Bypassing Detection Mechanisms

Another use of dynamic analysis is bypassing security mechanisms that block debugging or tampering. This section focuses on bypassing root detection, Frida detection, and biometric authentication using Frida. Root detection, which we discussed in the Android Application Static Analysis module, is a common technique used to prevent applications from running on rooted devices. By hooking into and modifying the app's root detection methods with Frida, we can alter its behavior at runtime and allow it to run on rooted devices. The same approach applies to Frida detection, which is often implemented to block instrumentation and code analysis.

This section will also cover bypassing biometric authentication, a mechanism that provides a high level of security by verifying a user's identity through physical characteristics (e.g., Face ID, fingerprint). In the following examples, we'll demonstrate how to bypass biometric checks using dynamic code instrumentation with Frida.

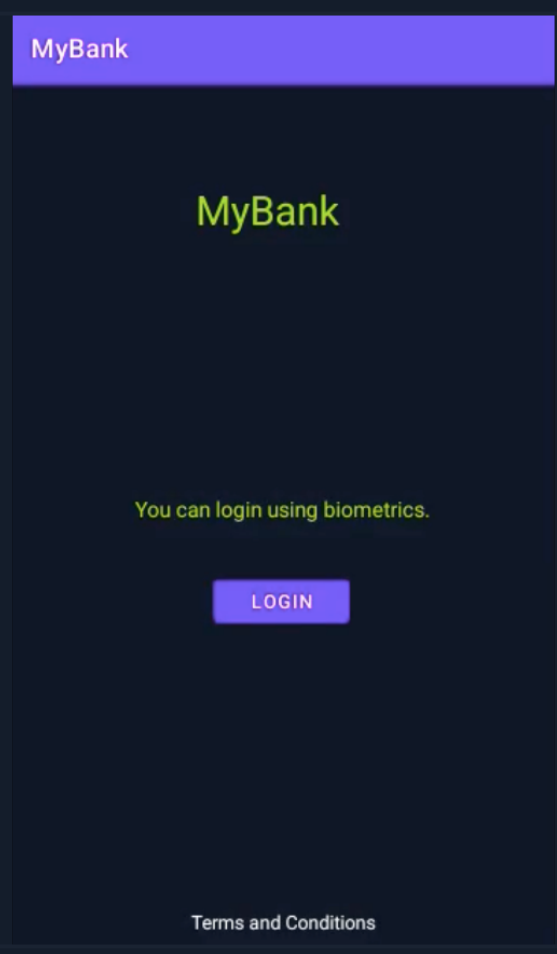
To follow along, you'll need an Android Virtual Device (AVD) with the `Google APIs` system image. If you're using a physical or different emulated device, ensure that it is rooted. Let's start by connecting to the device via ADB and installing the application.

Bypassing Detection Mechanisms

```
rl1k@htb[/htb]$ adb connect
rl1k@htb[/htb]$ adb install myapp.apk

Performing Streamed Install
Success
```

Running the application, we see a banking app asking the user to log in using biometrics.

