FH University of Applied Sciences TECHNIKUM WIEN

Pentest Report

FHTW

Date: 2024-10-08

Project-Number: 01

Version: 1.0.0

Classification: High Risk

Contents

Executive Summary

The purpose of this report is to find vulnerabilities conducting penetration tests for the app “InjuredAndroid”. For finding and assessing the vulnerabilities, we decided to use the following utilities:

- VisualCodeGrepper
- mobsf.live

In order for mobsf.live to work, we only needed to provide the .apk file. VisualCodeGrepper required us to extract the source code (Java). We used dex2jar and ADB for this.

The pentest found several (severe) vulnerabilities, such as a **Java Runtime vulnerability** that could allow for remote code execution, an **exported activity** allowing unauthorized access from other apps, improper **network security settings** exposing the app to data interception, using an **outdated Android version** and a misconfigured **backup flag**, risking data theft through ADB. More information about the findings, recommendations and possible solutions can be found below and are summarized in the last chapter of the report, ‘***Summary’***.

Assessment Overview

Contact Information

Martin Stropp, if22b197@technikum-wien.at

Philipp Wudernitz, if22b230@technikum-wien.at

Task

The task was to identify, categorize and review potential vulnerabilities and security risks in the app “InjuredAndroid” and also provide recommendations and possible solutions. The code quality was also partially assessed during the test, but this was not our priority. We focused on identifying and listing the most critical findings in this report.

Scope

A Samsung Galaxy S5 with Android 6.0 was used as the text subject. ADB (USB debugging) was enabled on the device. In order to simulate the device, genymotion and Virtual Box was used, although Virtual Box was only needed for genymotion to work.

Positive Findings

No trackers and malwares have been found.

No critical data is being stored in the logs.

Procedure

Methods und Approach

Genymotion was started (as well as Virtual Box) and the virtual machine (Samsung S5 with Android 6.0) was launched. A connection to the laptop was established using ADB (USB debugging) to retrieve the APK and subsequently the source code of “InjuredAndroid” using dex2jar.

After that, we used VisualCodeGrepper to find vulnerabilities using the source code. Both a .txt and .xml file including the findings were generated and assessed in this report.

Finally, we uploaded the .apk file to mobsf.live and generated the results. The findings on the website were used in this report.

Tools

genymotion: Virtualization of the Samsung S5 for the pentests

ADB: Help with debugging of android apps and the device using USB

mobsf.live: browser software for analyzing APK files of applications and creating a ‘report’

VisualCodeGrepper: software for analyzing .java files and creating a ‘report’

dex2jar: decompiles .apk files to .jar files

Severity Rating

The CVSS-Score system was used to assess the severity of the found vulnerability. The scale looks like the following:

- Low: 0.1-3.9
- Medium: 4.0-6.9
- High: 7-8.9
- Critical: 9.0-10.0

The higher the score, the more serious is the vulnerability. High and critical findings must be addressed as fast as possible, because they allow for severely dangerous attack opportunities.

