SECUREU
WE MAKE SECURITY ACCESSIBLE

ANDROID APP PENETRATION TESTING

SECURITY REPORT

DEMO CORP

DATE : DD MONTH YYYY

VERSION : 1.0

Confidential Document

Attention : This document contains confidential and privileged information for the intended recipient only. Any unauthorized disclosure, copying or distribution is prohibited. By accepting this document, you agree to maintain its confidentiality.

CONTENTS ...

1. Confidentiality Statement

This document is the exclusive property of Demo Corp and SECUREU. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires the consent of both Demo Corp and SECUREU.

Demo Corp may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

2. Disclaimer

This penetration test was performed by scanning the android application owned by Demo Corp for identifying potential security weaknesses and vulnerabilities.

A penetration test is considered as a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

3. Contact Information

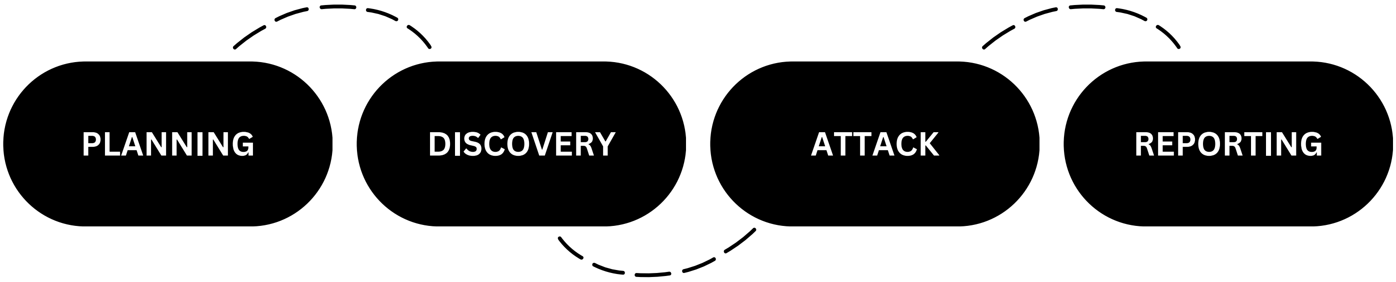
Name	Title	Contact Information
John	Lead Penetration Tester	john@secureu.in
Albert	Security Engineer	albert@secureu.in
Gilbert	Penetration Tester	gilbert@secureu.in

4. Assessment Overview

From <month> xth, 20XX to <month> yth, 20XX SECUREU attempted to evaluate the security posture of the Android application of Demo Corp and compared it to the current industry best practices by performing a penetration test

Phases of penetration testing activities include the following:

- **Planning** – Customer goals are gathered and rules of engagement obtained.
- **Discovery** – Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
- **Attack** – Confirm potential vulnerabilities through exploitation and perform additional discoveries upon new access.
- **Reporting** – Document all found vulnerabilities, exploits, failed attempts, and company strengths and weaknesses.



5. Assessment Components

5.1. Android App Penetration Test

Android app penetration testing, is the process of evaluating the security of an Android application by attempting to identify and exploit vulnerabilities that could be used by attackers to gain unauthorized access or perform malicious actions.

6. Finding Severity Ratings

The following table defines levels of severity and the corresponding CVSS score range used throughout the document to assess vulnerability and risk impact.

SEVERITY	CVSS V3 SCORE RANGE	DEFINITIONS
Critical	9.0-10.0	**Exploitation:** • Exploitation is straightforward. • Results in system-level compromise. **Plan of Action:** • Patch immediately.
High	7.0-8.9	**Exploitation**: • Exploitation is more difficult. Could cause elevated privileges. • Potential to cause loss of data or downtime. **Plan of Action**: • Patch as soon as possible.
Medium	4.0-6.9	**Exploitation:** • Vulnerabilities exist but are not exploitable. • Might require extra steps to make the vulnerability exploitable. **Plan of Action**: • Patch after high-priority issues has been resolved.
Low	0.1-3.9	**Exploitation:** • Vulnerabilities are non-exploitable. • Mitigation would reduce an organization’s attack surface. **Plan of Action:** • Patch during the next maintenance window.
Informational	N/A	• No vulnerability exists. • Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

7. Risk Factors

Risk is measured by two factors: Likelihood & Impact

7.1. Likelihood

Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the difficulty of the attack, the available tools, the attacker's skill level, and the client environment.

7.2. Impact

Impact measures the potential vulnerability’s effect on operations, including confidentiality, integrity and availability of client systems and/or data, reputational harm, and financial loss.

