

Introduction

Static analysis is a technique used to evaluate the security of an application by examining its source code without executing it. In the context of Android applications, static analysis helps identify potential vulnerabilities, coding errors, and violations of secure coding practices by analyzing APK contents, Smali code, or decompiled Java code. It is commonly used as part of a broader security auditing process to detect issues early in the development or assessment lifecycle.

Benefits of Static Analysis

Benefit	Description
Early Detection	Static analysis can be performed in the early stages of development, even before the application is fully functional. This allows for the early detection of issues, making them easier and less costly to address.
Complete Coverage	Unlike dynamic analysis, which only tests the parts of the application that are executed during the test run, static analysis can cover the entire codebase, including paths that are rarely executed.
Security Vulnerability Detection	Static analysis tools can detect a wide range of security vulnerabilities, such as SQL injection, buffer overflow, and insecure permissions.

How it Works

Static analysis tools parse an application's code and evaluate it against a set of predefined rules or patterns to identify potential issues. These tools use techniques such as data flow analysis, control flow analysis, and semantic analysis to understand the behavior of the code. Static analysis for Android applications often involves examining the Java source code, compiled DEX (Dalvik Executable) files, native libraries, and configuration files like the Android Manifest, which contains essential information about the app—such as its permissions and declared components.

Types of Vulnerabilities that Static Analysis Can Detect

Type	Description
Insecure Storage	Data stored in locations that other apps or users can access.
Hard-Coded Sensitive Information	Sensitive information such as passwords and API keys embedded directly into the code.
Insecure Communication	Unencrypted or poorly encrypted communication that can be intercepted and read.
Insufficient Cryptography	The use of weak or deprecated encryption algorithms.
Insecure Permissions	Requesting more permissions than necessary, giving the app access to sensitive data.

APK Extraction

The most common way to install Android applications on a device is through app stores, which handle both the download and installation processes automatically. However, for static analysis purposes, we need direct access to the app's APK file—not just a functional installation. In the following paragraphs, we'll explore three types of app stores and explain how to extract APK files for analysis, rather than simply installing the applications on a device.

Google Play Store

The majority of Android applications are distributed through the Google Play Store, the official app store developed by Google. While it allows users to install apps directly onto their devices, it does not provide access to the raw APK files needed for static analysis.

Manufacturer App Stores

Manufacturers that ship devices with customized versions of Android may not be licensed to include the Google Play Store by default. As a result, they often offer their own app stores as alternatives. Examples of such stores include:

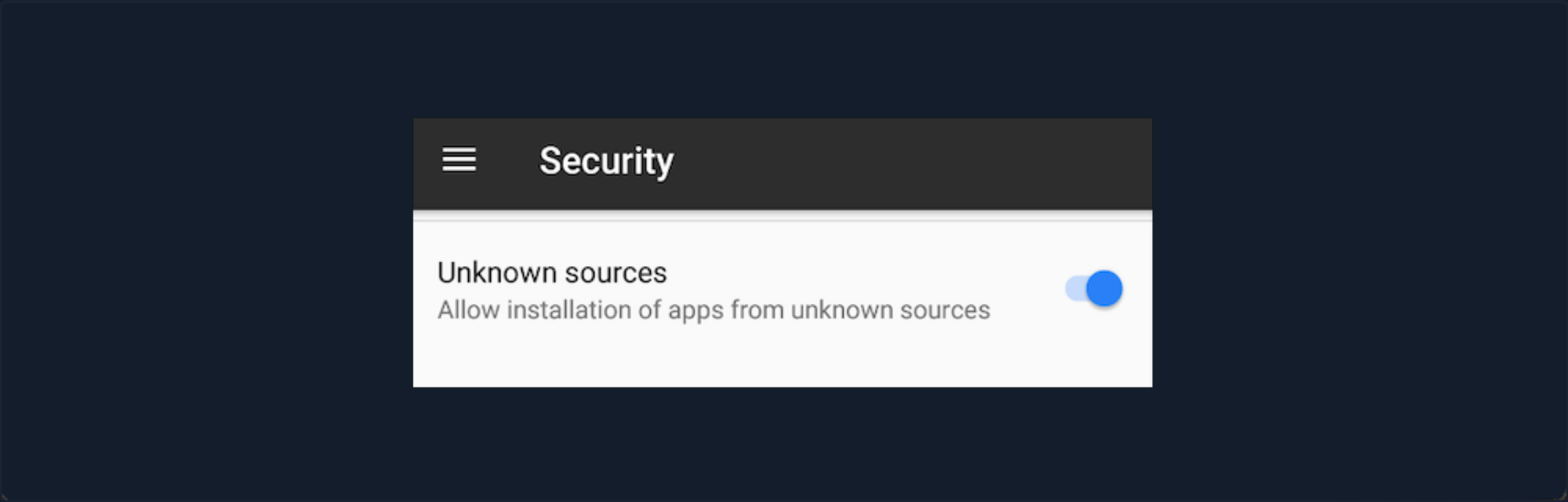
App Store
Samsung Galaxy Store
Amazon Appstore
Huawei AppGallery
Xiaomi Mi GetApps
OPPO App Market
VIVO App Store