VisualCodeGrepper

Findings

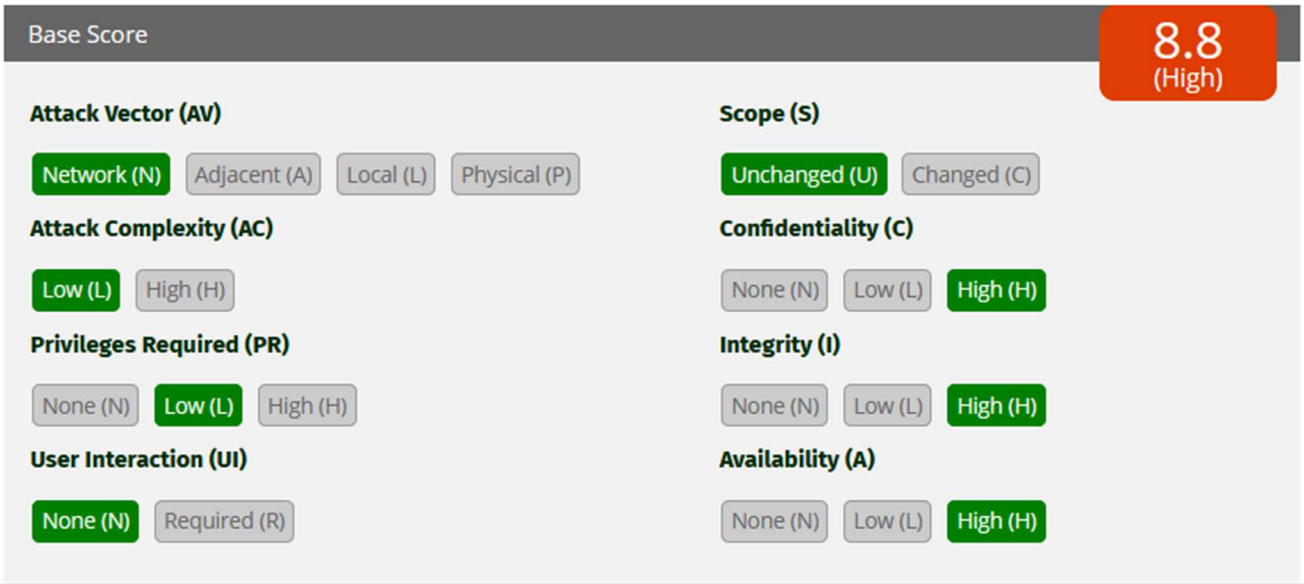
High Findings

<H1>: < Potentially Unsafe Code - java.lang.Runtime.exec Gets Path from Variable >

Description

The code in Runtime.exec() uses a dynamic path created from a variable (sb.toString()). This allows for attackers to inject malicious filenames into the program, which could lead to remote code execution.

CVSS-Score



Probability

Effects

After exploiting this vulnerability, the attacker could execute commands on the device. This might result in:

- Data theft (e.g. sensitive / confidential information)
- Modification of the device / operating system
- Service disruption (denial of service, …)

Proof of Concept

HIGH: Potentially Unsafe Code - java.lang.Runtime.exec Gets Path from Variable

Line: 311 - C:\Users\Martin\Documents\mobSec2024\484e0260e1e7ac5e295595986b16f603-java\b3nac\injuredandroid\RCEActivity.java

The pathname used in the call appears to be loaded from a variable. Check the code manually to ensure that malicious filenames cannot be submitted by an attacker.

```
Process exec = runtime.exec(sb.toString());
```

Recommendations

Avoid using dynamic paths in Runtime.exec() and potentially implement stricter input validation.

