Android Security Report

For Internal Purpose

Prepared For: com.foo.bar

Table Of Contents

Report Summary
Application Details
Audit Summary
Appknox Security Rating
Do not release apps that are debuggable
Noncompliant Code Example
Compliant Solution
Business Implication
Related Vulnerabilities
Do not release apps that are debuggable
Noncompliant Code Example
Compliant Solution
Business Implication
Related Vulnerabilities

Report Summary

Appknox conducted a security assessment of the mobile application for the Android platform. This report contains all the findings during the automated auditing process. It also contains the process of discovering those vulnerabilities in the first place, and ways to remediate those issues.

Application Details

Application Name	com.foo.bar
Application Namespace	com.foo.bar
Version	1.0
Audit Date	2017-03-03 09:17:31.568149+00:00
Application SHA1 Hash	d5341a898eb1d1e90a316d49e8f6b4f33ad08cf0
Application MD5 Hash	97d83c689f2cb0118e1901054165e8fe

Audit Summary

Application Debug Enabled Debugging was enabled on the app which makes it easier for reverse engineers to hook a debugger to it. This allows dumping a stack trace and accessing debugging helper classes.	High Risk
Application Debug Enabled Debugging was enabled on the app which makes it easier for reverse engineers to hook a debugger to it. This allows dumping a stack trace and accessing debugging helper classes.	High Risk

Priority Level	Number of failed test cases
High Risk	
Medium Risk	
Low Risk	

Appknox Security Rating

Appknox Security Rating: Unsecured

Do not release apps that are debuggable

Android allows the attribute `android:debuggable` to be set to true so that the app can be debugged. By default this attribute is disabled, i.e., it is set to false, but it may be set to true to help with debugging during development of the app. However, an app should never be released with this attribute set to true as it enables users to gain access to details of the app that should be kept secure. With the attribute set to true, users can debug the app even without access to its source code.

Risk Rating : High

Risk Assessment

Debugging was enabled on the app which makes it easier for reverse engineers to hook a debugger to it. This allows dumping a stack trace and accessing debugging helper classes.

```
AllowAllHostnameVerifier is instantiated in
org.apache.http.conn.ssl.SSLConnectionSocketFactory->
```

Noncompliant Code Example

This non-compliant code example shows an app that has the `android:debuggable` attribute set to true being accessed to reveal sensitive data.

```
$ adb shell
shell@android:/ $ run-as com.example.someapp sh
shell@android:/data/data/com.example.someapp $ id
uid=10060(app_60) gid=10060(app_60)
shell@android:/data/data/com.example.someapp $ ls files/
secret_data.txt
shell@android:/data/data/com.example.some $ cat files/
secret_data.txt
password=GoogolPlex
account_number=31974286
```

Clearly, with the `android:debuggable` attribute set to true, sensitive date related to the app can be revealed to any user.

Compliant Solution

Ensure that the `android:debuggable` attribute is set to false before the app is released:

```
android:debuggable="false"
```

Note that some development environments (including Eclipse/ADT and Ant) automatically set `android:debuggable` to true for incremental or debugging builds but set it to false for release builds.

Business Implication

Application can be debugged and reverse engineers can debug and manipulate the Runtime logic of the application.

Related Vulnerabilities

Do not release apps that are debuggable

Android allows the attribute `android:debuggable` to be set to true so that the app can be debugged. By default this attribute is disabled, i.e., it is set to false, but it may be set to true to help with debugging during development of the app. However, an app should never be released with this attribute set to true as it enables users to gain access to details of the app that should be kept secure. With the attribute set to true, users can debug the app even without access to its source code.

Risk Rating : High

Risk Assessment

Debugging was enabled on the app which makes it easier for reverse engineers to hook a debugger to it. This allows dumping a stack trace and accessing debugging helper classes.

```
AllowAllHostnameVerifier is instantiated in
org.apache.http.conn.ssl.SSLConnectionSocketFactory->
```

Noncompliant Code Example

This non-compliant code example shows an app that has the `android:debuggable` attribute set to true being accessed to reveal sensitive data.

```
$ adb shell
shell@android:/ $ run-as com.example.someapp sh
shell@android:/data/data/com.example.someapp $ id
uid=10060(app_60) gid=10060(app_60)
shell@android:/data/data/com.example.someapp $ ls files/
secret_data.txt
shell@android:/data/data/com.example.some $ cat files/
secret_data.txt
password=GoogolPlex
account_number=31974286
```

Clearly, with the `android:debuggable` attribute set to true, sensitive date related to the app can be revealed to any user.

Compliant Solution

Ensure that the `android:debuggable` attribute is set to false before the app is released:

```
android:debuggable="false"
```

Note that some development environments (including Eclipse/ADT and Ant) automatically set `android:debuggable` to true for incremental or debugging builds but set it to false for release builds.

Business Implication

Application can be debugged and reverse engineers can debug and manipulate the Runtime logic of the application.

Related Vulnerabilities