NeGD

App Hosting & Security Guidelines

Gov.in App Store

2-10-2025

Contents

App Hosting & Security Guideline

Purpose:

These guidelines are intended to establish **foundational security standards** for app owners, enabling them to safeguard **user data, privacy, and integrity** in line with global best practices. The primary objective is to **facilitate secure development, distribution, and management** of applications (including payment, government service, and citizen-centric apps) on the Gov.in App Store. All facets of mobile app architecture, design, development, and deployment, as well as the user environments used for transactions or data interchange, must align with these guidelines.

References and Alignments:

- **OWASP Mobile Application Security Verification Standard (MASVS)**
- **OWASP Mobile Security Testing Guide (MSTG)**
- **App Defense Alliance** (collaboration between Google, ESET, Lookout, Zimperium, etc.)
- **ISO 27001/27701** (where applicable for information security management)
- **NIST Special Publications** (SP 800-53, SP 800-171) for additional security controls

Scope:

The GOV.IN AppStore (www.apps.mgov.gov.in) is India's first locally developed app store, under the Ministry of Electronics and Information Technology (MeitY). This platform hosts citizen-centric mobile applications and is part of the Atmanirbhar Bharat Mission, which aims to strengthen the mobile app industry in India by providing secure applications to users.

The GOV.IN AppStore enhances transparency by sharing key security issues and the processes used to identify and address them with each developer. The in-house testing procedures comply with OWASP standards, which builds trust between app developers and the government. The testing relies on various parameters, including SSL Pinning, verification of app signatures during runtime, code obfuscation, and the secure export of Android application components, among others.

Following the testing, developers receive a comprehensive report detailing all identified issues, categorized by priority: High, Medium, and Low. To successfully host their apps, developers must address any high-priority issues before resubmitting. The GOV.IN team offers support through personal or group meetings to assist developers in resolving these issues.

High-Level Security Framework

1. **OWASP MASVS Alignment**
    - Each uploaded app is evaluated against **OWASP MASVS** levels (L1, L2, or R—depending on the criticality of the app). This includes checks for secure data storage, network communications, and code integrity.
2. **App Defense Alliance Principles**
    - Encouraging robust code analysis, malware scanning, and **runtime integrity checks** to spot malicious or fraudulent behavior before apps go live.
    - Continuous monitoring for malicious updates and exploit attempts through post-publication scanning.
3. **Risk Categorization**
    - Identify high, medium, and low-risk vulnerabilities to help developers **prioritize remediation** efforts.
    - Offer structured guidance (e.g., recommended encryption libraries,

secure coding patterns) to address discovered vulnerabilities.

Developer Account Creation & Verification

1.1 Create and Set Up Your Developer Account

- Developers must:
    1. Provide **accurate legal details** (organization, contact email, phone number).
    2. Undergo **identity verification** (e.g., domain-based email checks for official or enterprise entities).
- **App Testing Requirements for New Personal Developer Accounts**
    - New developers are initially restricted in how many apps they can publish or may be required to submit test builds for security validation prior to wide release.

1.2 Device Verification Requirements for New Developer Accounts

- Encourages strong security measures—**2FA** or device-based verification—to reduce fraudulent account creation, aligning with Gov.in's emphasis on **secure developer onboarding**.

2. Preparing Your Application

2.1 Set Up Your App on the Gov.in Dashboard

- **Steps**:
    1. Enter essential metadata (title, short/long description, category).
    2. Provide a valid **Privacy Policy URL** if the app collects sensitive data.
    3. Link to your developer or product support channels.

2.2 Inspect App Versions with the App Bundle Explorer (or Equivalent)

- **Open Source App Bundle Explorer** helps developers see the generated artifacts (APKs) for different devices.
- **Gov.in Approach**:

- Offers a simplified tool or interface to inspect **APK / AAB** versions.
- Encourages developers to confirm that all split configurations are correct for different device types, OS versions, etc.

2.3 Declare Permissions & Prepare Your App for Review

- All permissions must be declared, with justification if it's sensitive (location, camera, contacts).
- **Gov.in Policy**: **Requires explicit permission declarations**:
  1. Explain why each permission is needed.
  2. For banking or financial apps, ensure alignment with **RBI** or other local guidelines on user data.
  3. Provide a "permission rationale" to end users, consistent with best practices.

---

3. Application Build & Testing Process

3.1 Prepare and Roll Out a Release

- Involves internal testing tracks, closed testing, open testing, and then production release.
- **Steps**
  - Encourages a **multi-stage release process**:
    1. **Internal Test**: Developer's in-house testing.
    2. **Security/Compliance Test** by Gov.in's team (akin to a "pre-review check").
    3. **Beta Release**: Optionally, a small user group can test in real-world conditions.
    4. **Production Publication**: Final, publicly available release.