Medium Findings

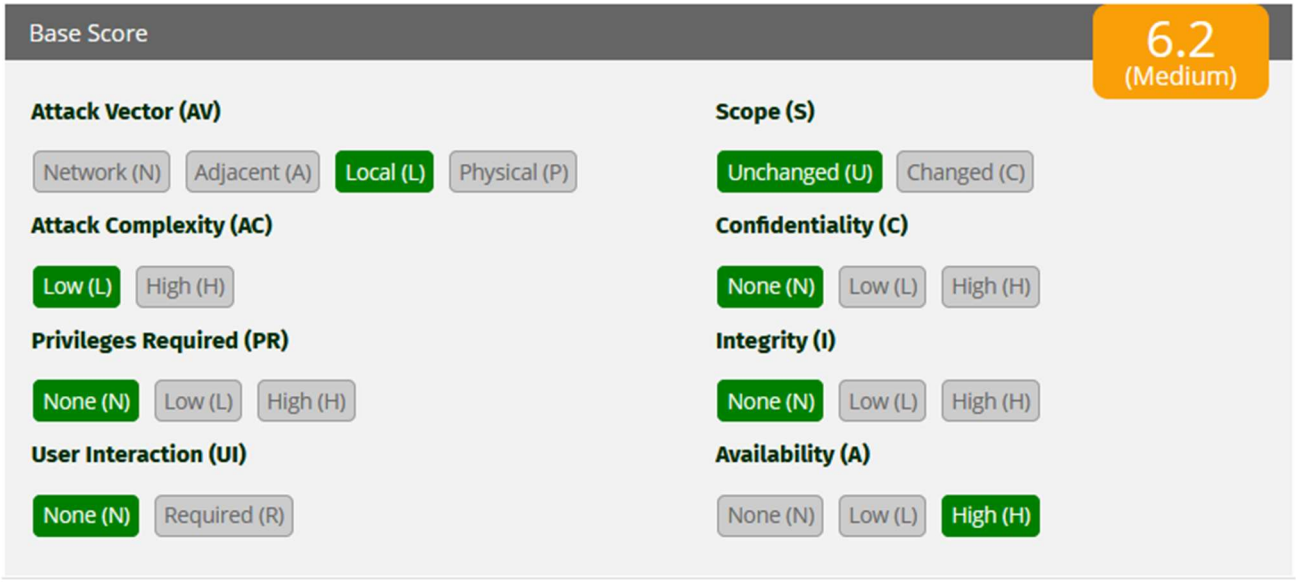
<M1>: < Potentially Unsafe Code - Failure To Release Resources In All Cases>

Description

The code lacks a *finally* block. This block is used to release resources if an exception / error occurs. Not using *finally* could lead to resource leaks and potentially Denial of Service.

Probability

CVSS-Score



Effects

Exploitation could lead to resource exhaustion, which might slow down or even crash the device. Results could be:

- Denial of Service
- Affected availability
- Impacted system performance

Proof of Concept

MEDIUM: Potentially Unsafe Code - Failure To Release Resources In All Cases

Line: 17 - C:\Users\Martin\Documents\mobSec2024\484e0260e1e7ac5e295595986b16f603-java\com\google\crypto\tink\BinaryKeysetWriter.java

There appears to be no 'finally' block to release resources if an exception occurs, potentially resulting in DoS conditions from excessive resource consumption.

Recommendations

Implement a *finally* block to ensure that the resources are released properly, even if an exception or error occurs.

Low Findings

<L1>: < Potentially Unsafe Code - Operation on Primitive Data Type >

Description

This code might lead to an integer overflow and unpredictable behavior, which could cause system anomalies or even crashes.

Probability

CVSS-Score

Effects

Exploitation of this vulnerability could lead to wrong calculations, crashes and unpredictable behavior. This could allow attackers to inject malicious code to manipulate the program, which could result in data corruption or security / data breaches.

Proof of Concept

LOW: Potentially Unsafe Code - Operation on Primitive Data Type

Line: 108 - C:\Users\Martin\Documents\mobSec2024\484e0260e1e7ac5e295595986b16f603-java\com\google\crypto\tink\JsonKeysetReader.java

The code appears to be carrying out a mathematical operation on a primitive data type. In some circumstances this can result in an overflow and unexpected behaviour. Check the code manually to determine the risk.

```
for (int i = 0; i < jSONArray.length(); i++) {
```

Recommendations

Implement overflow checks and use appropriate data types.

mobsf.live

Findings

Critical Findings

<C1>: < App can be installed on a vulnerable unpatched Android version >

Description

This application can be installed on an older version of android that has multiple unfixed vulnerabilities. These devices won't receive reasonable security updates from Google. Support an Android version => 10, API 29 to receive reasonable security updates.

Probability

CVSS-Score

Effects

Well known exploits (in older Android versions) can be leveraged by an attacker, which could lead to:

- Data theft
- Denial of Service
- Destruction of the integrity of the system

Proof of Concept

1	App can be installed on a vulnerable upatched Android version Android 6.0-6.0.1, [minSdk=23]	high	This application can be installed on an older version of android that has multiple unfixed vulnerabilities. These devices won't receive reasonable security updates from Google. Support an Android version => 10, API 29 to receive reasonable security updates.

