



Threatsys Technologies Pvt. Ltd.

ANDROID APP PENETRATION TESTING

A general guidance on decoding Android applications

Android Application Penetration Testing:

A general guidance on decoding Android applications

Mobile application penetration testing is commonly used for identifying weaknesses and vulnerabilities in mobile applications as part of cybersecurity procedures. Android penetration testing involves critical, high, medium, and low-level debugging, utilising tools like JADX-GUI, Drozer and Android Studio. Decoding the debugging procedures involves acknowledging the application structure and revealing the unknown secrets related to Application vulnerabilities.

Decoding Apps: Reversing Easy APK with JADX-GUI

JADX-GUI helps professionals to extract APKs in terms of v1, v2, and v3 according to their APK versions.

v1 (JAR Signature) defines an Android application in terms of originality and legacy signing schemes. This method is efficiently suitable for any Android version due to the utilisation of Java Archive signing procedures.

v2 (APK Signature Scheme v2) is defined as the upgrade of the original signing process established in Android 7.0. This enhancement scheme compiles with a separate APK signature block to increase the efficiency of the verification process and limit the signing-related hazards.

v3 (APK Signature Scheme v3) is also described as an improvement for the signing process for Android 9.0 that includes supportive key rotation similar to v2. This scheme additionally introduces strengthening security features for app signing.

When the application is only signed with v1, it's considered less secure for the users. It has a lot of absence of modern security features and options that are incorporated with the latest Android versions to enhance the potential to acknowledge the potential or certain types of Cyber-attacks.



```

APK signature verification result:

Signature verification succeeded
Valid APK signature v1 found

Signer CERT.RSA (META-INF/CERT.SF)

Type: X.509
Version: 1
Serial number: 0x1
Subject: CN=Android, CN=Android Debug
Valid from: Fri Sep 23 00:09:24 IST 2016
Valid until: Sun Sep 16 00:09:24 IST 2046

Public key type: RSA
Exponent: 65537
Module size (bits): 1024
Modulus: 1071003432676467126871343719933120109096176628405109223991346015248410231049416457140797505287415674602021401097549

Signature type: SHA1withRSA
Signature OID: 1.2.840.113549.1.1.5

MD5 Fingerprint: AF 5F F4 23 5E 64 1B AE 76 EB 04 D5 5C 2E B6 70
SHA-1 Fingerprint: AF BF FD BD 43 7C 68 39 57 23 C8 21 90 A8 EA 8A 29 04 C2 AF
SHA-256 Fingerprint: F1 D4 2F 10 D7 30 FF 29 08 CC 22 74 A6 00 B6 64 2D F0 37 9A 0E A2 62 31 64 E9 69 A9 00 6E 0E 20

Valid APK signature v2 found

Signer 1

Type: X.509
Version: 1

```

The professional needs to open the manifest file and recognise the API level, which should be above 21, and also needs to be the minimum SDK version. API level is recognised as a numerical identification tool that is utilised for recognising the Android operating system different versions. This level is recognised by the teachers and the abilities of each Android version, which is provided by the developer while creating the applications. The high level of API represents that developers can access more functionality and create more enhancements in the Android versions.



```

<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="owasp.sat.agooat">
    <uses-sdk android:minSdkVersion="18" android:targetSdkVersion="26"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debuggable="true" android:allowBackup="true">
        <activity android:name="owasp.sat.agooat.SplashActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

Analysing and identifying the unwanted permissions



```

<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="owasp.sat.agooat">
    <uses-sdk android:minSdkVersion="18" android:targetSdkVersion="26"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debuggable="true" android:allowBackup="true">
        <activity android:name="owasp.sat.agooat.SplashActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

Checking the components is essential to acknowledge the cybersecurity-related errors or any attacks. While utilising the Android export attribute, showcase the result through but it should be false, which is a clear indication of any real-life consequences.