8. Scope

ASSESSMENT	DETAILS
Android app penetration testing	abc.apk

8.1. Scope Exclusions

SECUREU did not perform any of the following attacks during testing:

- Denial of Service (DoS)
- Phishing/Social Engineering

All other attacks not specified above were covered in penetration testing of the android application.

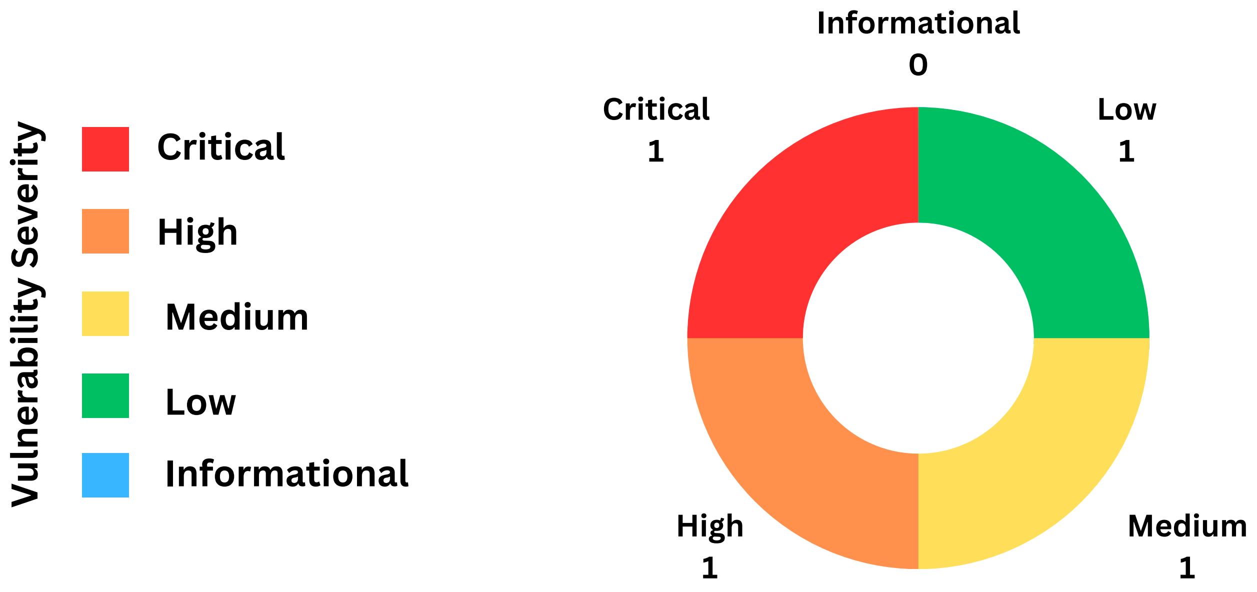
9. Executive Summary

SECUREU evaluated Demo Corp's security posture through an android app penetration test from <month> xth, 20XX to <month> Yth, 20XX. The following sections provide a high-level overview of vulnerabilities discovered, successful and unsuccessful attempts, and strengths and weaknesses.

B

Security Rating

VULNERABILITY DISTRIBUTION



9.1. Scoping and Limitations

Scoping during the engagement did not permit denial of service or social engineering across all testing components.

9.2. Testing Summary

The abc.apk Android application owned by Demo Corp has several security vulnerabilities that need to be addressed urgently to ensure the security of the application and its users. These vulnerabilities include **sensitive cache information disclosure**, **lack of SSL certificate pinning, no root check on Android applications, and debugging enabled.**

Sensitive cache information disclosure vulnerability can allow unauthorized access to sensitive data, such as **user credentials and API keys**, stored in the cache. Implementing encryption of sensitive data, proper cache management, and regular security audits can help remediate this vulnerability.

Lack of SSL certificate pinning vulnerability can allow **man-in-the-middle attacks**, enabling attackers to **intercept sensitive data transmitted over the network**. Implementing SSL certificate pinning is recommended to reduce the risk of exploitation.

No root check on Android applications vulnerability can enable **malicious users to escalate privileges and gain unauthorized access to sensitive data** or perform malicious actions on the device. Remediation includes checking for root access and implementing appropriate access controls.

Debugging-enabled vulnerability can allow attackers to **access sensitive information or modify the application's behavior,** leading to data theft, account takeover, or other security breaches. Setting debuggable to false in production applications can help remediate this vulnerability.

