

WEBINAR

Mobile Pentesting

Presented by rootbakar





ABOUT ME

R. Talaohu a.k.a RootBakar

Pentester

Bug Hunter

Top 5 Bug Hunter on CyberArmyID

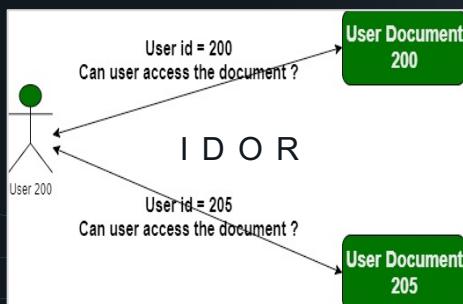
@rootbakar (Telegram)

ABOUT YOU

MATERI



Technique Pentest Web Application & Mobile



“ Pentest (Penetration Testing) adalah serangkaian proses berisi prosedur dan teknik mengevaluasi keamanan terhadap sistem atau jaringan dengan melakukan simulasi penyerangan untuk mengetahui letak celah-celah kerawanan pada sistem agar kemudian celah tersebut ditutup/diperbaiki.

Penetration Testing dilakukan sebagai langkah preventive untuk mengatasi terjadinya peretasan pada suatu sistem.

SECARA UMUM PENETRATION TESTING TERDIRI DARI 4 TAHAP:



MOBILE PENTESTING

Introduction



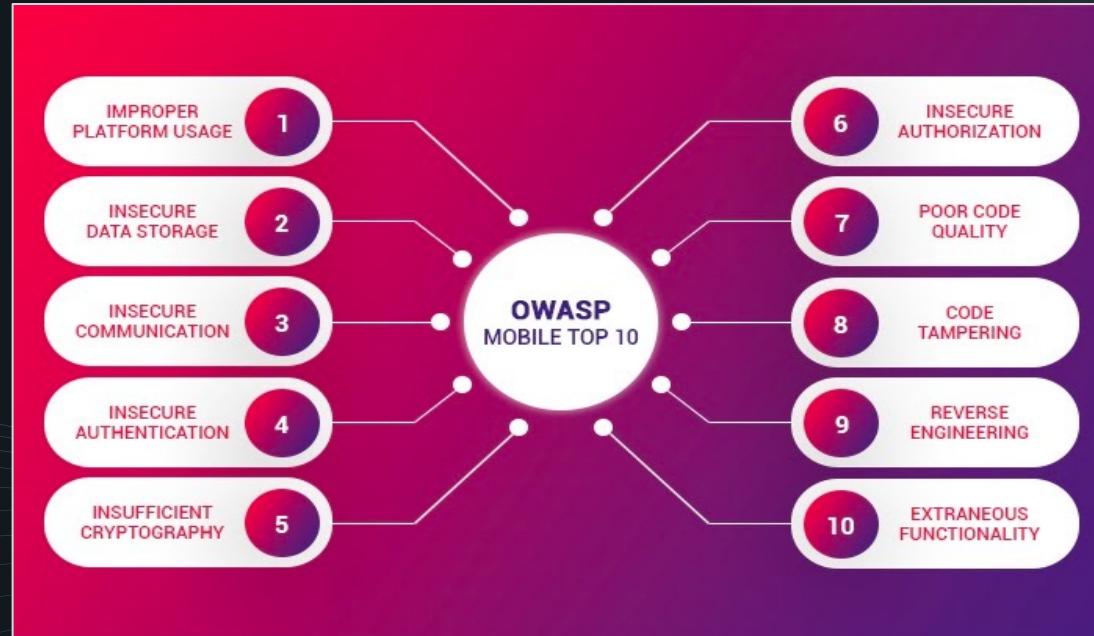
Aplikasi mobile banyak populer di kalangan perusahaan startup. mereka berlomba-lomba membuat produk yang mudah digunakan untuk konsumen dengan mobilitas tinggi. Sehingga muncul ancaman dan peluang baru di dunia Cyber Security mengenai keamanan dan ancamannya di aplikasi mobile. Sehingga muncul security framework baru untuk sebagai best practice dalam pengujian keamanan untuk hal tersebut.

Mobile App Penetration testing framework bisa menggunakan OWASP Mobile App Security Testing Guide.

<https://github.com/OWASP/owasp-mstg/releases>

Introduction

Untuk materi training Mobile Pentesting akan mengacu pada OWASP Top 10 Mobile Risks OWASP 2016



<https://owasp.org/www-project-mobile-top-10/>

Introduction

Sebagai seorang **Defensive Security Engineer** kalian akan banyak terlibat dalam implementasi standar dan framework tertentu.

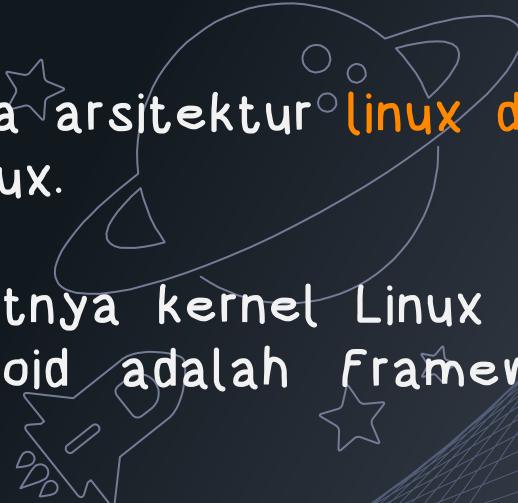
Pedoman yang diakui dan sudah diuji ini tentu akan menjadikan sistem perusahaan memiliki pondasi yang kuat terhadap kelayakan dalam pengadaan sistem elektronik dan digital.

Tentu dengan penerapan standar, akan menjadi pedoman yang diakui oleh banyak pihak karena standar yang kita pilih telah diuji oleh banyak peneliti keamanan dan terus dikembangkan oleh banyak profesional dibidang tersebut.

Architecture Android

Banyak orang beranggapan bahwa arsitektur linux dan android sama meski memiliki basis OS Linux.