3.2 Detect App Issues Early with Pre-Review Checks

- Integrates **OWASP MSTG** checks plus signature verification, code obfuscation checks, and basic privacy compliance validation.
- Any flagged issues must be resolved before the final listing.

3.3 App Testing Requirements for New Personal Developer Accounts

- Additional emphasis on verifying identity, ensuring first-time developers follow:
  1. Minimal viable security checks.
  2. Comply with store listing guidelines to reduce spam or policy violations.

4. Releasing & Publishing Your App

4.1 Publish Your App

1. **Final Security Clearance**: The Gov.in security team gives a "go/no-go" based on high-priority vulnerability resolution.
2. **Listing Confirmation**: Developers verify store details (screenshots, feature graphic, T&C).
3. **Live on Gov.in**: Users can now discover and download.

4.2 Helpful Tips to Get Your App Published

- **Google** suggests thorough documentation, minimal policy infractions, accurate store listings.
- **Gov.in**:
  - Make sure you have:
    1. **Clear descriptions** of functionality.
    2. **Comprehensive privacy policy** (if handling personal data).
    3. **No restricted content** (hate speech, adult content, etc.) as per Gov.in guidelines.

4.3 Best Practices for White Label Developers

- Each white-labeled version must have:
    1. Unique package name.
    2. Compliant branding and content.
    3. Separate listing if intended for a different user base or sector.

---

5. Updating & Managing Your Published App

5.1 Update or Unpublish Your App

1. **Patch Updates**: Must pass a shorter **re-test** cycle if changes affect critical permissions or app structure.
2. **Unpublish**: Developer can remove the app. Users with existing installs might continue usage, but no new downloads occur.

5.2 Control When App Changes Are Reviewed and Published

- Devs can choose to **delay** publication after receiving a successful review—helpful if they need a synchronized market launch or want final QA checks.

5.3 Release App Updates with Staged Rollouts

- Encourages a **phased approach** for major feature changes or new OS versions.
- Minimizes risk by letting a small subset test the update before full rollout.

5.4 Prompt Users to Update to Your Latest App Version

- Recommends using **native Android "in-app updates"** or custom prompts to keep users on the latest, most secure build.
- May enforce mandatory updates for critical security fixes (particularly for e-governance or financial apps).

---

6. Store Listing & Distribution Strategies

6.1 Manage Your Store Listings

1. **Gov.in** enforces consistent branding and accurate representation of features.
2. Local language support recommended for broader inclusivity.
3. Zero tolerance for misleading claims or keyword stuffing.

6.2 Manage Different Form Factor Releases (Dedicated Tracks)

- Encourages separate channels (e.g., "Tablet track," "Feature phone track" if applicable) to ensure optimum user experience across device categories.

6.3 Add or Test APK Expansion Files

- Large apps (GIS apps) may also store data externally with user consent.
- Must ensure **secure and encrypted** external storage to protect user data.

7. Using Third-Party SDKs & Target API Levels

7.1 Using Third-Party SDKs in Your App

1. Disclose all **analytics or ad networks** integrated.
2. **No malicious** or unverified SDKs that exfiltrate data.
3. If an SDK violates Gov.in policies (e.g., collecting personal data without consent), the developer must replace or remove it promptly.

7.2 Target API Level Requirements for Apps

- Parallel requirement: apps must periodically update their **targetSdkVersion** to the latest recommended level (e.g., target API level 33/34), ensuring modern security features.

8. Permissions & Policy Declarations

8.1 Declare Permissions for Your App

- We require declaration of all critical or sensitive permissions (SMS, call logs) require explicit justification and user-facing disclaimers.

8.2 Request More Time to Complete a Policy Declaration

1. If major policy changes occur (e.g., national data security guidelines are updated), developers can request a **grace period** for compliance.
2. Must show **legitimate progress** toward implementing the changes.

---

9. Additional Considerations

9.1 App Testing Requirements for Personal vs. Enterprise Developers

- We apply stricter scrutiny or a **pilot testing phase** for personal dev accounts to ensure they understand security protocols.

9.2 App Quality & User Engagement

- Encourages an **optimal user experience**, especially for citizen services (well-organized forms, minimal steps).
- Low crash rates and stable performance are essential for e-governance trust.