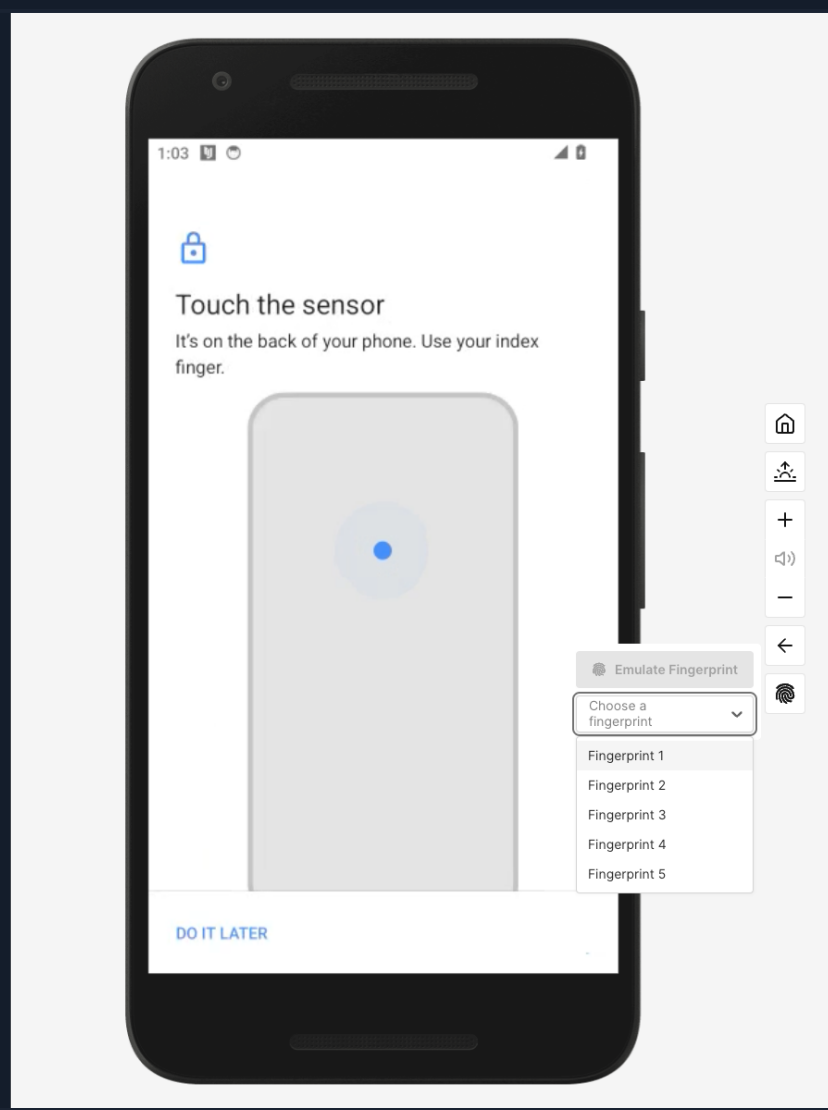


To enable and configure fingerprint authentication on the device, follow these steps:

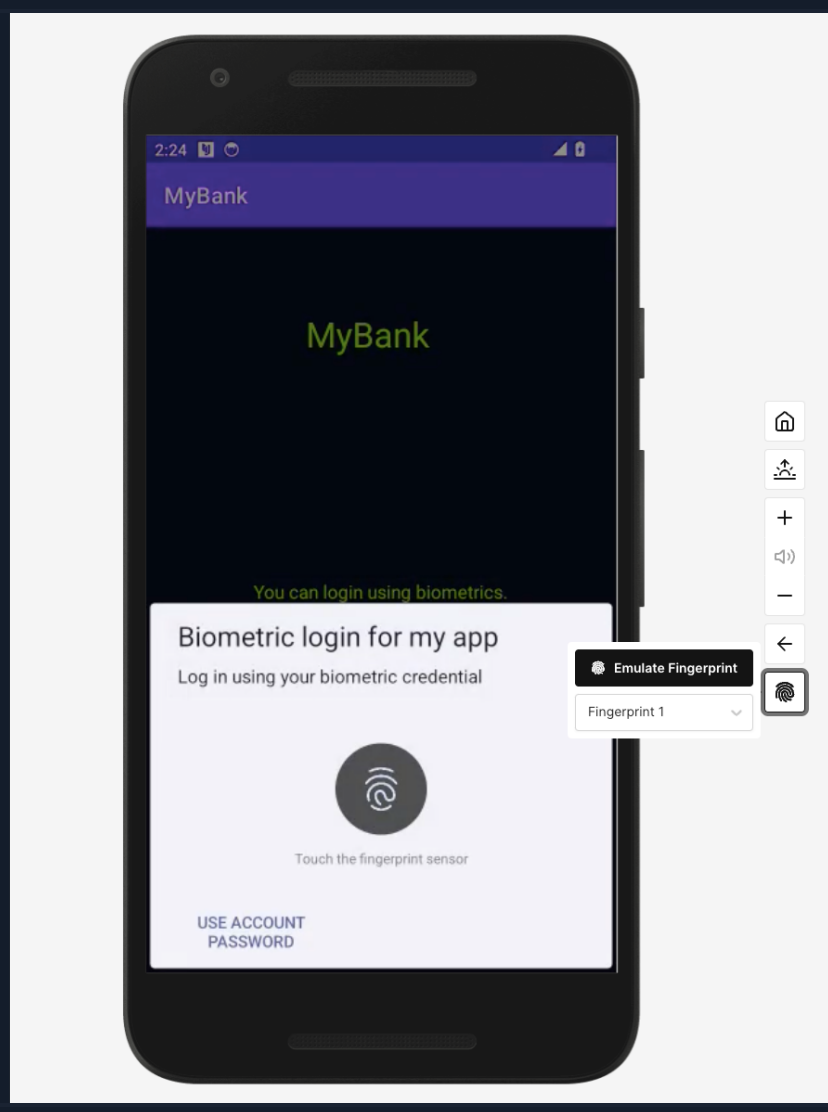
1. Open the device and navigate to:
   `Settings` → `Security & privacy` → `Device unlock` → `Screen lock`.
2. Choose the `PIN` option and enter `1234` as the PIN. Tap `Enter`.
3. On the following screens:
   - Re-enter the same PIN (`1234`).
   - Tap `CONFIRM`, then `DONE`.
4. Return to the `Device unlock` screen and select `Face & Fingerprint`.
5. When prompted, enter the PIN (`1234`) again and tap `Enter`.