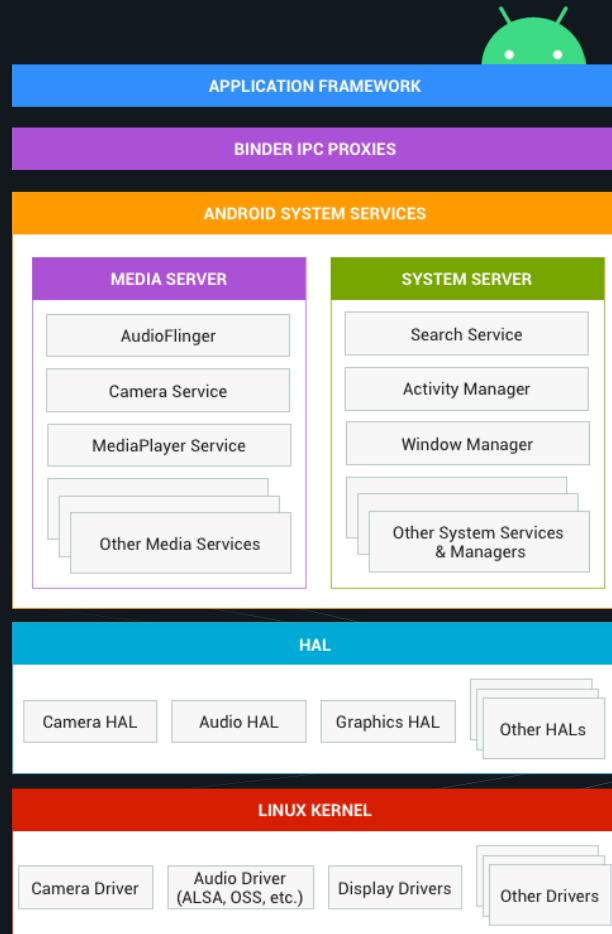
OS Linux. penjelasan lebih tepatnya kernel Linux adalah OS paling populer sementara Android adalah framework yang dibangun di atas kernel Linux.



Jadi setiap perangkat Android menjalankan kernel Linux juga. tetapi setiap perangkat Linux tidak memiliki Android.

Kita dapat menganggap kernel Linux sebagai dasar dimana Android dibangun.

Architecture Android

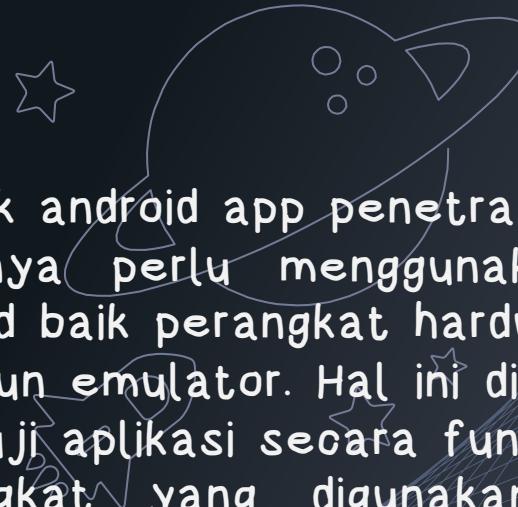


Seperti yang kalian lihat pada gambar disamping bahwa aplikasi android terletak pada bagian teratas sedangkan pemanggilan API tertentu dijembatani oleh **Android Framework**. Kemudian Android Framework akan menghubungkan antara Aplikasi dengan Hal dan Linux Kernel melalui **Android Runtime**.

Android Emulator

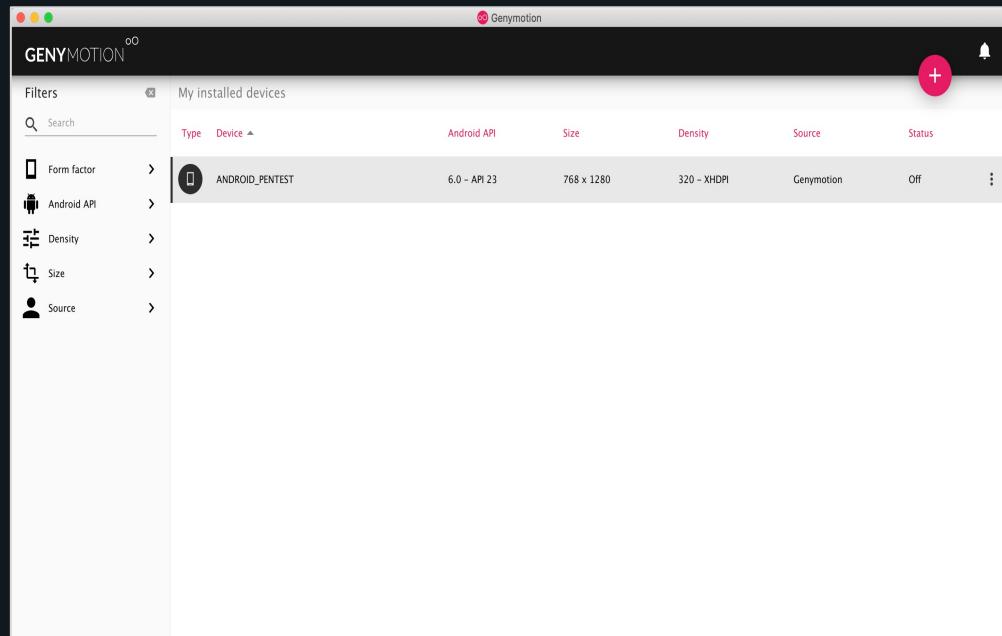
00
GENYMOTION
By Genymobile

<https://www.genymotion.com/download/>

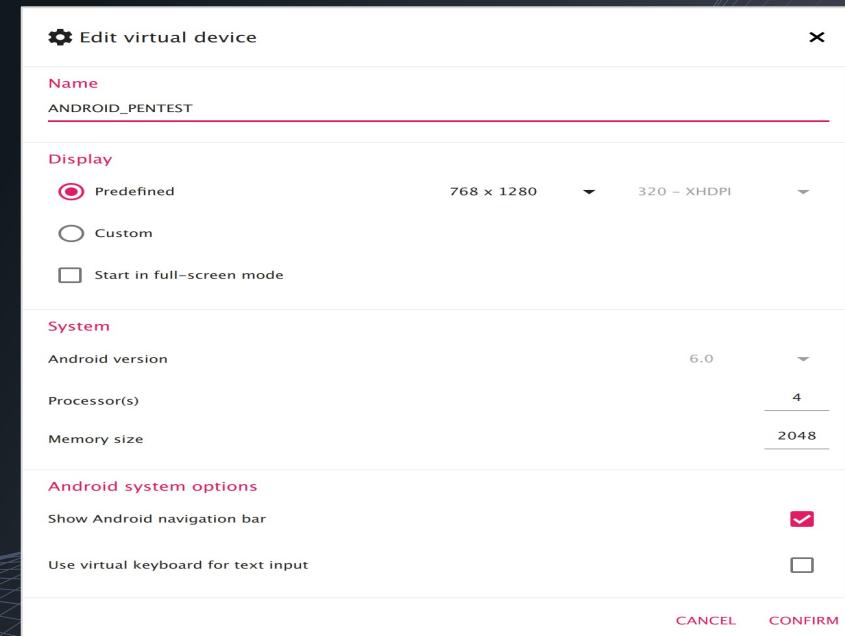


Proyek android app penetration testing biasanya perlu menggunakan perangkat android baik perangkat hardware asli ataupun emulator. Hal ini diperlukan untuk menguji aplikasi secara fungsional. Perangkat yang digunakan juga harus dalam kondisi memiliki akses root. Inilah kenapa pemilihan Android Emulator yang pas adalah Genymotion

