Signing Info Verification

Severity - MEDIUM

Status - FAIL

OWASP Reference 2017 v1.0

M9 - Reverse Engineering

Compliance/Security Control Reference

NIST SP800-53 R2 SC-18

Abstract

An App signed with debug key cannot be published to Google Play Store. However, it can be distributed via unauthorized parties or stores.

Vulnerability Description

If Android App is signed with a debug key that is created by the Android SDK build tools. If debug mode is used to build your App, the SDK tools invoke Keytool utility (included in the JDK) available publicly to create the debug keystore/key with predetermined names/passwords: Keystore name: "debug.keystore" Keystore password: "android" Key alias: "androiddebugkey" Key password: "android" CN: "CN=Android Debug,O=Android,C=US" To release the App for end-users, it must be signed with a suitable private key owned by a developer as it cannot be published to "Google Play Store".

Instance(s)

CN=Android Debug found in signing key. App signed using a debug keystore

Recommendations

Developers should build the App in release mode and use their own private key to sign the App.

Android Xml Content Verification

Severity - MEDIUM

Status - FAIL

OWASP Reference 2017 v1.0

M1 - Improper Platform Usage

Compliance/Security Control Reference

PCI DSS v1.2 5.1 HIPAA 164.308 (a)(5)(ii)(B) NIST SP800

Abstract

A Service is an App component that can perform long-running operations in the background and does not provide a user interface. Another App component can start a service and it will continue to run in the background even if the user switches to another App. Any malicious App can access the service of the App if proper permissions are not set for the App services, which may lead to breach of Integrity.The allowBackup attribute determines if an App data can be backed up and restored on external memory. This may have security consequences for an App. Once the data is backed up, all App data can be read by the malicious user, which might contain sensitive information and can be read or manipulated by the attacker. This may lead to breach of Confidentiality and Privacy.Android allows a developer to debug the App during development. The debugging mode can be activated by setting 'android:debuggable' attribute to true in AndroidManifest.xml file of the App. An attacker can establish a connection between debugger and malicious App, which may allow manipulation of the legitimate App. This can result in the execution of an arbitrary code and undesirable disclosure of sensitive data, which may lead to breach of Integrity and Confidentiality.

Vulnerability Description

Characterizes the potential risk implied in the permission and indicates the procedure the system should follow when determining whether or not to grant the permission to an application requesting it.Normal,Dangerous,Signature and Signature or System are the base permission types.Each protection level consists of a base permission type and zero or more flags. For example, the "dangerous" protection level has no flags. In contrast, the protection level "signature|privileged" is a combination of the "signature" base permission type and the "privileged" flag.A permission that the system grants only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.

Instance(s)

android_debuggable set to true : android:debuggable="true" activity with name com.ibm.android.analyzer.test.unsafereflection.UnsafeReflection has android:exported set to true activity with name com.ibm.android.analyzer.test.activityhijacking.ActivityHijacking has android:exported set to true activity with name com.ibm.android.analyzer.test.androidfragmentinjection.AndroidFragmentInjection has android:exported set to true activity with name com.ibm.android.analyzer.test.weakrandom.WeakRandom has android:exported set to true activity with name com.ibm.android.analyzer.test.servicehijacking.ServiceHijackingNonVulnerable has android:exported set to true activity with name com.ibm.android.analyzer.test.servicehijacking.ServiceHijacking has android:exported set to true activity with name com.ibm.android.analyzer.test.dos.DoS has android:exported set to true activity with name com.ibm.android.analyzer.test.dos.DoSUri has android:exported set to true activity with name com.ibm.android.analyzer.test.insecureencryption.InsecureEncryption has android:exported set to true activity with name com.ibm.android.analyzer.test.insecureencryption.SecureEncryptionWithMac has android:exported set to true activity with name com.ibm.android.analyzer.test.redos.Redos has android:exported set to true activity with name com.ibm.android.analyzer.test.regexinjection.RegexInjection has android:exported set to true activity with name com.ibm.android.analyzer.test.xpathinjection.XpathInjection has android:exported set to true activity with name com.ibm.android.analyzer.test.xpathinjection.XpathInjectionSafe has android:exported set to true activity with name com.ibm.android.analyzer.test.mitmphishing.MiTMPhishing has android:exported set to true activity with name com.ibm.android.analyzer.test.broadcasttheft.BroadcastTheft has android:exported set to true activity with name com.ibm.android.analyzer.test.ifp.InsecureFilePermissions has android:exported set to true activity with name com.ibm.android.analyzer.test.androidclasshijacking.AndroidClassHijacking has android:exported set to true activity with name com.ibm.android.analyzer.test.nativevuln.BufferOverflow has android:exported set to true activity with name com.ibm.android.analyzer.test.nativevuln.FormatString has android:exported set to true activity with name com.ibm.android.analyzer.test.uispoofing.UISpoofing has android:exported set to true activity with name com.ibm.android.analyzer.test.connectmanipulation.ConnectManipulation has android:exported set to true activity with name com.ibm.android.analyzer.test.intentinjection.IntentInjection has android:exported set to true activity with name