Unauthorised access makes the export attribute through, and it provides access to the other third-party applications or entities to allow access to the sensitive information. This also leads

to creating data leakage and exposing users' sensitive data and personal information that should be restricted. Mini cyber-attacks create manipulation with the application by creating security breaches and lead to unauthorised control through remote code execution.

```
</activity>
<receiver android:name="owasp.sat.agooat.ShowDataReceiver" android:enabled="true" android:exported="true"/>
<activity android:label="@string/hardcode" android:name="owasp.sat.agooat.HardCodeActivity"/>
<activity android:label="@string/sql" android:name="owasp.sat.agooat.InsecureStorageSQLiteActivity"/>
<activity android:label="@string/sp2" android:name="owasp.sat.agooat.InsecureStorageSharedPrefsActivity"/>
<activity android:label="@string/network" android:name="owasp.sat.agooat.TrafficActivity"/>
<activity android:name="owasp.sat.agooat.ContentProviderActivity"/>
<activity android:label="@string/emulator" android:name="owasp.sat.agooat.EmulatorDetectionActivity"/>
<activity android:label="@string/sdCard" android:name="owasp.sat.agooat.InsecureStorageSDCardActivity"/>
<activity android:label="@string/webviewAccess" android:name="owasp.sat.agooat.InputValidationsWebViewURLActivity"/>
<activity android:label="@string/oscmd" android:name="owasp.sat.agooat.InputValidationsOSCMDInjectionMain2Activity"/>
<service android:name="owasp.sat.agooat.DownloadInvoiceService" android:enabled="true" android:exported="true" />
<activity android:label="@string/BinaryPatching" android:name="owasp.sat.agooat.BinaryPatchingActivity"/>
<activity android:label="@string/clipboard" android:name="owasp.sat.agooat.ClipboardActivity"/>
<activity android:label="@string/InsecureStorage" android:name="owasp.sat.agooat.InsecureStorageActivity"/>
<activity android:label="@string/SideChannelLeakage" android:name="owasp.sat.agooat.SideChannelDataLeakageActivity"/>
<activity android:label="@string/InputValidations" android:name="owasp.sat.agooat.InputValidationsActivity"/>
<activity android:label="@string/dict" android:name="owasp.sat.agooat.KeyboardCacheActivity"/>
<meta-data android:name="android.support.VERSION" android:value="28.1.0" />
<meta-data android:name="android.arch.lifecycle.VERSION" android:value="27.0.0-SNAPSHOT" />
</application>
```

Allow backup is a procedure that helps the application to allow the data backup in an AndroidManifest.xml file format. It is easier to utilise for users, but it increases the risk of cybersecurity-related attacks and creates potential vulnerabilities. The attackers mostly target these Store backups to expose the sensitive data and violate the data privacy compliance. For this reason, encrypted data storage backup is essential as a security storage practice to minimise the potential risk of third-party entities.

When setting debuggable to true in the AndroidManifest.xml automatically opens the debugging mode on the application. It is easier for development but creates a strong security risk potential in the aspect of the production environment. Debuggable apps can be more sensitive towards exposing the private information of the users by unauthorised access or reverse engineering. During the setting, debuggable needs to be false to reduce the potential of cybersecurity-related risk and exposure of sensitive data.

```
        android:debuggable="true" android:allowBackup="true" />
```

Setting the android: usesc cleartexttraffic to true in the AndroidManifest.xml file provides access to the unencrypted network traffic. This creates a major security risk or challenges for the users that expose sensitive information during transmission. Professionals are recommended to set the result to false to provide security by utilising secure communication

protocols like HTTPS to provide clear protection and encrypted data to prevent unauthorised access or third-party integration.

android:usesCleartextTraffic="true"

Manual inspection and automated analysis are essential for identifying obfuscated source code.

Variable and method names are also considered as methods, where the short names for methods and variables can create challenges for the developers or the testers to acknowledge the purpose of the code.

Professionals face difficulties in understanding the code's logic or purpose due to obfuscation tools that remove any documentation from the comments of the specific code.

Professionals commonly identify obfuscation patterns, which are mostly the single-letter names or variables or any other alphanumeric sequence.

The obfuscation tool also helps in identifying the unused code that can create major confusion during the analysis.



The screenshot shows the Android APK decompiled code. A red box highlights the 'BinaryPatchingActivity' class. The code implements the 'onCreate' method and contains logic for finding views by ID and checking if a user is an admin. It also includes a comment indicating JADK INFO about modifier changes.

```

    this._$findViewCache = new HashMap();
    View view = (View) this._$findViewCache.get(Integer.valueOf(i));
    if (view == null) {
        View findViewByIdById = findViewById(i);
        this._$findViewCache.put(Integer.valueOf(i), findViewByIdById);
        return findViewByIdById;
    }
    return view;
}

public final boolean isAdmin() {
    return this.isAdmin;
}

/* JADK INFO: Access modifiers changed from: protected */
@Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.SupportActiv
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_binary_patching);
    if (this.isAdmin) {
        TextView isAdminText = (TextView) _$findCachedViewById(R.id.isAdminText);
        isAdminText.setText("You are Admin Now");
        Button adminButton = (Button) _$findCachedViewById(R.id.adminButton);
        isAdminText.setOnClickListener(adminButton);
        adminButton.setEnabled(true);
    }
    ((Button) _$findCachedViewById(R.id.adminButton)).setOnClickListener(new View.OnClickListener() { // from class: awasp.sat.ag
        @Override // android.view.View.OnClickListener
        public final void onClick(View v) {
            Toast.makeText(BinaryPatchingActivity.this, "You clicked on Adminstration button", 1).show();
        }
    });
}
}