---

10. Enforcement & Policy Compliance

1. Issues a **violation notice** for policy breaches with a compliance window.
2. Failure to rectify results in **app removal** or even developer account suspension.
3. A structured **appeals process** is there wherein Developers can write to Gov.in team to appeal.

Procedure for app upload and developer registration:

1. **Developer Registration**: A developer registers on the Gov.In developer portal.
2. **Admin Approval**: The admin reviews and approves the developer account.
3. **Application Upload**: The developer uploads their Android application in either APK or AAB format.
4. **Application Submission**: The application is submitted to the Gov.In appstore.
5. **Support Contact**: If the developer encounters issues during upload, they can reach out to the Gov.In appstore support team via the support email.
6. **Error Resolution**: The support team provides solutions for any specific errors faced by the developer.
7. **Security Testing Process**: Once the application is uploaded, the admin team forwards it to the security testing team.
8. **Testing Guidelines**: The security team tests the application within 48 hours and updates the developer via support email or contact details.
9. **Security Test Report**: The security team prepares a report detailing any vulnerabilities found.
10. **Developer Acknowledgment**: The developer acknowledges receipt of the security test report.
11. **Vulnerability Resolution**: The Gov.In appstore team assists the developer in resolving any identified vulnerabilities.
12. **Resubmission**: The developer resubmits the application after addressing the vulnerabilities.
13. **Repeat Process**: Steps 7 to 12 are repeated until all severe vulnerabilities are resolved.
14. **Application Publication**: Once all severe vulnerabilities are addressed, the application is made live on the Gov.In AppStore.
15. **Financial and Banking Apps:** Only developer ID created using banking domain is allowed or an **annexure** is required for authorization.

Category of Apps banned on Gov.in AppStore

To maintain ethical standards and safeguard user interests, here are some categories of apps that could ideally be banned or heavily regulated on a Gov.in Appstore:

1. **Fraudulent Financial Apps:**
   - Apps offering get-rich-quick schemes, Ponzi schemes, or unverified crypto trading platforms.
   - Payday loan apps with predatory interest rates.
2. **Explicit or Inappropriate Content Apps:**
   - Apps promoting adult content or containing sexually explicit material.
   - Apps with violent or harmful content, such as torture games or glorification of criminal activity.
3. **Misleading Health and Fitness Apps:**
   - Apps offering unverified health advice, fake medication, or quack cures.
   - Apps claiming to provide medical diagnoses without proper licensing or approval.
4. **Malware or Data Harvesting Apps:**
   - Apps that pose cybersecurity risks or covertly collect and misuse user data.
   - Apps without proper encryption for sensitive data.
5. **Hate Speech and Extremism Apps:**
   - Apps promoting hate speech, radicalization, or extremist ideologies.
   - Platforms fostering cyberbullying, trolling, or online harassment.
6. **Illegal Streaming or Piracy Apps:**
   - Apps providing access to pirated content, including movies, TV shows, or software.
7. **Unregulated Gambling or Gaming Apps:**
   - Apps promoting unregulated online gaming, including games

with hidden gambling mechanisms like loot boxes.

8. **Fake or Misleading Educational Apps:**

- Apps spreading disinformation under the guise of education.
- Fraudulent certification or degree-granting platforms.

9. **Apps Violating Public Morality or National Integrity:**

- Apps promoting anti-national activities, defaming the government, or disrupting public order.
- Apps endorsing or facilitating illegal activities such as drug sales or arms trading.

10. **Dating Apps:**

- Apps that facilitate exploitative or harmful relationships, such as those promoting child exploitation, human trafficking, or abusive behaviors.

11. **Betting and Gambling Applications:**

- Apps promoting or facilitating gambling, betting, lottery schemes, or games of chance involving monetary or valuable considerations.

12. **Applications Promoting Criminal Activity:**

- Apps facilitating, encouraging, or inciting criminal acts such as fraud, theft, hacking, human trafficking, or organized crime.

13. **Self-Harm and Suicide-Related Applications:**

- Apps promoting, glorifying, or facilitating suicide, self-harm, or behaviors that endanger life or mental health.

14. **Applications Harmful to Public Morality and Order:**

- Apps containing obscene, vulgar, or explicit content that contravenes public decency and morality.

15. **Apps Promoting Hate Speech or Discrimination:**

- Apps promoting hate speech, communal disharmony, or discrimination based on religion, caste, gender, or ethnicity.

16. **Applications Threatening National Security and Sovereignty:**

- Apps associated with anti-national activities, espionage, or unauthorized collection of sensitive data.