6. Scroll to the bottom of the page and tap `I AGREE` to accept the terms.
7. On the right side of the emulator window, click the fingerprint icon in the vertical toolbar.
8. In the drop-down menu that appears, select `Fingerprint 1`.
9. Click `Emulate Fingerprint` multiple times until you see the message `Fingerprint added`.
10. Once the fingerprint is successfully added, tap `DONE`.

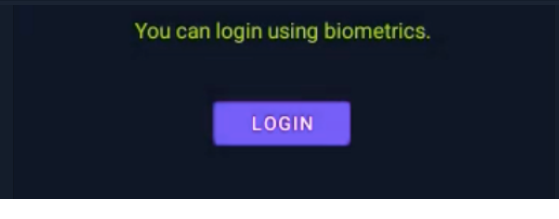
The device is now configured for fingerprint authentication.

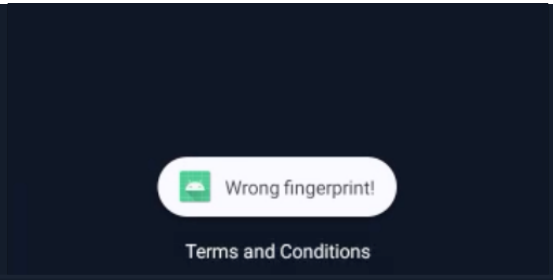


Once fingerprint authentication is enabled, try to log in to the application. Open the app, click on the fingerprint icon from the vertical menu on the righ of the device once again, and then click `Emulate Fingerprint`.



Unfortunately, the message `Wrong fingerprint!` is displayed.





Let's use JADX to read the application's source code.

Bypassing Detection Mechanisms

```
rl1k@htb[/htb]$ jadx-gui myapp.apk
```

```
com
  google
  hackthebox.myapp
    databinding
    HomeActivity
    MainActivity
    R
  scottyab.rootbeer
kotlin
kotlinx.coroutines
org
Resources
APK signature
Summary
```

```
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app.ComponentActivity, android
43  public void onCreate(Bundle savedInstanceState) {
44      super.onCreate(savedInstanceState);
45      setContentView(R.layout.activity_main);
47      this.btn_login = (Button) findViewById(R.id.btn_login);
48      tv_output = (TextView) findViewById(R.id.tv_output);
49      this.tv_title1 = (TextView) findViewById(R.id.tv_title1);
51      if (!isAppSignatureValid()) {
54          finish();
            return;
        }
59      this.executor = ContextCompat.getMainExecutor(this);
60      this.biometricPrompt = new BiometricPrompt(this, this.executor, new BiometricPrompt.AuthenticationCallback() {
// from class: com.hackthebox.myapp.MainActivity.1
            @Override // androidx.biometric.BiometricPrompt.AuthenticationCallback
63          public void onAuthenticationError(int errorCode, CharSequence errString) {
64              super.onAuthenticationError(errorCode, errString);
65              Toast.makeText(MainActivity.this.getApplicationContext(), "Authentication error: " + ((Object) errString), 0).show();
            }

            @Override // androidx.biometric.BiometricPrompt.AuthenticationCallback
70          public void onAuthenticationSucceeded(BiometricPrompt.AuthenticationResult result) {
75              if (System.currentTimeMillis() - MainActivity.this.startTime > Integer.parseInt(MainActivity.this.n88b11())) {
76                  Toast.makeText(MainActivity.this.getApplicationContext(), "Wrong fingerprint!", 1).show();
                    return;
                }
78              super.onAuthenticationSucceeded(result);
79              Toast.makeText(MainActivity.this.getApplicationContext(), "Authentication succeeded!", 0).show();
81              Intent intent = new Intent(MainActivity.this, HomeActivity.class);
82              intent.putExtra("SECRET_KEY", MainActivity.this.h79j31());
83              MainActivity.this.startActivity(intent);
            }

            @Override // androidx.biometric.BiometricPrompt.AuthenticationCallback
88          public void onAuthenticationFailed() {
89              super.onAuthenticationFailed();
90              Toast.makeText(MainActivity.this.getApplicationContext(), "Authentication failed", 0).show();
            }
        });
```

