INSECURE DATA STORAGE

&

SENSITIVE DATA EXPOSURE

Common security flaws that compromise the privacy and security of sensitive data in systems and apps include insecure data storage and sensitive data exposure.

Insecure Data Storage

When data kept on a device, server, or database is not adequately protected, it may be subject to unwanted access. This is known as insecure data storage. It happens when private information, financial information, user passwords, or other sensitive data is kept in storage without the appropriate encryption, access controls, or tampering precautions.

Report by Komalseerut Kaur

Common causes of Insecure Data Storage:

- Lack of encryption for sensitive data.
- Using weak or reversible encryption algorithms.
- Failure to set access controls, allowing unauthorized users to access data.
- Insufficient protections for data stored in mobile apps, databases or local storage.

Sensitive Data Exposure

Sensitive data exposure describes circumstances in which inadequate or non-existent safeguards during data processing, transfer, or storage make sensitive information available to unauthorized persons. Weak encryption, inappropriate handling of sensitive data, or unencrypted data transmission over the network can all lead to this vulnerability.

Common causes:

- Transmitting sensitive data over unencrypted channels.
- Improperly configured SSL/TLS settings.
- Exposing sensitive data in logs or error messages.

- Insufficient protection of data in client-side or server-side storage.

Key Differences-

Insecure Data Storage- Vulnerabilities in the way data is stored on devices, databases, or files.
Sensitive Data Exposure- Any flaw that permits unauthorized access to data, whether it is in transit or storage.

Methods to find Insecure Data Storage:

Security analysts usually combine automated technologies and manual methods to identify vulnerabilities related to sensitive data exposure and insecure data storage. The following are some efficient techniques for identifying these weaknesses:

Finding Insecure Data Storage Vulnerabilities:

1. Static Analysis-
    - Use SAST Tools- To check the source code for unsafe data storage methods, such as storing sensitive data in plaintext, using inadequate encryption, or having poor cryptographic features, use Static Application Security Testing (SAST) tools like SonarQube, Fortify, or Veracode.
    - Verify the code for incorrect permission settings, insecure encryption techniques, and sensitive data that has been hardcoded.

2. Reverse Engineering- (for mobile apps)
    - You may examine the source code and decompile mobile programs (APK for Android, IPA for iOS) using tools like Jadx, ApkTool, and MobSF.
    - Keep an eye out for indications of unsafe data storage in the code, such as data kept in unprotected places like application-specific folders, caches, or local storage.

3. File and Database Inspection-
    - Examine the application's local databases (such as SQLite) and files to look for private information kept in plaintext.

- To find any sensitive data that isn't encrypted, tools such as ADB (Android Debug Bridge) for Android apps can be used to analyze files and databases on devices or emulators.

4. Manual Code Review-
    - To find unsafe storage methods, such as inadequate or nonexistent encryption, a lack of access controls, or direct access to private information, perform a manual code review.

5. Platform-specific Security Testing-
    - To verify whether apps are safely storing sensitive data utilizing the platform's secure protocols, use iOS Keychain or Android Keystore testing.

Finding Sensitive Data Exposure Vulnerability

1. Dynamic Analysis-
    - To examine data transfer, use Dynamic Application Security Testing (DAST) tools such as Acunetix, OWASP ZAP, or Burp Suite. They are able to determine whether private information is being sent across unsecure channels (such as HTTP rather than HTTPS).
    - Verify whether session or authentication tokens are stored insecurely in cookies or local storage, or whether they are exposed during transmission.

2. Intercepting Traffic-
    - To intercept network communication between the client (such as a web application or mobile app) and the server, use a proxy tool such as OWASP ZAP or Burp Suite. Look for:
      sensitive information sent in plaintext (e.g., over HTTP).
      inadequate SSL/TLS setups, which could point to a data exposure issue.
    - Keep an eye out for sensitive data being transmitted without encryption, such as login passwords or personal information.

3. Manual Penetration Testing-
    - To examine the exposure of sensitive data across the application levels, conduct manual penetration testing.