Security Testing Toolkit and Checklist:

S.No.	Vulnerability	Description
1	Improper Export of Android Application Components	The application exports a component without adequately restricting which applications can access it or the data it contains.
2	Hard-coded Sensitive Information in Application Code	Sensitive information is stored in code, making it accessible to attackers who could exploit it.
3	Insecure Logging of the Application	The app outputs sensitive information to logs, which can be accessed through commands like logcat, exposing potentially sensitive data.
4	Insecure Permissions	The application sets insecure permissions, posing a security threat by not properly isolating its operations from others.
5	Debuggable Flag Set to True	If an app is marked as debuggable, attackers can inject code to execute within the app's process.
6	Vulnerable to Reverse Engineering Attack	Lack of binary protections allows adversaries to easily analyze and modify the app, potentially exposing sensitive intellectual property.
7	Insecure Data Storage in File System	Sensitive data is inadequately protected in the filesystem, making it accessible to malicious users or malware.
8	Backup Flag Set to True	By default, the app's data can be backed up, including private files, which can be a security risk if not managed properly.
9	WebView with JavaScript Enabled	Using WebView to display web content with JavaScript enabled can expose the app to various attacks, such as cross-site scripting (XSS).

10	Vulnerable to Runtime Analysis and Manipulation	The app can be analyzed during runtime, allowing attackers to track method calls and manipulate app behavior.
11	Weak Cryptographic Algorithms	The use of weak or broken cryptographic algorithms can compromise the confidentiality and integrity of sensitive data, making it easy for attackers to exploit.
12	Application Works on Rooted Device	The app does not check if it is installed on a rooted device, which can lead to security vulnerabilities.
13	SSL Pinning Not Implemented Properly	SSL pinning is not implemented correctly, which can allow attackers to perform man-in-the- middle attacks.
14	Code Tampering Attack	Attackers can reverse engineer, modify, and re- sign the app, then distribute it to users, leading to data theft or device compromise.
15	User Authentication for Sensitive Data	Sensitive information is displayed without proper user authentication, risking exposure of critical data.
16	Cleartext Traffic Set to True	The app uses cleartext network traffic, which lacks confidentiality and protection against tampering, making it vulnerable to interception.
17	Manifest File Reveals Sensitive Information	The application's manifest file contains sensitive data like API keys, which can be exploited by attackers.

Android Dynamic Pen Testing Checklist:

S.No	Vulnerability Description
1	SQL Injection Attack: An SQL injection attack involves inserting an SQL query via client input. It can lead to reading sensitive data, modifying the database, executing administrative operations, recovering file contents, and issuing OS

	commands.
2	Sensitive Information Exposure: Sensitive data in requests and responses should be encrypted properly. This includes passwords, account details, and personal identity information.
3	Invalid SSL Certificate: The SSL certificate may be expired or not yet valid. Browsers may display warnings that confuse users, leading them to question the authenticity of the site.

Materials and Equipment:

Tool	**Description**
Android Debug Bridge (ADB)	A command-line tool utilized for security analysis of mobile applications, allowing for the installation, removal, and data extraction on Android devices.
MobSF	An open-source Mobile Security Framework that automates security assessments for Android and iOS apps, conducting both static and dynamic evaluations to detect and mitigate vulnerabilities.
Apktool	An open-source utility that enables users to decompile, modify, and recompile Android application packages (APKs), allowing developers and security experts to analyze and customize Android apps.
Frida	A dynamic instrumentation toolkit that allows developers, reverse engineers, and security analysts to inject code into running application processes, useful for tasks like bypassing SSL pinning and analyzing network traffic.
Jadx	A command-line tool for reverse engineering Android applications, designed to decompile and analyze application bytecode into a human- readable Java source code format, aiding in understanding app functionality and identifying vulnerabilities.
Burp Suite	A tool for conducting security tests on mobile applications, requiring configuration to proxy mobile device traffic through Burp Proxy. This allows interception and modification of all HTTP/S requests and responses for penetration testing.

Post-Publication Monitoring & Maintenance

1. **Auto-Scanning for Updates**
   - Each new app update triggers an incremental security scan. High severity vulnerabilities must be resolved prior to re-publication.
2. **Incident Response Mechanism**
   - If a live app is reported to have critical security gaps, the Gov.in AppStore team can **immediately delist or suspend** the app until patched.
3. **Continuous Security Training**
   - Developers are encouraged to join Gov.in's training sessions on secure coding, **OWASP** top threats, and the latest in **App Defense Alliance** research.
4. **Community Feedback Loop**
   - Collect user feedback about suspicious behavior or performance anomalies. Automate "flagging" of potential malicious updates.