Genymotion



Tampilan Genymotion

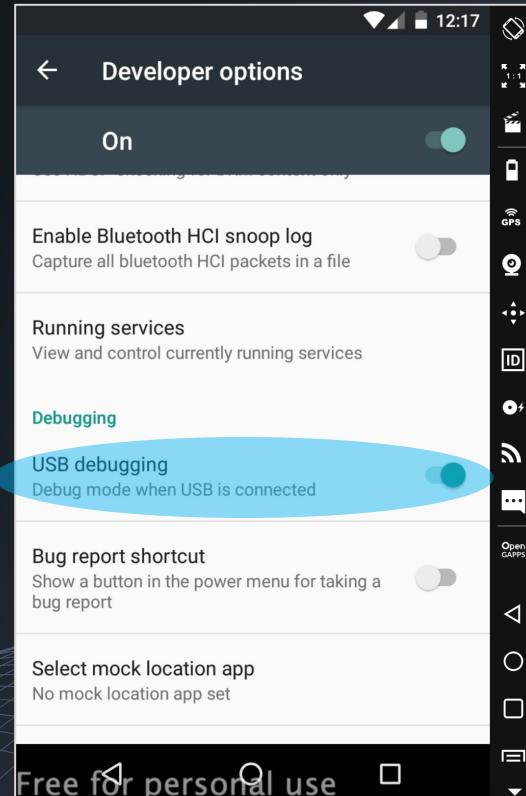


Membuat Virtual Device

Genymotion

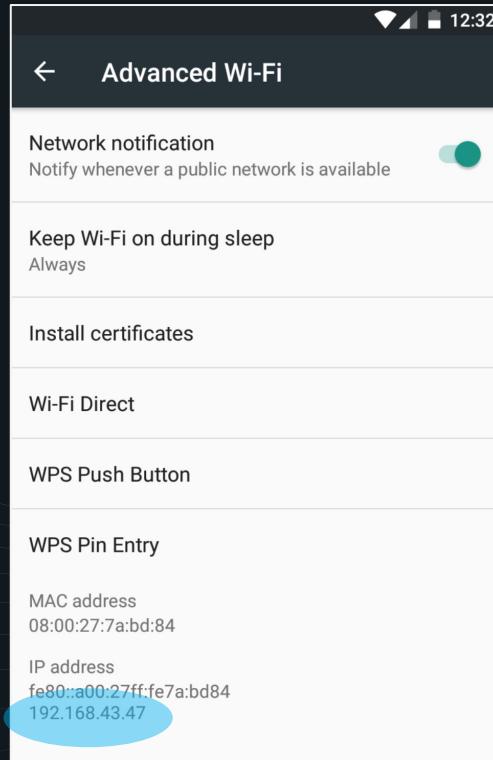


Tampilan Virtual Device



Enable USB Debugging

Genymotion



Virtual Device IP Address

```
root@ubuntu-server:~# adb devices
List of devices attached
* daemon not running; starting now at tcp:5037
* daemon started successfully
```

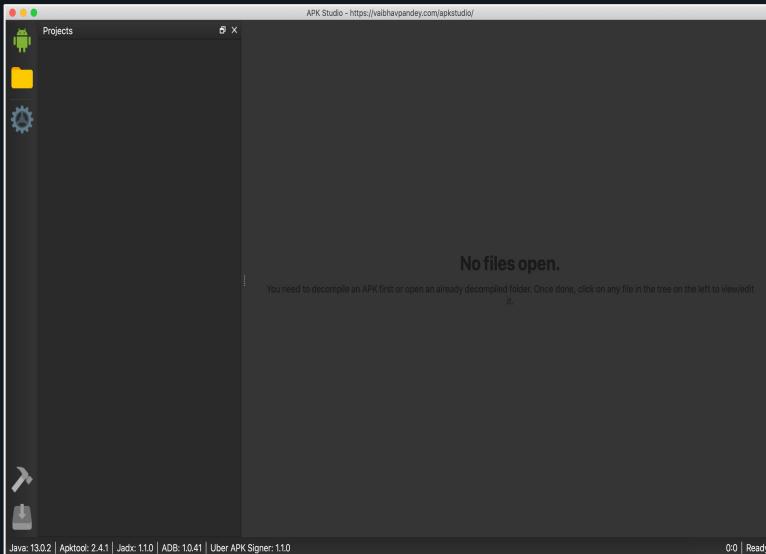
```
root@ubuntu-server:~# adb devices
List of devices attached
```

```
root@ubuntu-server:~# adb connect 192.168.43.47
connected to 192.168.43.47:5555
root@ubuntu-server:~# adb devices
List of devices attached
192.168.43.47:5555      device
```

```
root@ubuntu-server:~# adb shell
root@vbox86p:/ # whoami
root
```

```
root@ubuntu-server:~# adb shell getprop ro.product.cpu.abi
x86
```

APK Studio



<https://github.com/vaibhavpandeyvpz/apkstudio/releases>

<https://github.com/skylot/jadx>

<https://github.com/patrickfav/uber-apk-signer/releases>

Aplikasi ini akan membantu kita untuk merekayasa balik aplikasi yang istilah kerennya reverse engineering. Ini dapat melakukan decompile APK ke bentuk yang hampir asli dan membangunnya kembali setelah melakukan beberapa modifikasi. Istilah kerennya adalah Cracking