These vulnerabilities can cause significant security breaches and data loss, resulting in reputational damage and financial losses for the company. Immediate action is recommended to address these vulnerabilities and conduct regular security audits to identify and remediate new vulnerabilities that may arise.

10. Key Strengths and Weaknesses

The following identifies the **key strengths** found during the assessment:

- Use of technologies that have no exploits available out in the open.

The following identifies the **key weaknesses** found during the assessment:

- Good security practices are not being followed.

11. Vulnerability Distribution Table

The following table illustrates the vulnerabilities found by findings and recommended remediations:

FINDINGS	SEVERITY	RECOMMENDATIONS
MPT-001: Sensitive Cache Information Disclosure	Critical	Do not store sensitive information in a local cache file or encrypt the sensitive information before storing into the local cache.
MPT-002: No SSL Certificate Pinning	High	Implement SSL pinning.
MPT-003: No Root Check on Android Application	Medium	Implement root detection before beginning the runtime of the application.
MPT-004: Debugging Enabled	Low	Disable application debugging in the "AndroidManifest.xml" file. This can be done by setting the android:debuggable property to false .

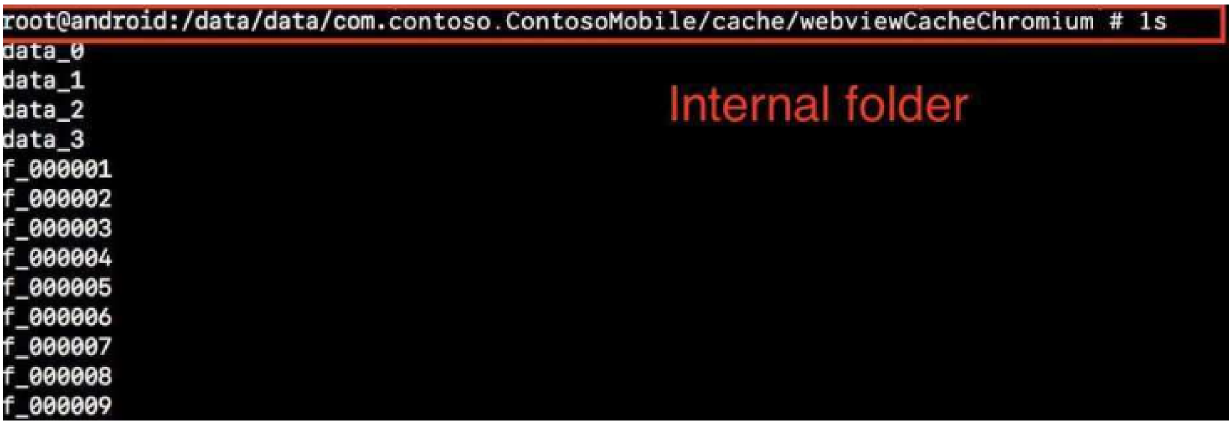
12. Android application Penetration Testing FIndings

MPT-001: Sensitive Cache Information Disclosure (Critical)

DESCRIPTION:	The mobile application may expose potentially sensitive information through local cache files. An attacker could get more information than necessary, and try to create a new attack vector over the application or the server back-end. Additionally, in this case, an attacker could get the application code and search for sensitive data inside it.
APPROACH:	Manual
TOOLS USED:	N/A
REFERENCES:	1. Insecure Data Storage (OWASP)

Evidence

Since the application was lacking root checks, internal files of the application were exposed revealing chromium's cache files.



```
root@android:/data/data/com.contoso.ContosoMobile/cache/webviewCacheChromium # ls
data_0
data_1
data_2
data_3
f_000001
f_000002
f_000003
f_000004
f_000005
f_000006
f_000007
f_000008
f_000009
```

Fig 1: Cache files revealed

Risk

- ### Likelihood

The likelihood of sensitive cache information disclosure vulnerability depends on the security measures implemented by the application, with poor encryption or management makes it **highly likely**.

- ### Impact

The impact of sensitive cache information disclosure vulnerability is that **sensitive data can be compromised**, leading to unauthorized access, account takeover, or data theft. The **impact can be severe** depending on the type of sensitive information disclosed.

Remediation

Do not store sensitive information in a local cache file or encrypt the sensitive information before storing into the local cache.