Recommendations

The App should not be installed on older / not supported Android versions. Publishers should update the code to newer Android standards and set the minimum supported Android version higher.

High Findings

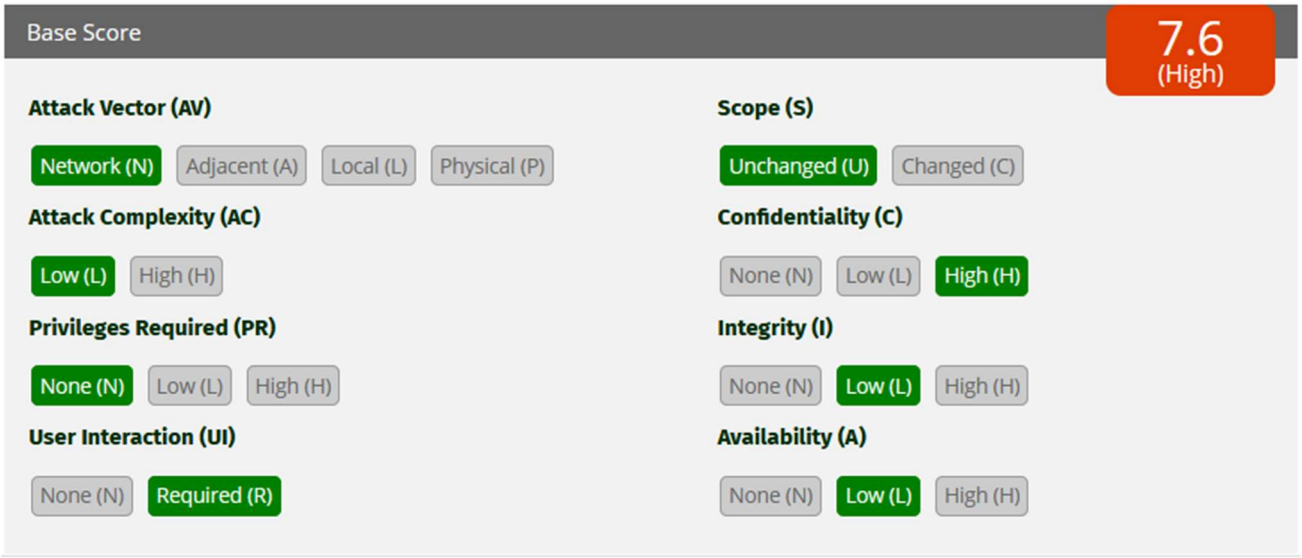
<H1>: < App Link assetlinks.json file not found>

Description

App Link asset verification URL (http://b3nac.com/.well-known/assetlinks.json) not found or configured incorrectly. (Status Code: None). App Links allow users to redirect from a web URL/email to the mobile app. If this file is missing or incorrectly configured for the App Link host/domain, a malicious app can hijack such URLs. This may lead to phishing attacks, leak sensitive data in the URI, such as PII, OAuth tokens, magic link/password reset tokens and more. You must verify the App Link domain by hosting the assetlinks.json file and enabling verification via [android:autoVerify="true"] in the Activity intent-filter.

Probability

CVSS-Score



Effects

Without the assetlinks.json file, attackers could hijack URLs, deploy phishing attacks and expose sensitive data like tokens. This compromises user privacy and might lead to unauthorized access and data breaches.

Proof of Concept

4	App Link assetlinks.json file not found [android:name=b3nac.injuredandroid.CSPBypassActivity] [android:host=http://b3nac.com]	high	App Link asset verification URL (http://b3nac.com/.well-known/assetlinks.json) not found or configured incorrectly. (Status Code: None). App Links allow users to redirect from a web URL/email to the mobile app. If this file is missing or incorrectly configured for the App Link host/domain, a malicious app can hijack such URLs. This may lead to phishing attacks, leak sensitive data in the URI, such as PII, OAuth tokens, magic link/password reset tokens and more. You must verify the App Link domain by hosting the assetlinks.json file and enabling verification via [android:autoVerify="true"] in the Activity intent-filter.