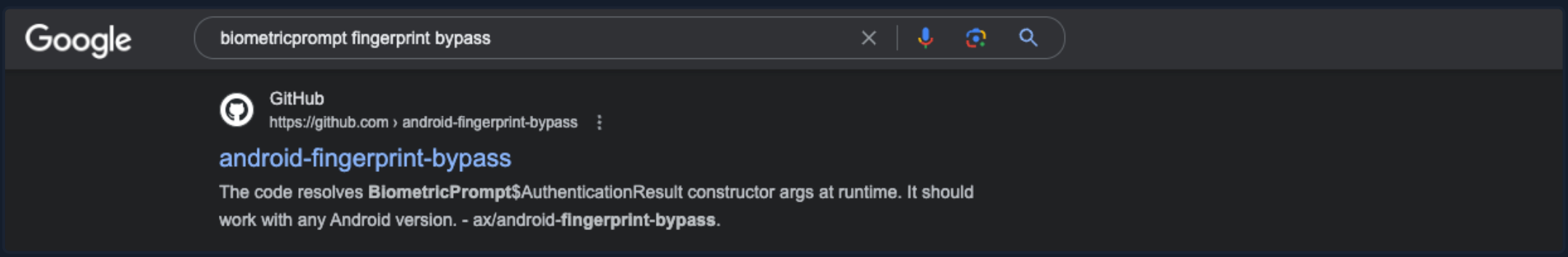
The `MainActivity` class shown above reveals that the app uses the `BiometricPrompt` library for its fingerprint authentication. This is confirmed by the imported package at the top of the class: `import androidx.biometric.BiometricPrompt`.

```
android.support.v4
androidx
com
  google
  hackthebox.myapp
    databinding
    HomeActivity
    MainActivity
    R
  scottyab.rootbeer
```

```
package com.hackthebox.myapp;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.biometric.BiometricPrompt;
import androidx.core.content.ContextCompat;
import java.util.concurrent.Executor;
```

The snippet also includes the statement `if (!isAppSignatureValid())`, which suggests that the application implements anti-patching signature verification. A quick search for `biometricprompt fingerprint bypass` returns the following as the second result.



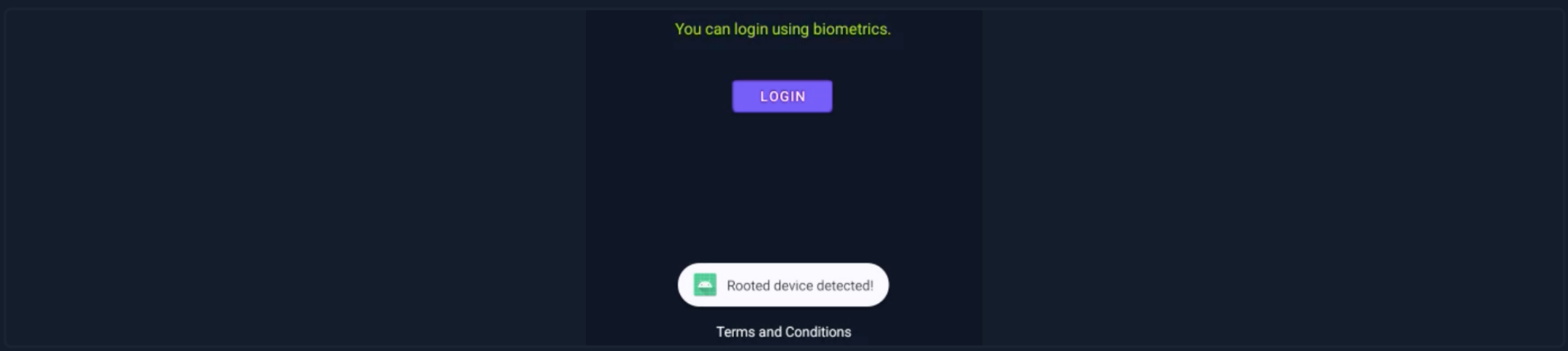
Let's download the script, run it using Frida, and tap `LOGIN` once the app is launches.