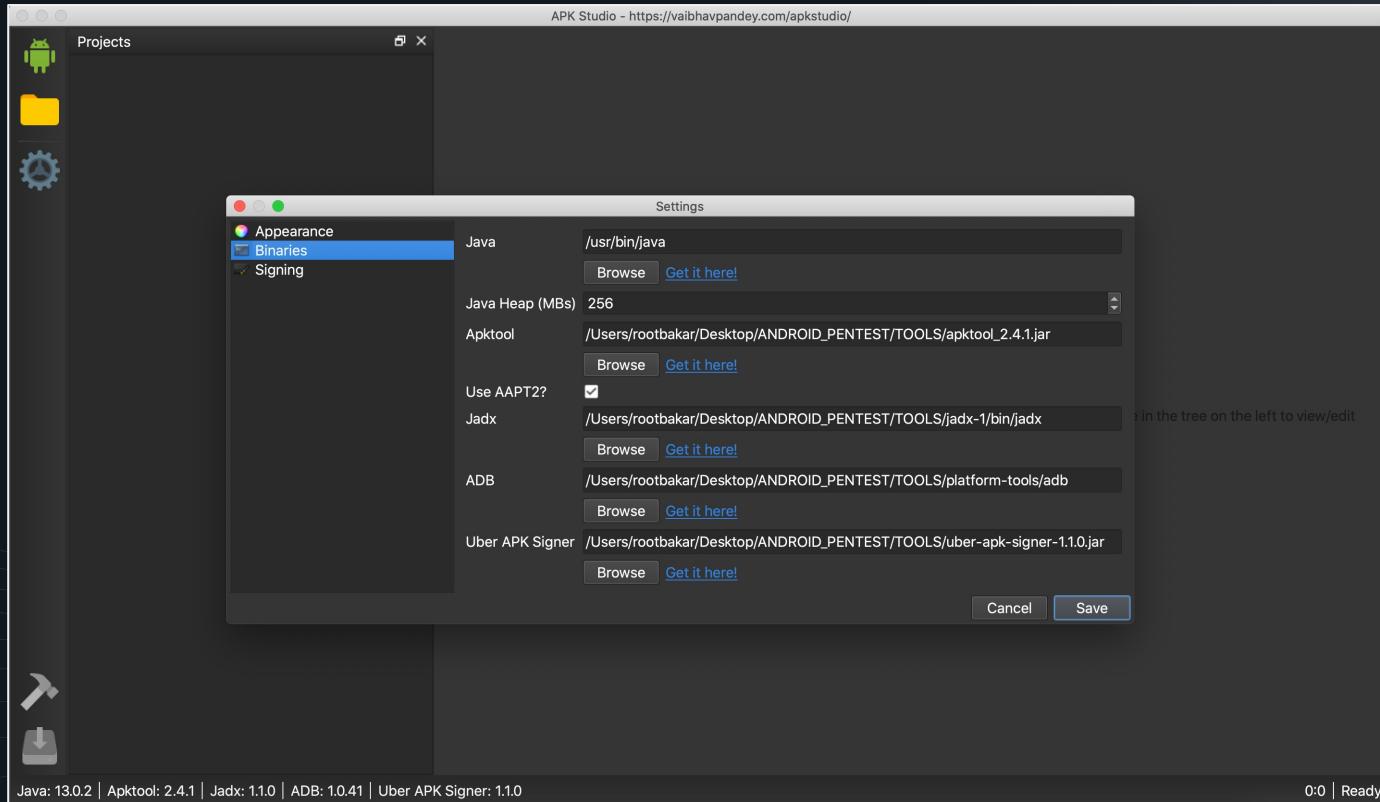
Jika teman-teman menggunakan windows pasti pernah menggunakan istilah cracking pada aplikasi yang berbayar sehingga bisa dijalankan secara gratis. Nah, seperti itulah cara kerjanya

Dalam proyek mobile app penetration testing, teknik ini bisa kita gunakan untuk melakukan analisis statis secara Black Box.

Lebih jelasnya, meski kita menggunakan Black Box kita bisa melakukan decompile APK agar kita bisa merubahnya ke bentuk source code kembali. Dari situ kita akan mudah dalam mengidentifikasi kerentanan dengan membaca source code tersebut.

Sebelum kita membahas lebih dalam mempelajari teknik reverse engineering. Teman-teman perlu menyiapkan terlebih dahulu lab untuk pengujian. Pertama, kalian harus mempersiapkan APK Studio

APK Studio



Binaries Settings on APK Studio

Frida



Frida merupakan aplikasi yang bekerja dalam membantu kita melakukan analisis secara dinamis. Tool ini akan menginjeksi aplikasi dengan intruksi atau script dengan dukungan banyak bahasa pemrograman seperti Python, NodeJS, dan lain-lain.

Frida adalah alat instrumentasi biner dinamis (*Dynamic Binary Instrumentation*). Mungkin sedikit gambaran, pada tahap melakukan penetration testing, kita mengenal 2 tahapan testing yaitu **Static Analysis** dan **Dynamic Analysis**. Tools Frida ini biasa digunakan ketika kita melakukan **Dynamic Analysis**.

Frida

Frida Server merupakan tool agent yang perlu dijalankan di device yang menjadi target of evaluation kita. Jika kita ingin melakukan mobile app penetration testing maka frida server harus dijalankan di perangkat android tentunya dengan permission root.

Karena emulator android yang kita gunakan menggunakan arsitektur i686 atau mudahnya menggunakan prosessor intel / amd maka kita gunakan frida-server yang sesuai.

```
root@ubuntu-server:~# wget https://github.com/frida/frida/releases/download/12.11.4/frida-server-12.11.4-android-x86.xz
--2020-07-25 14:59:06--  https://github.com/frida/frida/releases/download/12.11.4/frida-server-12.11.4-android-x86.xz
Resolving github.com (github.com)... 13.229.188.59
Connecting to github.com (github.com)|13.229.188.59|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/9405122/c068c600-ce32-11ea-9152-e4110f8c1272?
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200725%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200725T145906Z&X-Amz-Expires=3000&X-Amz-Signature=d5045fd6653c8f188592acd5b4aa962cb56eba50c31bc9cb928f9c7edf5e5a9&X-Amz-SignedHeaders=host&actor_id=0&repo_id=9405122&response-content-disposition=attachment%3B%20filename%3Dfrida-server-12.11.4-android-x86.xz&response-content-type=application%2Foctet-stream [following]
--2020-07-25 14:59:07--  https://github-production-release-asset-2e65be.s3.amazonaws.com/9405122/c068c600-ce32-11ea-9152-e4110f8c1272?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200725%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200725T145906Z&X-Amz-Expires=3000&X-Amz-Signature=d5045fd6653c8f188592acd5b4aa962cb56eba50c31bc9cb928f9c7edf5e5a9&X-Amz-SignedHeaders=host&actor_id=0&repo_id=9405122&response-content-disposition=attachment%3B%20filename%3Dfrida-server-12.11.4-android-x86.xz&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.110.107
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|52.216.110.107|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7780136 (7.4M) [application/octet-stream]
Saving to: 'frida-server-12.11.4-android-x86.xz'
```

```
root@ubuntu-server:~# ls
frida-server-12.11.4-android-x86.xz
root@ubuntu-server:~# xz -d frida-server-12.11.4-android-x86.xz
root@ubuntu-server:~# ls
frida-server-12.11.4-android-x86
root@ubuntu-server:~#
```

```
root@ubuntu-server:~# adb push frida-server-12.11.4-android-x86 /data/local/tmp
frida-server-12.11.4-android-x86: 1 file pushed. 2.4 MB/s (28188900 bytes in 11.211s)
root@ubuntu-server:~#
```

```
root@ubuntu-server:~# adb shell chmod +x /data/local/tmp/frida-server-12.11.4-android-x86
root@ubuntu-server:~#
```

```
root@ubuntu-server:~# adb shell
root@vbox86p:/ # cd /data/local/tmp
root@vbox86p:/data/local/tmp # ls
frida-server-12.11.4-android-x86
root@vbox86p:/data/local/tmp # █
```

```
root@vbox86p:/data/local/tmp # ./frida-server-12.11.4-android-x86
```