Recommendations

Ensure that the assetlinks.json file is correctly configured and a verification is implemented to prevent unauthorized programs to potentially hijack URLs.

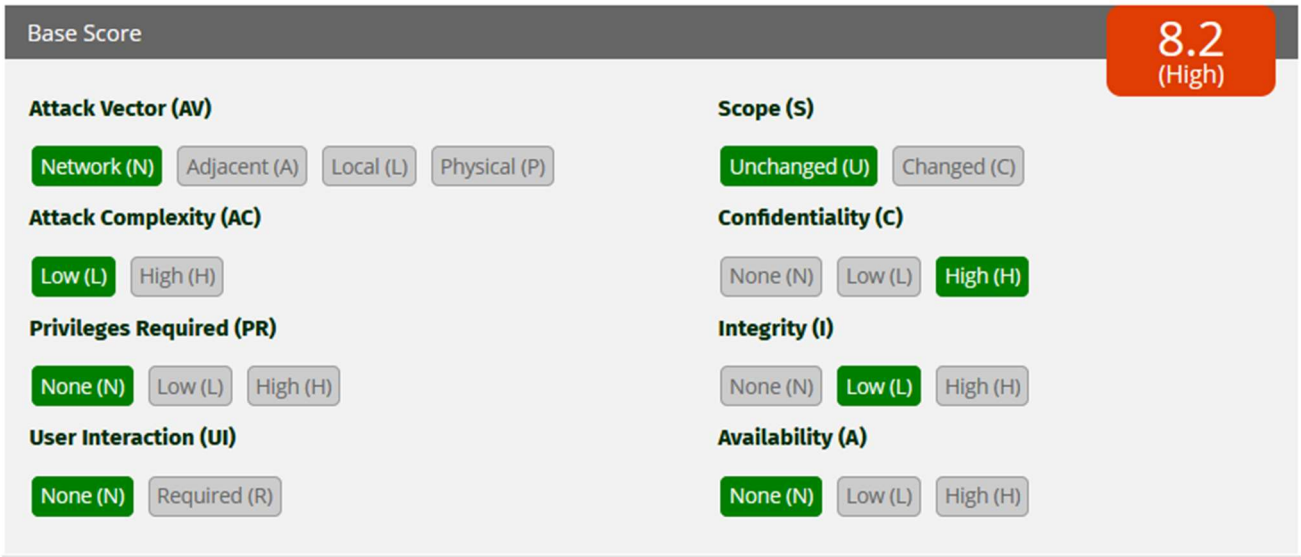
<H2>: < App has a Network Security Configuration>

Description

The Network Security Configuration feature lets apps customize their network security settings in a safe, declarative configuration file without modifying app code. These settings can be configured for specific domains and for a specific app.

Probability

CVSS-Score



Effects

A wrong configuration of this could lead to data interception / man-in-the-middle attacks and exposure to malicious networks. This can cause data breaches and jeopardize confidential user information.

Proof of Concept

2	App has a Network Security Configuration [android:networkSecurityConfig=@xml/network_security_config]	info	The Network Security Configuration feature lets apps customize their network security settings in a safe, declarative configuration file without modifying app code. These settings can be configured for specific domains and for a specific app.

Recommendations

Enforce https, use and validate certificates and do not allow for unencrypted traffic.