Bypassing Detection Mechanisms

```
rl1k@htb[/htb]$ wget https://raw.githubusercontent.com/ax/android-fingerprint-bypass/main/fingerprint-bypass.js
rl1k@htb[/htb]$ frida -U -l fingerprint-bypass.js  -f com.hackthebox.myapp


     ____
    / _  |   Frida 16.1.11 - A world-class dynamic instrumentation toolkit
```

```
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Android Emulator (id=emulator-5554)
Spawned `com.hackthebox.myapp`. Resuming main thread!
[Android Emulator::com.hackthebox.myapp ]-> Hooking BiometricPrompt.authenticate()...
Hooking BiometricPrompt.authenticate2()...
Hooking FingerprintManagerCompat.authenticate()...
Hooking FingerprintManager.authenticate()...
[BiometricPrompt.BiometricPrompt()]: cancellationSignal: android.os.CancellationSignal@8d06f3a, executor: , callback: android
[*] Overload number ind: 0
cryptoInst:, android.hardware.biometrics.BiometricPrompt$CryptoObject@af69948 class: android.hardware.biometrics.BiometricPro
[BiometricPrompt.BiometricPrompt()]: callback.onAuthenticationSucceeded(NULL) called!
```



The message `Authentication succeeded!` is displayed, followed by the unfortunate the message `Rooted device detected`. Notice that although the login mechanism is bypassed, the login screen is still being displayed. This is likely due to the app detecting a rooted device. Examination of the `onCreate()` method in the `MainActivtiy` class (shown in the screenshot above) reveals that the user will be transferred to the `HomeActivity` on successful login. Reading its source code in JADX, we see the following.

```
com
  google
  hackthebox.myapp
    databinding
    HomeActivity
    MainActivity
    R
  scottyab.rootbeer
kotlin
kotlinx.coroutines
org
Resources
APK signature
Summary
```

```
    /* JADX INFO: Access modifiers changed from: protected */
    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app.ComponentActivity, android
37  public void onCreate(Bundle savedInstanceState) {
38      super.onCreate(savedInstanceState);
39      setContentView(R.layout.activity_home);
41      this.textViewBalance = (TextView) findViewById(R.id.textViewBalance);
43      if (!isAppSignatureValid()) {
46          finish();
            return;
        }
50      setColors();
53      String stringExtra = getIntent().getStringExtra("SECRET_KEY");
54      if (stringExtra == null || !stringExtra.equals(v66f24())) {
55          Toast.makeText(getApplicationContext(), "Unauthorized attempt to start the App!", 0).show();
56          finish();
61      } else if (this.rootBeer.isRooted()) {
62          Toast.makeText(this, "Rooted device detected!", 1).show();
63          finish();
66      } else if (isFridaDetected()) {
67          Toast.makeText(this, "Frida detected!", 1).show();
68          finish();
        }
    }
```

There are two additional checks, along with the anti-patching signature check "`if (!isAppSignatureValid())`". The first checks if the device is rooted, and the second checks if the Frida tool is used. Ironically, we can bypass both of these security checks using Frida. Let's open the `fingerprint-bypass.js` file that we downloaded earlier and use it to bypass the login mechanism. Add the following snippet at the end of the file:

Code: js

```js
// Hook RootBeer's isRooted method
function rootBypass() {
    var RootBeer = Java.use('com.scottyab.rootbeer.RootBeer');
    RootBeer.isRooted.overload().implementation = function () {
        console.log('\nRoot detection bypassed');
        return false; // Always return false to indicate the device is not rooted
    };
}
```

This script overrides the `isRooted` method of the `RootBeer` class, forcing it to always return `false` and effectively bypassing the root detection mechanism. Next, we add the function call `rootBypass();` within the `Java.perform(function () {...})` block to ensure it executes alongside the

mechanism. Next, we add the function call `rootBypass();` within the `Java.perform(function () { ... })` block to ensure it executes alongside the other hooking functions. After these changes, the code snippet within `Java.perform(function () { ... })` should appear like so:

Code: js

```
Java.perform(function () {
    //Call in try catch as Biometric prompt is supported since api 28 (Android 9)
    try { hookBiometricPrompt_authenticate(); }
    catch (error) { console.log("hookBiometricPrompt_authenticate not supported on this android version") }
    try { hookBiometricPrompt_authenticate2(); }
    catch (error) { console.log("hookBiometricPrompt_authenticate not supported on this android version") }
    try { hookFingerprintManagerCompat_authenticate(); }
    catch (error) { console.log("hookFingerprintManagerCompat_authenticate failed"); }
    try { hookFingerprintManager_authenticate(); }
    catch (error) { console.log("hookFingerprintManager_authenticate failed"); }

    rootBypass();
});
```

Save the changes, run the Frida script again, and tap the `LOGIN` button.

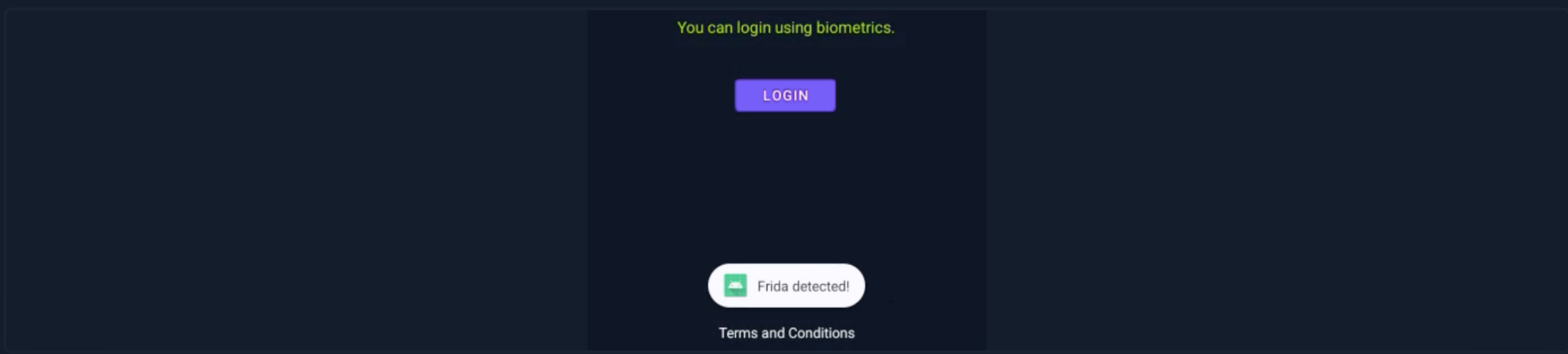
Bypassing Detection Mechanisms

```
rl1k@htb[/htb]$ frida -U -l fingerprint-bypass.js  -f com.hackthebox.myapp                                  2m 3s

<SNIPT>
[*] Overload number ind: 0
cryptoInst:, android.hardware.biometrics.BiometricPrompt$CryptoObject@a14a35c class: android.hardware.biometrics.BiometricPro
[BiometricPrompt.BiometricPrompt()]: callback.onAuthenticationSucceeded(NULL) called!

Root detection bypassed
```

Ths time, the message `Root detection bypassed` is displayed in Frida's output, but the application's message hasnow changed to `Frida detected!`.



Let's add one more snippet of Javascript at the end of the `fingerprint-bypass.js` file to bypass the Frida detection mechanism.