Third-Party App Stores

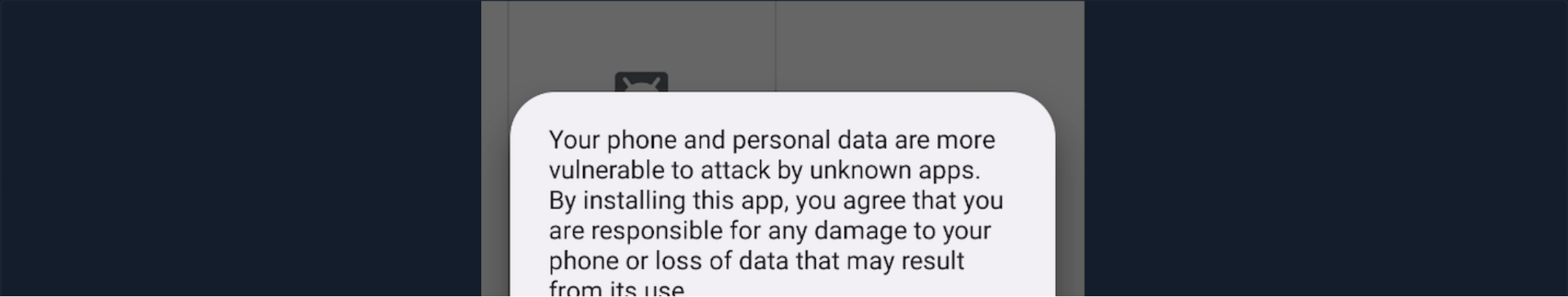
Third-party app stores serve as alternative sources for downloading Android applications and are not bundled or pre-installed on most devices. These platforms often allow users to download APK files directly, enabling manual installation and analysis. They can also provide access to older app versions or apps that are restricted or not available on the Google Play Store. Some well-known third-party app stores include:

App Store
Amazon Appstore
APKMirror
APKPure
APKCombo
Aptoide
F-Droid

Installing Apps from Third-Party App Stores requires the **Unknown Sources** option to be enabled on the device. In Android versions older than 7.1 (Nougat), this can be done by navigating to **Settings -> Security -> Unknown sources**.



In Android version 8.0 (Oreo), when we tap an APK file to install it, a dialog box appears stating that the current settings do not allow installation from this source. From the dialog, we can open the relevant security settings and toggle the switch to allow installations from that particular source.