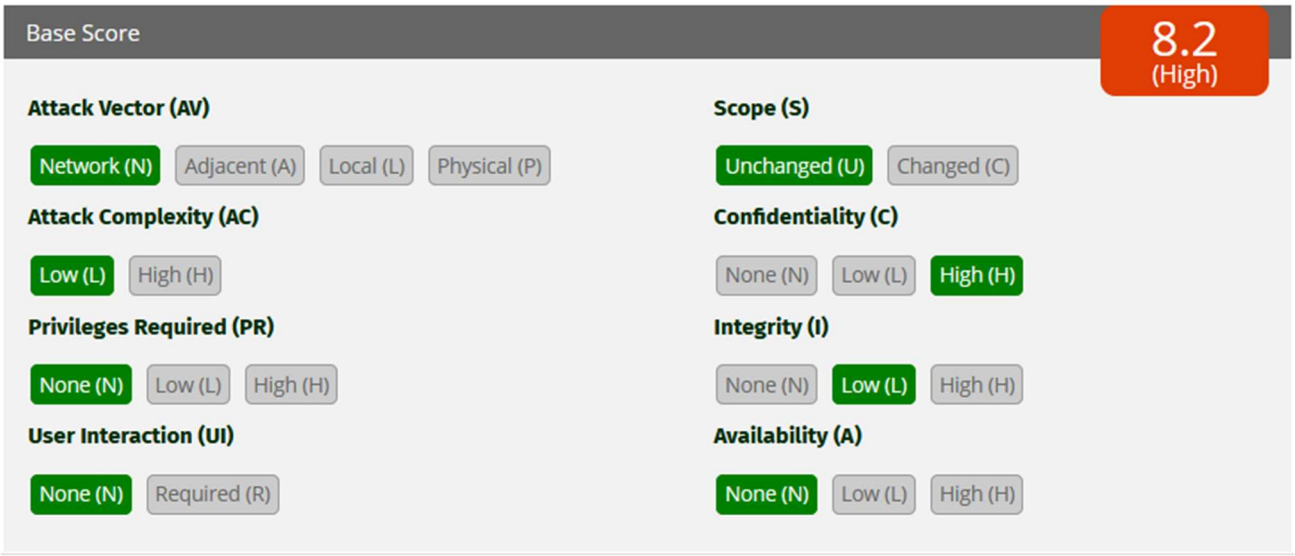
*<H3>: < **Activity** (b3nac.injuredandroid.CSPBypassActivity) is not Protected>*

Description

An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Activity is explicitly exported.

Probability

CVSS-Score



Effects

Since there is no protection for this exported activity, unauthorized programs (installed on the device) could invoke the activity, leading to data breaches, data manipulation and malicious actions from an attacker.

Proof of Concept

6	**Activity** (b3nac.injuredandroid.CSPBypassActivity) is not Protected. An intent-filter exists.	warning	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Activity is explicitly exported.

Recommendations

The intent-filter should be removed or a solid access control using permissions should be implemented to prevent other applications from accessing exported activities. It is recommended to not export and share any activities unless it is absolutely necessary.