```

Firebase databases provide the ability to the developers to store data and sync the stored data into the NoSQL cloud-hosted database efficiently. A network call can help in identifying a misconfigured firewall dataset.



```

47 <string name="abc_search_hint">Search...</string>
48 <string name="abc_searchview_description_clear">Clear query</string>
49 <string name="abc_searchview_description_query">Search query</string>
50 <string name="abc_searchview_description_search">Search</string>
51 <string name="abc_searchview_description_submit">Submit query</string>
52 <string name="abc_searchview_description_voice">Voice search</string>
53 <string name="abc_shareactionprovider_share_with">Share with</string>
54 <string name="abc_shareactionprovider_share_with_application">Share with %s</string>
55 <string name="abc_toobar_collapse_description">Collapse</string>
56 <string name="activity">Unprotected Android Components</string>
57 <string name="app_name">Androgoat - Insecure App (Kotlin)</string>
58 <string name="clipboard">Clipboard - Copy and Paste</string>
59 <string name="clipboardHint">Objectives: \n 1. What is Clipboard? \n 2. Vulnerable Code \n 3. Risk associated with it</string>
60 <string name="dict">Keyboard Cache</string>
61 <string name="dictHint">Objectives: Find where keystroke logs, for username and Password, saved in device</string>
62 <string name="emulator">Emulator Detection</string>
63 <string name="firebasaur">https://androgoat-42597.firebaseio.com/</string>
64 <string name="github">Code is Available at Github (https://github.com/satishpatnayak/AndroGoat)</string>
65 <string name="hardcode">Hardcode Issue</string>
66 <string name="logging">Insecure Logging</string>
67 <string name="mail">For any suggestions and feedback please satishkumarpatnayak@live.com</string>
68 <string name="network">Network Intercepting</string>
69 <string name="oscd">Input Validations - OS CMD Injection</string>
70 <string name="root">Root Detection</string>
71 <string name="rootHint">Objectives: \n 1. Understand what is Rooting \n 2. How app can detect if device is Rooted? \n 3. Bypass Root</string>
72 <string name="sdCard">External Storage - SDCard</string>
73 <string name="search_menu_title">Search</string>
74 <string name="sp1">Shared Preferences - Part1</string>
75 <string name="sp2">Shared Preferences - Part2</string>
76 <string name="sql">SQLite</string>
77 <string name="sql1">Input Validations - SQLi</string>
78 <string name="status_bar_notification_info_overflow">999</string>
79 <string name="tempFile">Temp File</string>

```

Professionals copy the URL and paste it into the browser to acknowledge the output as shown in the picture below



Professional checks each code according to the API keys, internal IP address, hard-coded certifications, usernames, passwords and many other things.

WebView in an Android application helps developers in fixing the web browser in their Android application. It allows developers to directly analyse the internal user interface by displaying the web content related to different websites and web pages.

```

public final void onClick(View v) {
    WebSettings webSettings1 = webView11.getSettings();
    Intrinsics.checkNotNullExpressionValueIsNotNull(webSettings1, "webSettings1");
    webSettings1.setJavaScriptEnabled(true);
    TextView url2 = url;
    Intrinsics.checkNotNullExpressionValueIsNotNull(url2, "url");
    webView11.loadUrl(url2.getText().toString());
}

```

Unlocking Secrets: Exploring/Exploiting Internal Components with Drozer

Drozer defines it as a security analysis framework that provides an assistant to acknowledge the cyber-attacks and their potential impact vulnerabilities, along with testing security controls by third-party applications. After the installation of Drozer, the procedures help in recognising the interaction between different components.

Firstly, the shown button needs to be on in the Drozer agent.

Need the full report with all the bits and bytes?

🔗 Find the download link in the comments 