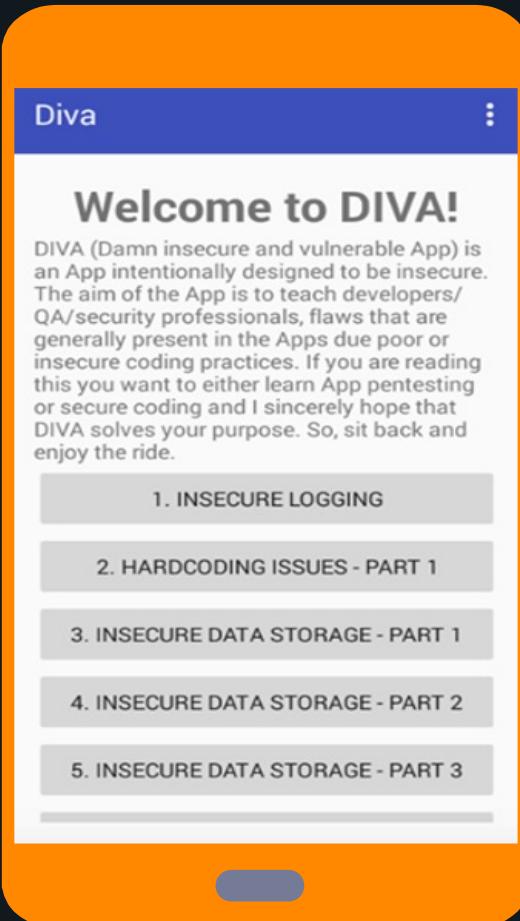
Frida

```
root@ubuntu-server:~# frida-ps -U
PID  Name
-----
134  adbd
1030 android.process.acore
880 android.process.media
277 batteryd
1462 com.android.calendar
1191 com.android.deskclock
1496 com.android.email
849 com.android.inputmethod.latin
930 com.android.launcher3
1329 com.android.messaging
1587 com.android.musicfx
907 com.android.phone
1412 com.android.providers.calendar
1258 com.android.settings
1094 com.android.smspush
717 com.android.systemui
923 com.genymotion.genyd
1532 com.genymotion.superuser
894 com.genymotion.systempatcher
282 debuggerd
280 diskiod
284 drmserver
1762 frida-server-12.11.4-android-x86
132 gatekeeperd
290 genybaseband
301 healthd
```

Frida client ini adalah sebuah tool yang dijalankan di komputer pentester guna untuk menyuntikkan kode atau instruksi tertentu untuk merubah alur dari program ataupun aksi lain guna membantu pentester menganalisis aplikasi.

Perintah **frida-ps -U** untuk mendapatkan daftar aplikasi yang berjalan pada perangkat android kita. Jika tidak muncul daftar aplikasi tersebut, maka bisa diasumsikan bahwa **frida-server** tidak berjalan semestinya.

Decompile APK with APK Studio



Decompile merupakan teknik mengonversi kode program yang dapat dieksekusi (siap dijalankan) (kadang-kadang disebut kode objek) menjadi beberapa bentuk bahasa pemrograman tingkat tinggi sehingga dapat dibaca oleh manusia.

DIVA atau sering dikenal juga dengan sebutan Damn Insecure and Vulnerable Apps merupakan aplikasi mobile yang dibuat untuk belajar keamanan seputar aplikasi mobile. Sehingga bisa dipastikan aplikasi didalamnya sangat tidak aman.

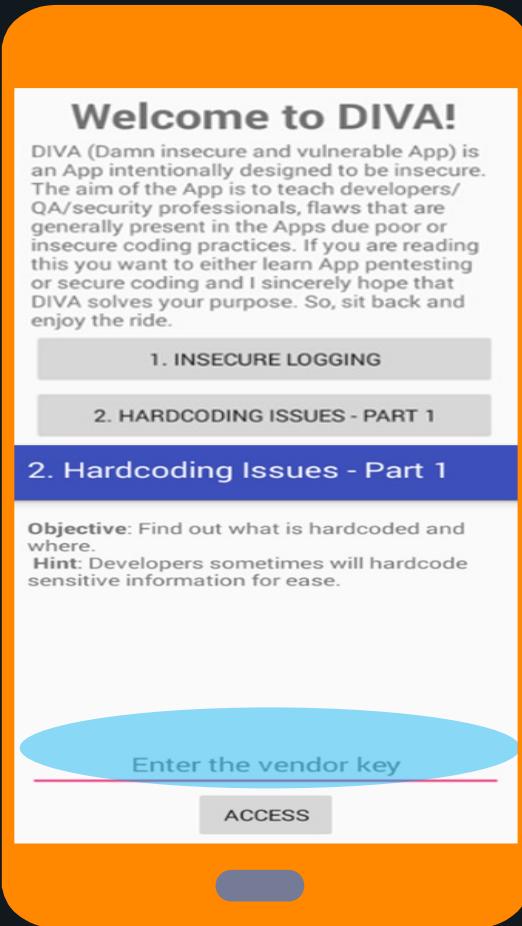
Download APK : Damn Insecure and Vulnerable App (DIVA)
<http://p1.hol.es/test/diva-beta.apk>

The screenshot shows a web browser displaying the contents of the 'diva-beta.apk' file. The URL in the address bar is 'p1.hol.es/test/'. The page title is 'Index of /test/'. Below the title is a table showing the file structure:

Name	Last modified	Size	Description
Parent Directory	26-Jul-2020 05:25	-	
diva-beta.apk	26-Jul-2020 05:22	1468k	