Medium Findings

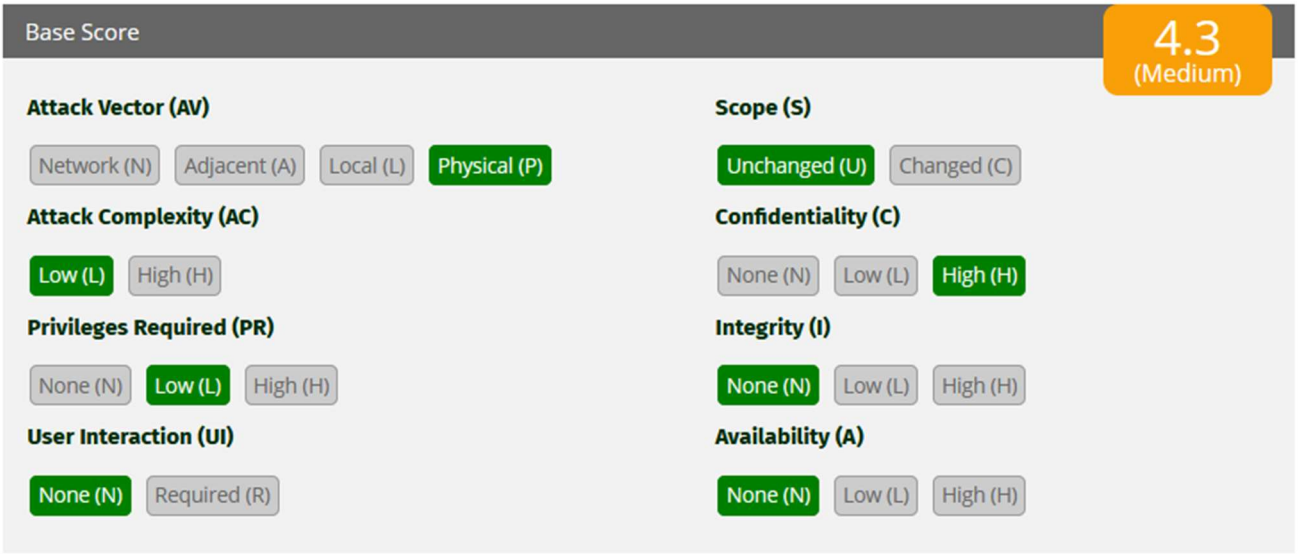
<M1>: < Application Data can be Backed up>

Description

The flag [android:allowBackup] should be set to false. By default it is set to true and allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.

Probability

CVSS-Score



Effects

If the flag is set to true, sensitive user data can be indirectly extracted from the device via ADB by an attacker. This could lead to data theft, unauthorized access and privacy violations.

Proof of Concept

3	Application Data can be Backed up [android:allowBackup] flag is missing.	warning	The flag [android:allowBackup] should be set to false. By default it is set to true and allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.

Recommendations

Set the flag to false to prevent unauthorized access through ADB (USB debugging).

Application Permissions

PERMISSION	STATUS	INFO
android.permission.READ_EXTERNAL_STORAGE	dangerous	read external storage contents
android.permission.READ_PHONE_STATE	dangerous	read phone state and identity
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/ delete external storage contents
android.permission.ACCESS_NETWORK_STATE	normal	view network status
android.permission.INTERNET	normal	full Internet access

These permissions does “InjuredAndroid” receive from the system:

- READ_EXTERNAL_STORAGE: allows it to access and read files on external storages
- READ_PHONE_STATE: allows it to access phones status, e.g. battery information
- WRITE_EXTERNAL_STORAGE: allows it to access and write files on external storages
- ACCESS_NETWORK_STATE: allows it to access network status and connections
- INTERNET: allows it to access the internet

Summary

Positive Findings: The app demonstrates solid functionality and does not contain trackers or malware, non the less the security configuration requires serious enhancement.

Vulnerabilities:

1. **Java Runtime Vulnerability**: A dynamic path in *Runtime.exec()* is created from a variable, which could allow for the remote execution of malicious code.
2. **Exported Activity:** An Activity was exported and shared with other applications on the device, which allows for unauthorized access from other apps.
3. **Network Security Configuration:** Improper configurations might lead to data interception or man-in-the-middle attacks.
4. **Backup Flag:** The *allowBackup* flag was set to true, which enabled the creation of unauthorized backup and retrieval of sensitive data using ADB (USB debugging).
5. **Android Version:** The app could be installed on older / unsupported Android versions using outdated APKs, which offers more opportunities for potential attacks to succeed.

Recommendations:

- Do not use dynamic paths in *Runtime.exec()* and implement a strict input validation.
- If possible, do not export activities unless it is absolutely needed. If it is needed, implement strict access control to prevent unauthorized accessibility from other apps on the device.
- Implement strict access control by using https and certificates. Make sure that there is no unencrypted network traffic.
- Set *allowBackup* to false to prevent attackers to possibly create a backup and retrieve sensible data over ADB.
- Ensure that “InjuredAndroid” does not run on an unsupported / outdated Android version. Update the APK and code accordingly.