- ### More Information

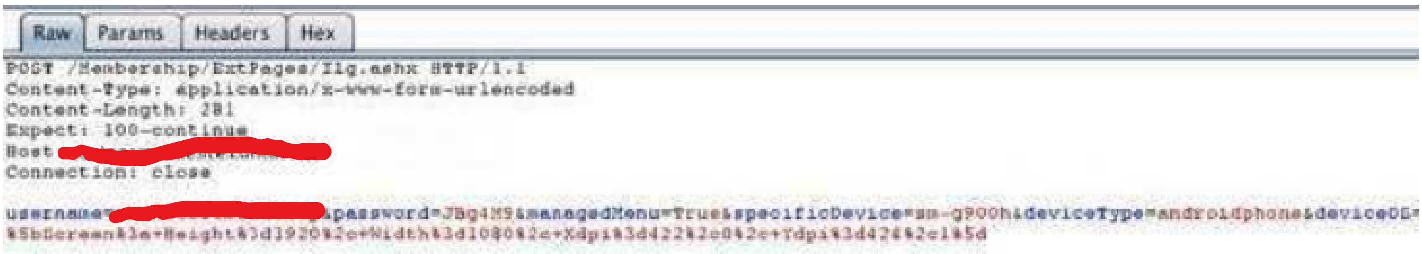
  - Data Storage on Android (OWASP MASTG)
  - Security tips (Android developer)

MPT-002: No SSL Certificate Pinning (High)

DESCRIPTION:	Certificate pinning is used to verify that client communications are being sent to the intended remote server, so if the server presents an unknown certificate, the application will not transmit the data. Without pinning, an attacker can implement their own intercepting proxy with an arbitrary certificate to view requests to and from their client device in cleartext.
APPTOACH:	Manual
TOOLS USED:	BurpSuite
REFERENCES:	1. SSL pinning walkthrough (IndusFace) 2. Certificate and public key pinning

Evidence

Below is an interception proxy being used by the assessor to intercept and modify requests. Upon examination of the request it was apparent that the application is sending data in cleartext format.



```
POST /Membership/ExtPages/Ilg.ashx HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 281
Expect: 100-continue
Host [redacted]
Connection: close

username=[redacted]&password=JBg4M9&managedMenu=True&specificDevice=sm-g900h&deviceType=androidphone&deviceOS=
%5bScreen%3a+Height%3d1920%2c+Width%3d1080%2c+Xdpi%3d422%2c0%2c+Ydpi%3d424%2c1%5d
```

Fig 2: Stored XSS

Risk

- ### Likelihood

  The likelihood of exploitation of lack of SSL certificate pinning varies but not implementing it **increases the risk of man-in-the-middle attacks**, making it **easier for attackers to intercept sensitive** data such as user credentials, **increasing the likelihood** of successful attacks.

- ### Impact

  The impact of not implementing SSL certificate pinning can be severe as it **increases the risk of man-in-the-middle attacks**, potentially leading to data theft, account takeover, and other security breaches. The impact depends on the sensitivity of the intercepted data.

Remediation

The remediation for lack of SSL certificate pinning is to implement SSL certificate pinning in the application, which ensures that the application only communicates with trusted servers, reducing the risk of man-in-the-middle attacks and enhancing the security of the application.

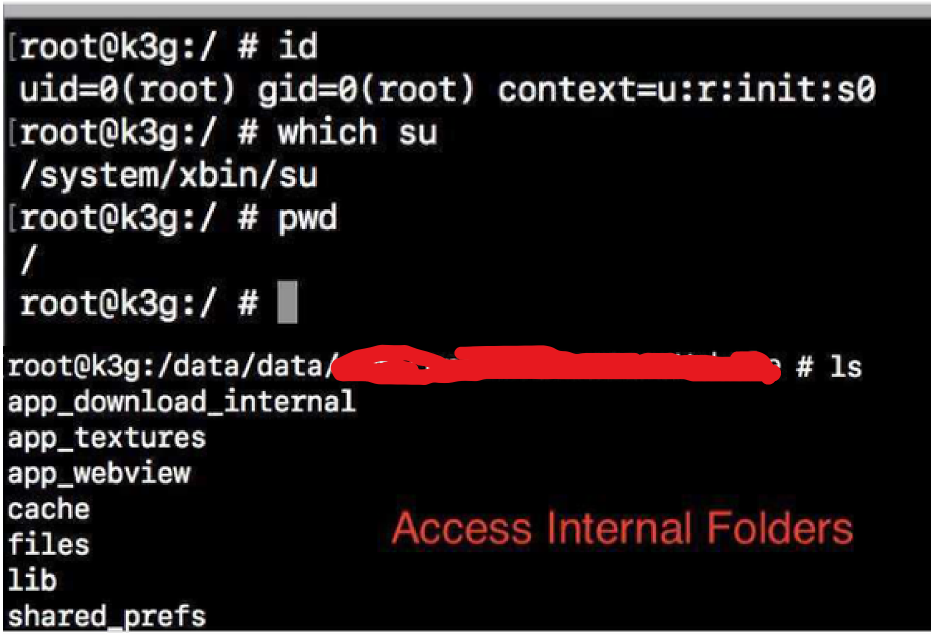
- **More Information**
  - Android developer documentation ssl pinning
  - Netguru 3 ways to implement ssl pinning