Code: js

```
// Hook the isFridaDetected native method
function fridaBypass() {
    var MainActivity = Java.use('com.hackthebox.myapp.HomeActivity');
    MainActivity.isFridaDetected.implementation = function () {
        console.log('Frida detection bypassed');
        return false; // Always return false to indicate Frida is not detected
    };
}
```

Along with the correpsonding function call `fridaBypass();`. Place it within the `Java.perform(function () { ... })` block to ensure it gets executed alongside the other hooking functions. After the changes, the code snippet within `Java.perform(function () {...})` should look like this.

Code: js

```
Java.perform(function () {
    //Call in try catch as Biometric prompt is supported since api 28 (Android 9)
    try { hookBiometricPrompt_authenticate(); }
    catch (error) { console.log("hookBiometricPrompt_authenticate not supported on this android version") }
    try { hookBiometricPrompt_authenticate2(); }
    catch (error) { console.log("hookBiometricPrompt_authenticate not supported on this android version") }
    try { hookFingerprintManagerCompat_authenticate(); }
    catch (error) { console.log("hookFingerprintManagerCompat_authenticate failed"); }
    try { hookFingerprintManager_authenticate(); }
    catch (error) { console.log("hookFingerprintManager_authenticate failed"); }

    rootBypass();
    fridaBypass();
});
```

Finally, we run the script again and tap the `LOGIN` button.

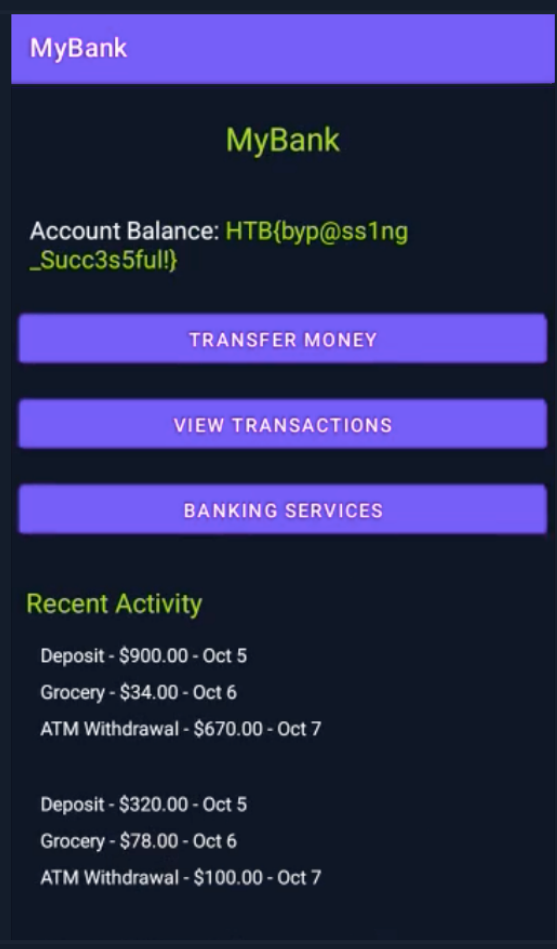
Bypassing Detection Mechanisms

```
rl1k@htb[/htb]$ frida -U -l fingerprint-bypass.js  -f com.hackthebox.myapp                                11m 22s

<SNIPT>
[*] Overload number ind: 0
cryptoInst:, android.hardware.biometrics.BiometricPrompt$CryptoObject@8d06f3a class: android.hardware.biometrics.BiometricPro
[BiometricPrompt.BiometricPrompt()]: callback.onAuthenticationSucceeded(NULL) called!

Root detection bypassed
Frida detection bypassed
```

This time, the message `Frida detection bypassed` is also displayed in Frida's output, indicating that this security mechanism is successfully bypassed. Looking at the application, we can also see that the login is successful, and we are provided with a user overview screen.



Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Cheat Sheet

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

+ 5 Log in to the MyBank app. What is displayed as the Account Balance?

Submit your answer here...

+10 Streak pts Submit detection_bypass.zip

Previous Next

Cheat Sheet

Go to Questions

Table of Contents

- Hooking Java Methods
- Altering Method Values
- Hooking Native Methods
- Bypassing Detection Mechanisms
- Authentication Token Manipulation

Intercepting HTTP/HTTPS Requests

- Intercepting API Calls
- IDOR Attack
- SSL/TLS Certificate Pinning Bypass

Skills Assessments

- Skills Assessment

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left