com.ibm.android.analyzer.test.intentinjection.IntentInjectionSafe has android:exported set to true activity with name
com.ibm.android.analyzer.test.sqlinjection.SqlInjection has android:exported set to true activity with name
com.ibm.android.analyzer.test.sqlinjection.SqlInjectionSafe has android:exported set to true activity with name
com.ibm.android.analyzer.test.cmdinjection.CommandInjection has android:exported set to true activity with name
com.ibm.android.analyzer.test.cmdinjection.LoadLibrary has android:exported set to true activity with name
com.ibm.android.analyzer.test.intentextras.IntentExtras has android:exported set to true activity with name
com.ibm.android.analyzer.test.disabledcomponents.DisabledActivity has android:exported set to true activity with name
com.ibm.android.analyzer.test.flows.StringToFile has android:exported set to true activity with name
com.ibm.android.analyzer.test.formatstring.FormatString has android:exported set to true activity with name
com.ibm.android.analyzer.test.informationleakage.ExternalStorageWriting has android:exported set to true service with name
com.ibm.android.analyzer.test.dos.DoSService has android:exported set to true service with name
com.ibm.android.analyzer.test.nativevuln.BufferOverflowService has android:exported set to true service with name
com.ibm.android.analyzer.test.nativevuln.FormatStringService has android:exported set to true receiver with name
com.ibm.android.analyzer.test.sqlinjection.SqlInjectionReceiver has android:exported set to true activity with name
com.ibm.android.analyzer.test.ipi.InsecurePendingIntentNonVulnerable has android:exported set to true activity with name
com.ibm.android.analyzer.test.libraryloading.ExternalLibraryLoading has android:exported set to true

Recommendations

Set "android: exported" = "false" for all Services which do not interact with other Apps.Specify appropriate permissions, if it is required to allow other Apps to invoke the service or interact with it.If different permissions are needed for different callable Service methods, insert checkCallingPermission() calls in the methods. The following example demonstrates how to set permission in Service tag:>Example:service android:name=".MailListenerService" android:permission="com.example.perm.BIND_TO_MSG_LISTENER" android:enabled="true" android:exported="true"> service> Developer should explicitly set 'android:allowBackup' to 'false' under App tag in Androidmanifest.xml. The following code snippet demonstrates how to disable App backup:Example:App android:icon="@drawable/icon" android:allowBackup="false" >Set "android:debuggable" = "false" in application tag of App Manifest file to prevent run time manipulation by an attacker or malware

Permissions Scan	Severity - MEDIUM Status - FAIL

OWASP Reference 2017 v1.0

M1 - Improper Platform Usage

Compliance/Security Control Reference

PPCI DSS v1.2 7.1.1 HIPAA 164.308 (a)(3)(ii)(A) NIST SP

Abstract

Developer can define custom permissions to protect publicly available activities and custom data providers in App. If protection level of custom permission is not declared securely, then the permission gives access to private data or has another potentially negative impact such as using contact list, accessing calendar and so on. This may result in loss of Confidentiality.

