines, and damage to the organization's reputation.
- Data loss or theft: Misconfigurations can result in the loss or theft of sensitive data, leading to legal and financial consequences.
- Downtime and disruption: Exploitation of security misconfigurations can lead to app downtime, service disruption, or compromised functionality, affecting user experience and business operations.
- Damage to brand reputation: Publicly disclosed security incidents can damage the organization's reputation, leading to loss of customer trust and potential loss of business.

Am I Vulnerable to Security Misconfigurations?

Mobile apps are vulnerable to security misconfigurations if they have not been properly configured to follow security best practices. Common indicators of vulnerability to security misconfigurations include:

- Default settings not reviewed: Using default configurations without reviewing security settings, permissions and default credentials.

- Lack of secure communication: Using unencrypted or weakly encrypted communication channels.
- Weak or absent access controls: Allowing unauthorized access to sensitive functionality or data.
- Failure to update or patch: Not applying necessary security updates or patches to the app or underlying components.
- Improper storage of sensitive data: Storing sensitive data in plain text or weakly protected formats.
- Insecure file provider path settings: a file content provider that was meant for internal application use is exposed to other apps or users, which could potentially compromise sensitive data or allow unauthorized access to application resources.
- Exported activities: an activity that is meant for internal application use is exported and/or browsable, which exposes an additional attack surface.

To determine if your app is vulnerable to security misconfigurations, you should conduct a thorough security assessment, including code review, security testing, and configuration analysis.

How Do I Prevent Security Misconfigurations?

Preventing security misconfigurations in mobile apps requires following secure coding and configuration practices. Here are some key prevention measures:

- Secure default configurations: Ensure that default settings and configurations are properly secured and do not expose sensitive information or provide unnecessary permissions.
- Default credentials: Refrain from using hardcoded default credentials.
- Insecure permissions: Avoid storing application files with overly permissive permissions like world-readable and/or world-writable.
- Least privilege principle: Request only the permissions necessary for the proper functioning of the application
- Secure network configuration: Disallow cleartext traffic and use certificate pinning when possible.
- Disable Debugging: Disable debugging features in the production version of the app.
- Disable backup mode (Android): By disabling backup mode on Android devices, you prevent the inclusion of app data in the device's backup, ensuring that sensitive data from the app is not stored in the device backup.
- Limit application attack surface by only exporting activities, content providers and services that are necessary to be exported

Example Attack Scenarios

The following scenarios showcase security misconfigurations in mobile apps:

Scenario #1: Insecure default settings.

A mobile app is released with default settings that have weak security configurations enabled. This includes using insecure communication protocols, leaving default usernames and passwords

unchanged, and not disabling debugging features in release builds. Attackers exploit these misconfigurations to gain unauthorized access to sensitive data or perform malicious actions.

Scenario #2: Insecure file provider path settings.

A mobile app exposes its root path in an exported file content provider, allowing other apps to access its resources.

Scenario #3: Overly permissive storage permissions.

A mobile app that stores application shared preferences with world-readable permissions, allowing other apps to read them

Scenario #4: Exported activity.

A mobile app exports some activity that is meant for internal use, giving attackers extra attack surface to the application.

Scenario #5: Unnecessary permissions.

A mobile app requests excessive permissions that are not essential for its core functionality. For instance, a simple flashlight app requesting access to the user's contacts, location, and camera. This exposes user data to unnecessary risks, as the app could potentially misuse the granted permissions or unintentionally leak sensitive information.

References

- OWASP
  - OWASP API Security Top 10
  - OWASP Top 10
- External
  - External References

Edit on GitHub

Spotlight: Appdome

The Appdome mission is to protect every mobile app in the world at work and play, with billions of users protected today. Appdome delivers the only full lifecycle Unified Mobile App Defense platform with 300+ defenses spanning security, anti-malware, anti-fraud, anti-social engineering, mobile anti-bot, anti-cheat, geo compliance, MiTM attack prevention, code obfuscation, social engineering, and other protections. Automatically build security and privacy in your mobile CI/CD pipelines with Zero Code and No SDKs. Get OWASP MASVS Compliant in less than 5 minutes!

Corporate Supporters

Become a corporate supporter