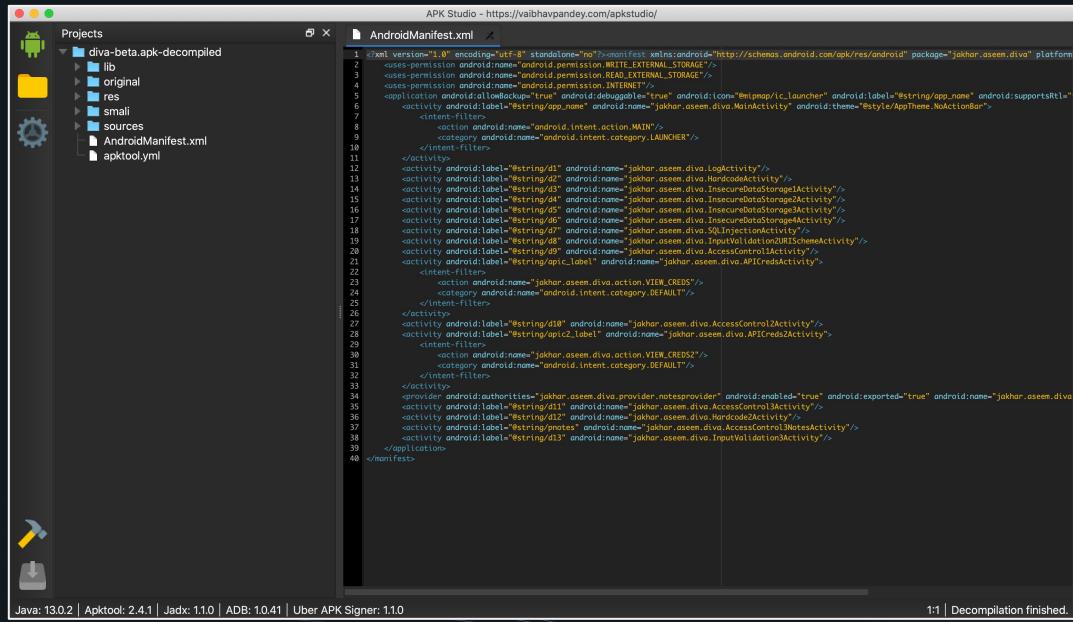
At the bottom of the page, there is a footer message: 'Proudly Served by LiteSpeed Web Server at p1.hol.es Port 80'

Decompile APK with APK Studio



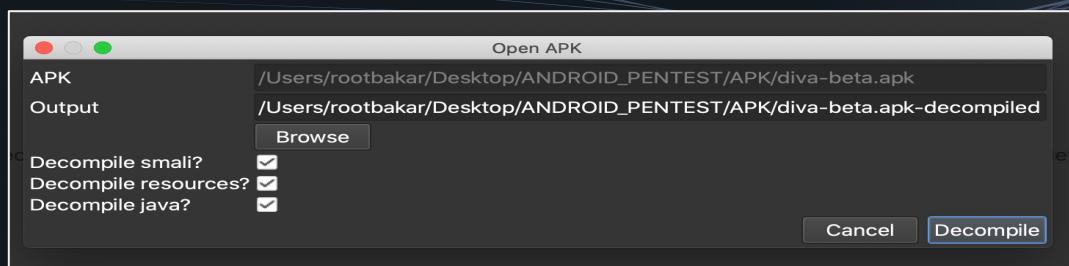
Jalankan DIVA pada android emulator dan akan tampil seperti pada gambar disamping. pilih HARDCODING ISSUE - PART 1. Di dalamnya terdapat sebuah menu yang mengharuskan kita memasukkan vendor key yang belum pernah kita ketahui sebelumnya. Nah untuk mengetahui kode tersebut kita harus membongkar aplikasi DIVA dengan menggunakan APK Studio yang tadi sudah kita bahas pada slide sebelumnya.

Decompile APK with APK Studio



Untuk mengetahui vendor key kita bisa mulai jalankan APK Studio.

Kemudian klik icon android yang ada pada posisi kiri dan pilih apk file dilokasi komputer kalian.



AndroidManifest

AndroidManifest.xml

```
1 <?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" package="jakhar.aseem.diva" platform
2   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
3   <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
4   <uses-permission android:name="android.permission.INTERNET"/>
5   <application android:allowBackup="true" android:debuggable="true" android:icon="@mipmap/ic_launcher" android:label="@string/app_name" android:supportsRtl="true"
6     <activity android:label="@string/app_name" android:name="jakhar.aseem.diva.MainActivity" android:theme="@style/AppTheme.NoActionBar">
7       <intent-filter>
8         <action android:name="android.intent.action.MAIN"/>
9         <category android:name="android.intent.category.LAUNCHER"/>
10      </intent-filter>
11    </activity>
12    <activity android:label="@string/d1" android:name="jakhar.aseem.diva.LogActivity"/>
13    <activity android:label="@string/d2" android:name="jakhar.aseem.diva.HardcodeActivity"/>
14    <activity android:label="@string/d3" android:name="jakhar.aseem.diva.InsecureDataStorageActivity"/>
15    <activity android:label="@string/d4" android:name="jakhar.aseem.diva.InsecureDataStorageActivity"/>
16    <activity android:label="@string/d5" android:name="jakhar.aseem.diva.InsecureDataStorageActivity"/>
17    <activity android:label="@string/d6" android:name="jakhar.aseem.diva.InsecureDataStorageActivity"/>
18    <activity android:label="@string/d7" android:name="jakhar.aseem.diva.SQLInjectionActivity"/>
19    <activity android:label="@string/d8" android:name="jakhar.aseem.diva.InputValidation2URISchemeActivity"/>
20    <activity android:label="@string/api_label" android:name="jakhar.aseem.diva.APIcredsActivity">
21      <intent-filter>
22        <action android:name="jakhar.aseem.diva.action.VIEW_CRED5"/>
23        <category android:name="android.intent.category.DEFAULT"/>
24      </intent-filter>
25    </activity>
26    <activity android:label="@string/d10" android:name="jakhar.aseem.diva.AccessControl2Activity"/>
27    <activity android:label="@string/api2_label" android:name="jakhar.aseem.diva.APIcreds2Activity">
28      <intent-filter>
29        <action android:name="jakhar.aseem.diva.action.VIEW_CRED52"/>
30        <category android:name="android.intent.category.DEFAULT"/>
31      </intent-filter>
32    </activity>
33    <activity android:label="@string/d11" android:name="jakhar.aseem.diva.AccessControl3Activity"/>
34    <provider android:authorities="jakhar.aseem.diva.provider.notesprovider" android:enabled="true" android:exported="true" android:name="jakhar.aseem.diva.
35      <activity android:label="@string/d12" android:name="jakhar.aseem.diva.Hardcode2Activity"/>
36      <activity android:label="@string/notes" android:name="jakhar.aseem.diva.AccessControl3NotesActivity"/>
37      <activity android:label="@string/d13" android:name="jakhar.aseem.diva.InputValidation3Activity"/>
38    </provider>
39  </application>
40 </manifest>
```

```
<activity android:label="@string/app_name" android:name="jakhar.aseem.diva.MainActivity" android:theme="@style/AppTheme.NoActionBar">
<intent-filter>
  <action android:name="android.intent.action.MAIN"/>
  <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
```

Android Manifest (Nama file: AndroidManifest.xml) adalah sebuah xml yang berisi informasi mengenai aplikasi Seperti nama package, level SDK yang digunakan, beserta icon dan nama yang diberikan untuk aplikasi. Selain itu di dalam file tersebut akan terdefinisikan Class mana yang akan dieksekusi pertama kali. Sekedar tambahan informasi juga bahwa permission juga dilampirkan di dalam file tersebut.