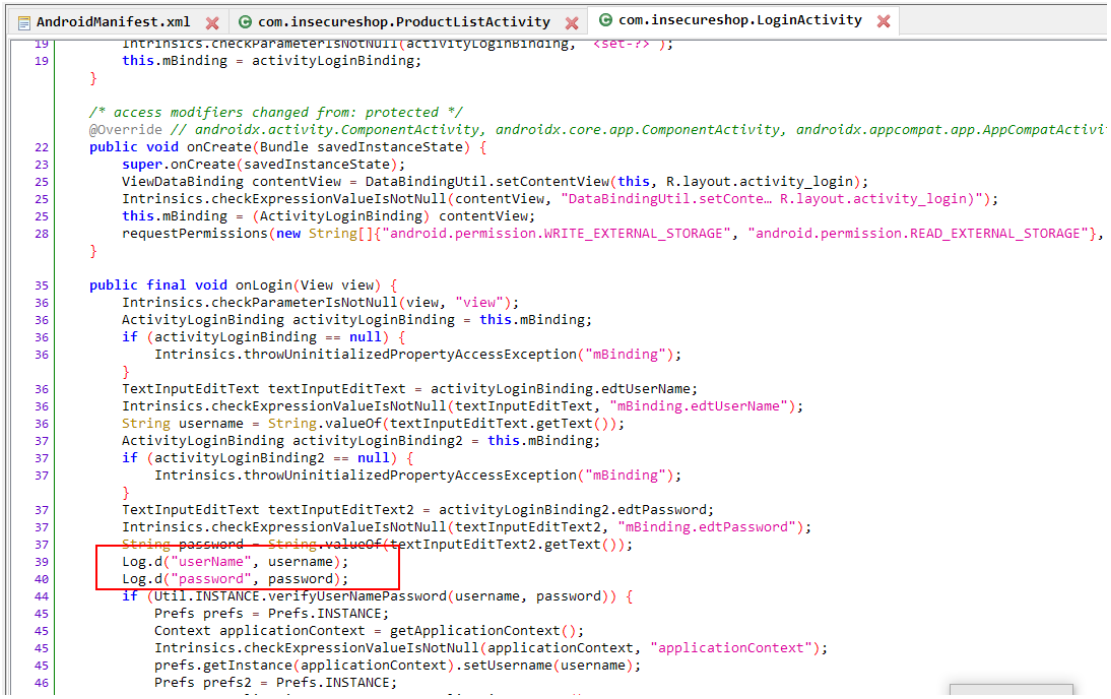
4. Configuration and SSL/TLS Testing-
    - To evaluate the security of SSL/TLS implementations and look for flaws like antiquated protocols or weak cypher suites that could expose sensitive data, use SSL Labs or TestSSL.sh.
    - Make that HTTP is correctly routed to HTTPS and that HTTPS is enforced.

5. Sensitive Data in Logs-
    - Check application logs for any exposed sensitive information. Check for passwords, private data, or tokens that are being logged in plaintext.
    - Ensure sensitive data is masked, encrypted, or omitted from logs.

6. Content Security Policy (CSP) Testing:
    - To avoid inadvertent data exposure through client-side scripts or third-party services, confirm that CSPs are implemented.
    - This lessens the possibility of sensitive data (such as forms or session tokens) being exposed to injected scripts.

Lab Solved-

Insecure Logging:



Insecure Data Storage:

```
C:\Users\Pc\Downloads\InsecureShop-Writeup>adb shell
2026:/ $ run-as com.insecureshop
2026:/data/user/0/com.insecureshop $ ls -la
total 72
drwx------   7 u0_a478 u0_a478        4096 2022-04-06 10:38 .
drwxrwx--x 357 system  system        20480 2022-04-06 10:23 ..
drwxrwx--x   2 u0_a478 u0_a478        4096 2022-04-06 10:38 app_textures
drwx------   3 u0_a478 u0_a478        4096 2022-04-06 10:38 app_webview
drwxrws--x   4 u0_a478 u0_a478_cache  4096 2022-04-06 10:38 cache
drwxrws--x   2 u0_a478 u0_a478_cache  4096 2022-04-06 10:22 code_cache
drwxrwx--x   2 u0_a478 u0_a478        4096 2022-04-06 10:38 shared_prefs
2026:/data/user/0/com.insecureshop $ cd shared_prefs/
2026:/data/user/0/com.insecureshop/shared_prefs $ ls
Prefs.xml  WebViewChromiumPrefs.xml
2026:/data/user/0/com.insecureshop/shared_prefs $ cat Prefs.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">!ns3csh0p</string>
    <string name="productList">[{&amp;quot;id&amp;quot;:1,&amp;quot;imageUrl&amp;quot;:&amp;quot;https
    <string name="username">shopuser</string>
</map>
2026:/data/user/0/com.insecureshop/shared_prefs $
```

Report by Komalseerut Kaur

REFERNCES:

1. https://owasp.org/www-project-mobile-top-10/2023-risks/m9-insecure-data-storage
2. https://medium.com/mobile-penetration-testing/03-insecure-data-storage-part-1-593177b56a1d
3. https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure
4. https://securiti.ai/blog/sensitive-data-exposure/
5. https://medium.com/@jeetpal2007/how-i-discovered-sensitive-data-exposure-in-android-app-e9d02ae0381d
6. https://medium.com/@Clownx_99/insecure-data-storage-9ca8410319ea