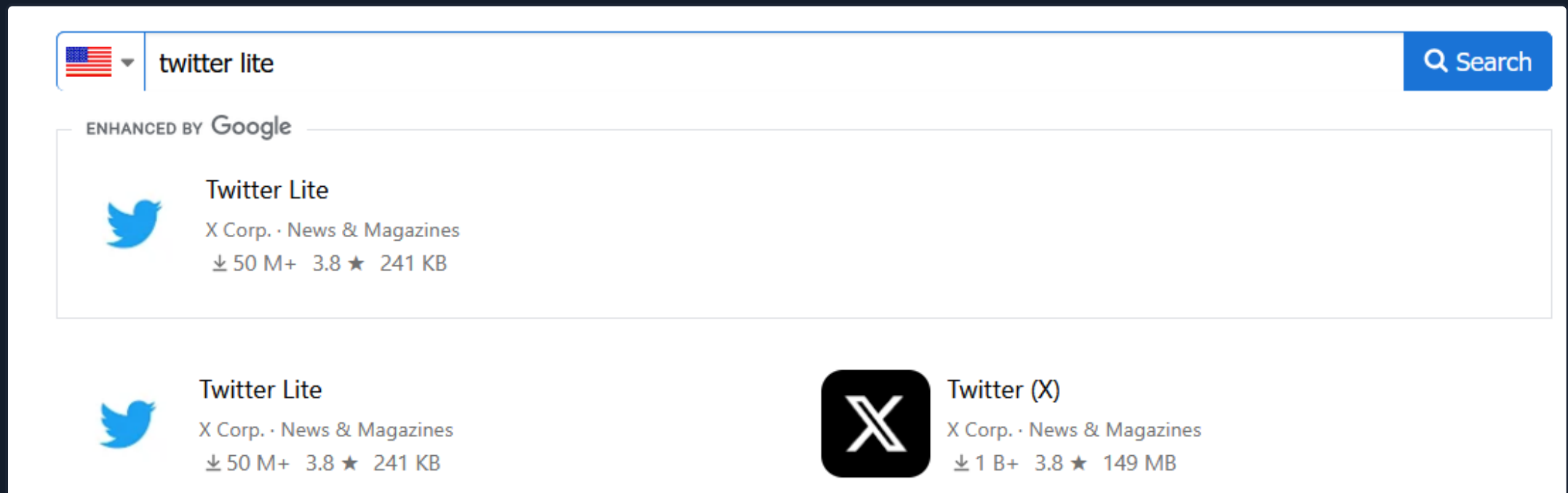


Cancel Continue

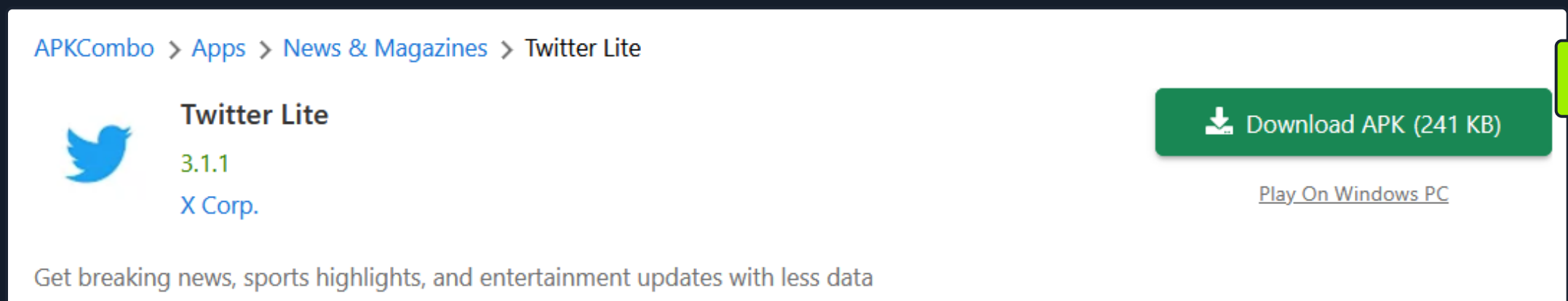
As we mentioned earlier, in order to start with application static analysis, we need to have the APK file. Let's take a look at three different ways to extract the APK file from an app.

Finding APK Online

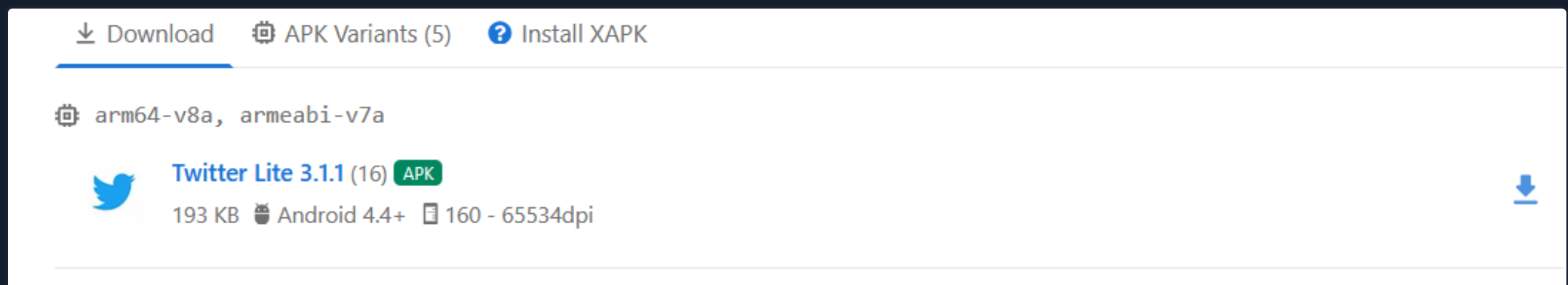
In this approach, we will find and download the APK file from Third-Party App Stores. Let's navigate to [APKCombo](#) and search for the application we want to analyze. Let's search for the Twitter Lite app.