Vulnerability Description

There are different protection levels, using which an App can declare access to available activities and custom data providers. Apps having custom permission as normal or dangerous are prone to security attacks.The normal custom permission is a default value. It gives requesting Apps, access to isolated App-level features. The system automatically grants this type of permission to a requesting App at installation, without asking for the user's explicit approval. Whereas dangerous custom permission gives a requesting App access to private user data or control over the device that can negatively impact the user.

Instance(s)

Dangerous permissions found : "com.ibm.android.analyzer.test.READ_FILE", "com.ibm.android.analyzer.test.WRITE_FILE", ".INTERNET", ".ACCESS_NETWORK_STATE", ".ACCESS_FINE_LOCATION", ".ACCESS_FINE_LOCATION", ".ACCESS_COARSE_LOCATION", ".RECEIVE_SMS", ".RECEIVE_SMS", ".READ_EXTERNAL_STORAGE", ".READ_EXTERNAL_STORAGE", ".WRITE_EXTERNAL_STORAGE", ".WRITE_EXTERNAL_STORAGE", ".READ_PHONE_STATE", ".READ_PHONE_STATE", ".SEND_SMS", ".SEND_SMS",

Recommendations

Have 'signature' based custom permission. The following code snippet demonstrates how to set custom permission: Example:permission android:name="com.example.perm.READ_INCOMING_MSG" android.protectionLevel="signature" android:permissionGroup="android.permission-group.PERSONAL_INFO"/> Reference: http://developer.android.com/guide/topics/manifest/permission-element.html

Smali Files Scan

Severity - MEDIUM

Status - PASS

OWASP Reference 2017 v1.0

M10 - Lack of Binary Protections

Compliance/Security Control Reference

Not Applicable

Abstract

Obfuscation is the deliberate act of creating obfuscated code, i.e. source or machine code that is difficult for humans to understand

Vulnerability Description

Check for the unencrypted data present in smali files.Programmers must deliberately obfuscate code to conceal its purpose or its logic, in order to prevent tampering, deter reverse engineering, or as a puzzle or recreational challenge for someone reading the source code

Instance(s)

Recommendations

Use Dotfuscator or similar tools to obfuscate the code before delivering to the client.

Asset Folder Scan	Severity - MEDIUM Status - PASS

OWASP Reference 2017 v1.0

M10 - Lack of Binary Protections

Compliance/Security Control Reference

NIST SP800-53 R2 CM-5

Abstract

In CORS, XMLHttpRequest are allowed to be sent across domains without asking the user for permission, thus actually requests are sent without the user noticing them. This can be used to break the security requirement Access control through abusing a user session. This might lead to breach of Accountability and Integrity.Malicious apps / Attackers can modify the app's presentation layer (HTML/JS/CSS) of the app within the phone and execute modified JavaScript. This allows attackers/malious apps to perform some hidden malicious activity.

Vulnerability Description

Check that "Allow-Intent-Navigation" and "Access-Control-Allow-Origin" is not set to *.Also check for the presence of config.xml and Network-Security-Config.xml file.Cross-Origin Resource Sharing (CORS) enables clients making cross origin requests using XMLHttpRequests. It makes it possible to send XMLHttpRequests across domains if a new HTTP header which is called 'Access-Control-Allow-Origin' is defined. With this HTTP header a website can allow to be accessed by an XMLHttpRequest sent from JavaScript running under a foreign domain. A web App built out of many parts of different origins can send requests using XMLHttpRequest to foreign domains as well to update the data on the Client.The app must be able to detect code tampering at runtime. The app must be able to react appropriately at runtime to a code integrity violation, otherwise This can lead to breach of Confidentiality and Integrity of Data.

Instance(s)

Recommendations

Do not set the header "Access-Control-Allow-Origin" and "Allow-Intent-Navigation" to *.Set the Access-Control-Allow-Origin header to specific domain name(s)Example:Access-Control-Allow-Origin: www.mydomain.comDo not base access control on the origin header. An attacker through sending a faked origin header can modify this header.App can consist of zipped www folder. This folder must be unzipped at , runtime when it is needed. Only App native source code should be able to extract/access its files.