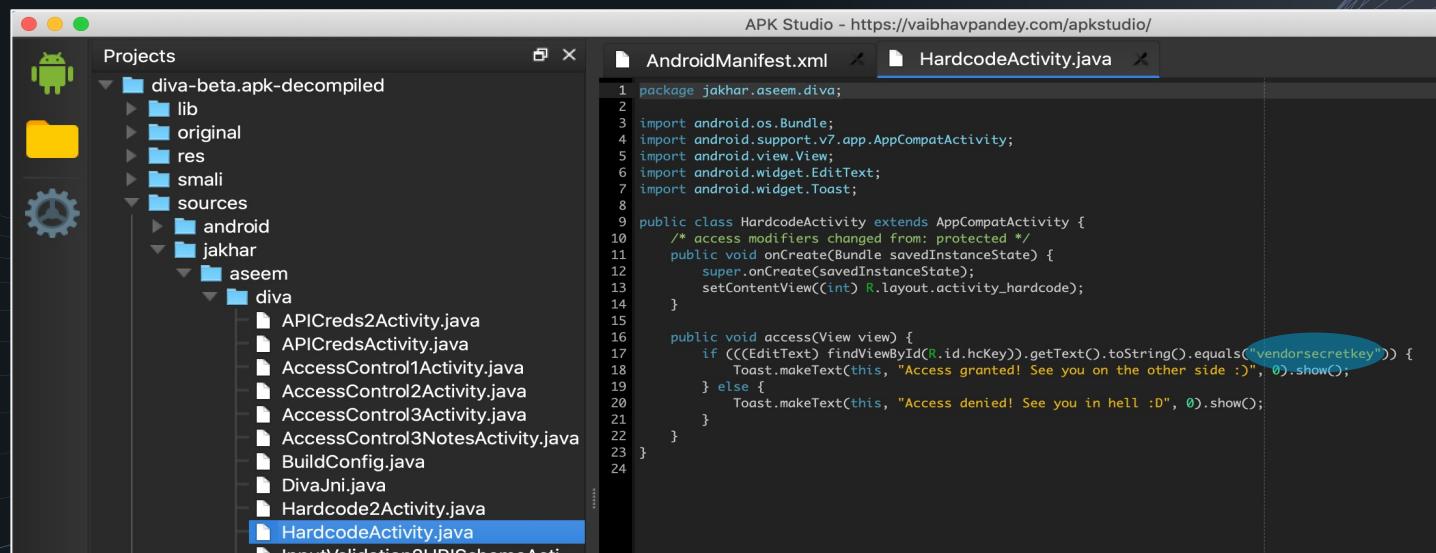
Jika kita amati lebih dalam dari source code diatas, bisa dijelaskan bahwa MainActivity memiliki intent-filter dengan kategori sebagai launcher dan action sebagai main. Artinya MainActivity yang terletak pada folder jakhar/aseem/diva/MainActivity.java adalah Class yang dieksekusi pertama kali.

Activity

Activity adalah komponen pada aplikasi Android yang menampilkan dan mengatur halaman aplikasi sebagai tempat interaksi antara pengguna dengan aplikasi. Dalam hal ini, halaman Hardcore Issue terdapat di lokasi `jakhar/aseem/diva/HardcodeActivity.java` sesuai dengan kutipan baris berikut :

```
<activity android:label="@string/d2" android:name="jakhar.aseem.diva.HardcodeActivity"/>
```

Bisa kalian lihat password secara jelas tertulis sebagai `vendorsecretkey`. Bisa kalian coba langsung di android benar dan tidaknya password tersebut.



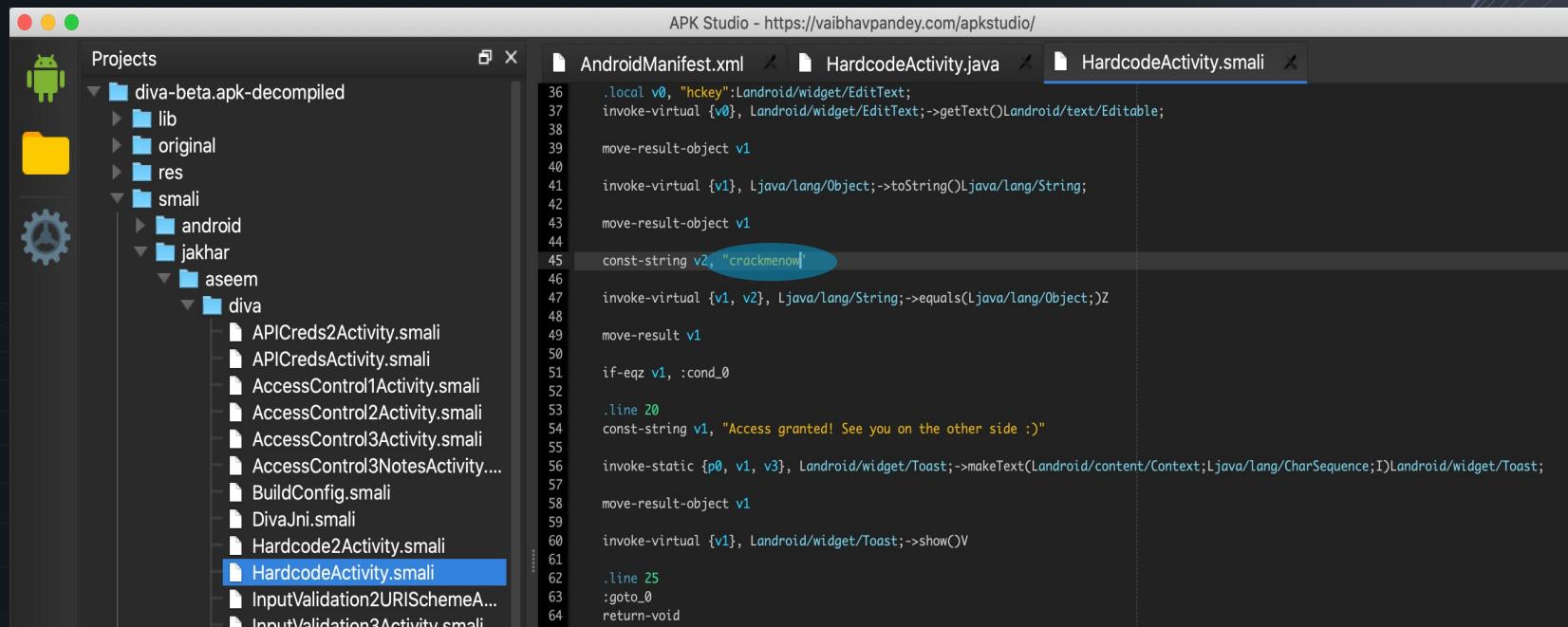
The screenshot shows the APK Studio interface. On the left, the 'Projects' panel displays a tree view of the decompiled APK files, including 'diva-beta.apk-decompiled', 'lib', 'original', 'res', 'smali', 'sources', 'jakhar', 'aseem', and 'diva'. Inside 'diva', several Java files are listed: APIcreds2Activity.java, APIcredsActivity.java, AccessControl1Activity.java, AccessControl2Activity.java, AccessControl3Activity.java, AccessControl3NotesActivity.java, BuildConfig.java, DivaJni.java, Hardcode2Activity.java, HardcodeActivity.java (which is currently selected), and InputValidation2URISchemeActi...'. On the right, the code editor shows the content of HardcodeActivity.java:

```
1 package jakhar.aseem.diva;
2
3 import android.os.Bundle;
4 import android.support.v7.app.AppCompatActivity;
5 import android.view.View;
6 import android.widget.EditText;
7 import android.widget.Toast;
8
9 public class HardcodeActivity extends AppCompatActivity {
10     /* access modifiers changed from: protected */
11     public void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView((int) R.layout.activity_hardcode);
14     }
15
16     public void access(View view) {
17         if (((EditText) findViewById(R.id.hcKey)).getText().toString().equals("vendorsecretkey")) {
18             Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
19         } else {
20             Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
21         }
22     }
23 }
24 }
```