MPT-003: No Root Check on Android Application (Medium)

DESCRIPTION:	This Vulnerability in Android applications refers to the security flaw where an **application does not check whether the device is rooted or not**. **This can allow malicious users to bypass security** measures and gain escalated privileges on the device, potentially leading to unauthorized access, data theft, or other security breaches.
APPROACH:	Manual
TOOLS USED:	jadx-gui
REFERENCES:	1. OWASP 2. Fluidattacks

Evidence

Switching the user to root and browsing the directories of the application the same sensitive data was encountered.



```
[root@k3g:/ # id
 uid=0(root) gid=0(root) context=u:r:init:s0
[root@k3g:/ # which su
 /system/xbin/su
[root@k3g:/ # pwd
 /
 root@k3g:/ # 
root@k3g:/data/data/[redacted] # ls
app_download_internal
app_textures
app_webview
cache
files
lib
shared_prefs
```

Access Internal Folders

Fig 3: The application doesn't check for root thus granting access to internal files.

Risk

- **Likelihood**

  The likelihood of the No Root Check vulnerability depends on the attacker's skills and the application's implementation, with the absence of a root check, making it highly likely and thus enabling attackers to gain escalated privileges and perform unauthorized actions on the device.

- **Impact**

  The **No Root Check vulnerability** enables **attackers to gain escalated privileges and perform unauthorized actions** on the device, potentially leading to data theft, account takeover, and other security breaches. The impact depends on the type and sensitivity of the data accessed or actions performed.

Remediation

Implement image resolution validation before uploading the image or the back-end code should trim any image to acceptable and stable dimensions that can be rendered safely on the client side.

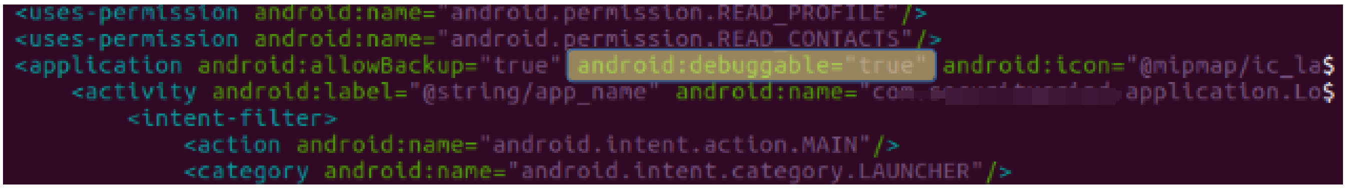
- **More Information**
  - How to Implement Root Detection in Android Applications (Indus Face)
  - Root detection that cannot be bypassed (Stack Overflow)

MPT-004: Debugging Enabled (Low)

DESCRIPTION:	**Debugging was found to be enabled** for this Android application which aids an attacker in dumping stack trace, accessing debugger helper classes, etc.
APPROACH:	Manual
TOOLS USED:	jadx-gui
REFERENCES:	1. Exploiting debuggable android applications (Infosec Institute) 2. Possible exploits over a debuggable APK (Stackexchange)

Evidence

Screenshot below proves that the debuggable parameter is set to true for this application.



```
<uses-permission android:name="android.permission.READ_PROFILE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<application android:allowBackup="true" android:debuggable="true" android:icon="@mipmap/ic_la$
    <activity android:label="@string/app_name" android:name="com.[redacted]application.Lo$
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
```

Fig 4: Debugging enabled

Risk

- **Likelihood**

  Setting the debuggable parameter to true makes it **highly likely** for the attackers to access sensitive information or perform malicious actions on the device.

- **Impact**

  The Debuggable set to true vulnerability **enables attackers to access and modify sensitive information, leading to data theft, account takeover, or other security breaches, with significant impact** depending on the data accessed or modified.

Remediation

Disable application debugging in the "AndroidManifest.xml" file. This can be done by setting the "android:debuggable" property to false

- **More Information**
  - Disable app debug (developer.android.com)

SECUREU

WE MAKE SECURITY ACCESSIBLE

Questions? Contact us at

Email : contact-us@secureu.in Phone : +91 8010450348