Select the first result, and click the green **Download APK** button.



Now, click on the Twitter Lite icon to download the APK file.



That's it. The Twitter Lite APK has been downloaded successfully. If you're using the host machine to download the file, you can install it on the Android device via ADB using the following command:

```
[!bash!]$ adb install TwitterLite.apk
```

If you download the APK directly from the device (e.g., using a mobile browser), you'll find it in the **Downloads** folder via the file manager. Simply tap the file to install it.

Extracting the APK using Third-Party Tools

Another method for obtaining APK files is by using third-party tools designed for this purpose. One such tool is **APK Export**, an Android application that automatically extracts the APK file of any app installed on the device and saves it locally. APK Export can be downloaded from [APKCombo](#) and installed via ADB as demonstrated earlier.



APK Export (Backup & Share)

3.2.4

Area 51bis



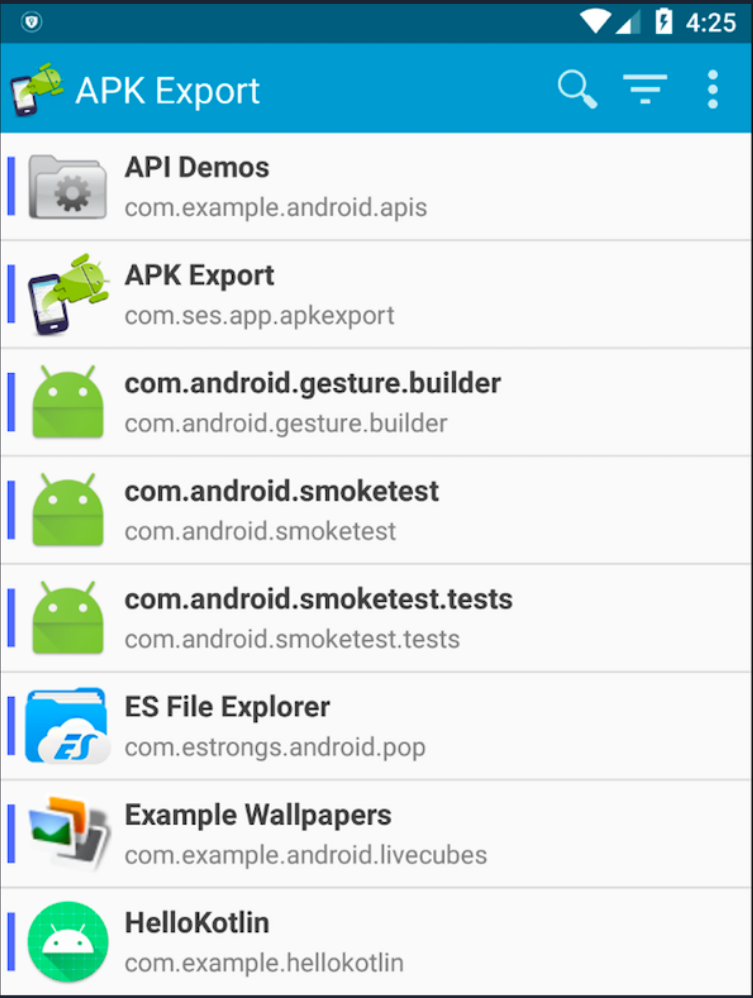
Download APK (137 KB)

[Play On Windows PC](#)

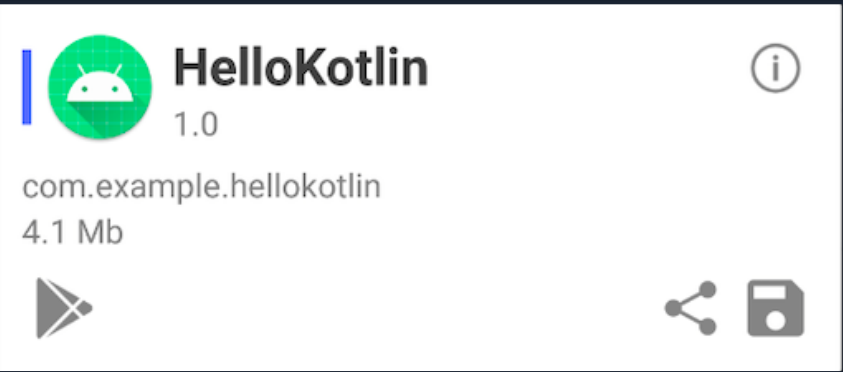
Manage and extract your apps.



Once installed, launch the app and navigate to the installed application you want to extract.




Let's try extracting the APK file of the app **HeLLoKotLin**. Tap on the app name, then tap the disk icon in the bottom-right corner to save the APK.



The exported APK will appear in the **DownLoads** folder, and you can install it by tapping on the file.

Extracting The APK From The Device

A more manual approach to APK extraction involves retrieving the APK file of an already installed app directly from the device. APKs are typically located in the directory `/data/app/<package-name>-1/base.apk`. For example, if the package name is `com.example.myapplication`, the path would likely be `/data/app/com.example.myapplication-1/base.apk`.

 **Note:** The suffix (-1) might vary, and could be a sequence number or a randomly generated string depending on the Android version.

Access to the `/data/app/` directory is restricted for non-rooted users, which makes it difficult to explore or guess full package paths. However, if you already know the package name, you can locate the exact path using ADB commands.

Before proceeding, make sure you've completed the steps in the [Android Fundamentals](#) module. If you're using an **Android Virtual Device (AVD)**, refer to the [Android Emulators](#) section for guidance.


```
[!bash!]$ adb shell pm path com.example.myapplication

package:/data/app/com.example.myapplication-1/base.apk
```

If you don't know the exact package name of the app, the following command will usually help, as the app name is often part of the package name:

```
[!bash!]$ adb shell pm list packages | grep myapp


com.example.myapplication
```

 **Note:** If you're using a Windows command shell, remember to wrap the entire pipeline in quotes.

Once you've identified the correct package, you can pull the `base.apk` file from the device to your local machine using the following command:

```
[!bash!]$ adb pull /data/app/com.example.myapplication-1/base.apk .

/data/app/com.example.myapplication-1/base.apk: 1 file pulled, 0 skipped. 27.3 MB/s (113799083 bytes in 3.969s)
```



Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location


UK

39ms

▼

Terminate Pwnbox to switch location

Start Instance

 / 1 spawns left

Enable step-by-step solutions for all questions ⓘ ✨

Questions

Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

Target(s): [Click here to spawn the target system!](#)

+ 3

Download the "myapp_intro.zip" file and install the application on an Android emulator. Launch the app, configure the IP and PORT settings, and tap `Install App`. If prompted, grant any necessary installation permissions. Then, use Android Debug Bridge (ADB) to extract the newly installed APK named "extractme". What is the sha256 hash of the APK?

Submit your answer here...

+10 Streak pts

Submit

myapp_intro.zip

Hint

Next →

Cheat Sheet

Go to Questions

Table of Contents

Extracting and Enumerating APK Files

- [Introduction](#)
- Disassembling the APK
- Understanding Smali

Analyzing Application's Source Code

- Reading Hardcoded Strings
- Bad Cryptography Implementation
- Reversing Hybrid Apps
- Reading Obfuscated Code
- Deobfuscating Code

Analyzing Native Libraries

- Reversing Shared Objects
- Reversing DLL Files

Application Patching

- Authentication Bypass
- Modifying Game Apps
- License Verification Bypass
- Root Detection Bypass

Skills Assessment

- Skills Assessment

OFFLINE

▶

Start Instance

∞ / 1 spawns left