Smali

Teknik selanjutnya kita akan melakukan cracking aplikasi untuk mengganti password **vendorsecretkey** menjadi **sekolahhacker**. Kita akan melakukannya dengan memodifikasi source code **HardcodeActivity.smali**.

ganti **vendorsecretkey** dengan password yang kalian inginkan. contoh : **crackmenow**



APK Studio - <https://vaibhavpandey.com/apkstudio/>

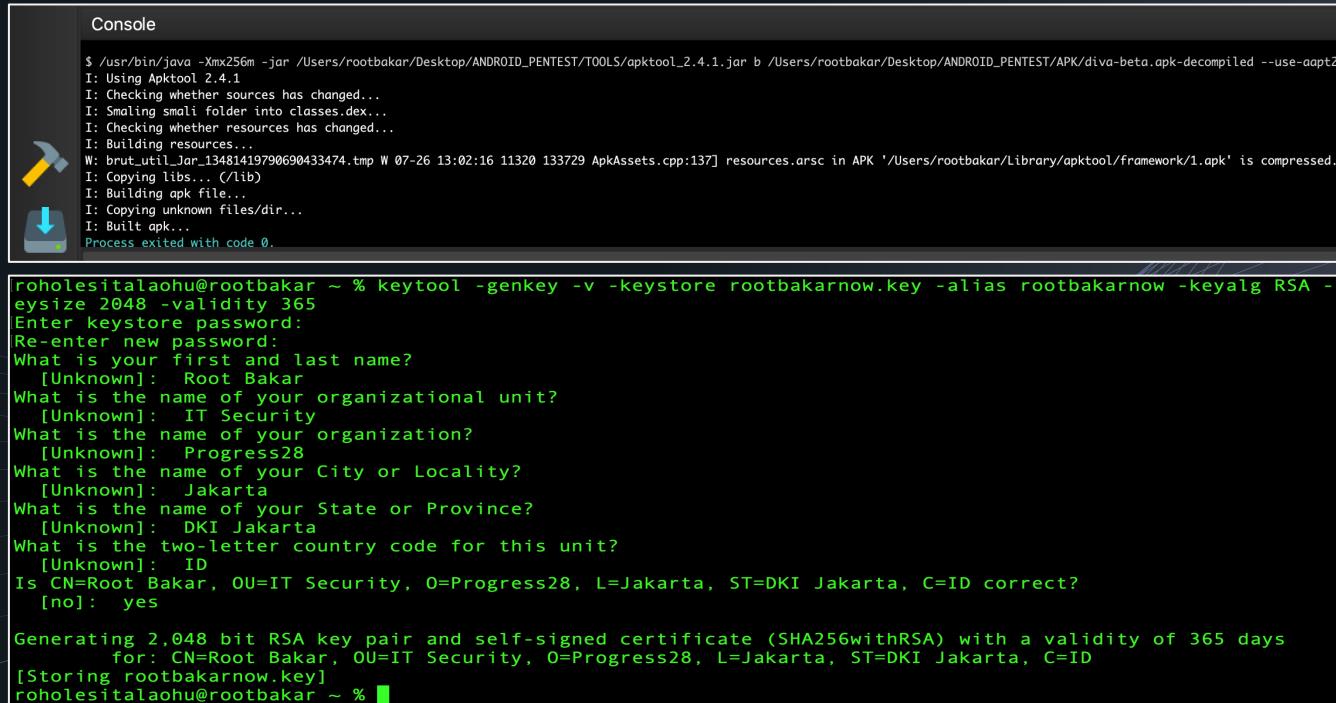
Projects

- diva-beta.apk-decompiled
 - lib
 - original
 - res
 - smali
 - android
 - jakhar
 - aseem
 - diva
 - APICreds2Activity.smali
 - APICredsActivity.smali
 - AccessControl1Activity.smali
 - AccessControl2Activity.smali
 - AccessControl3Activity.smali
 - AccessControl3NotesActivity....
 - BuildConfig.smali
 - DivaJni.smali
 - Hardcode2Activity.smali
 - HardcodeActivity.smali
 - InputValidation2URISchemeA...
 - InputValidation3Activity smali

Build & Patch

Jika sudah diubah menjadi **crackmenow** silahkan simpan dan tekan **CTRL+S** lalu tekan **BUILD** untuk recompile file smali yang telah diubah.

Selanjutnya buat sebuah sertifikat untuk menginstal kembali project dalam bentuk APK yang sudah di tanda tangani dengan menggunakan **keytool**. kemudian tekan **SIGN** maka project berhasil di tanda tangani.



The terminal window shows two sessions:

Session 1 (Apktool build):

```
$ /usr/bin/java -Xmx256m -jar /Users/rootbakar/Desktop/ANDROID_PENTEST/TOOLS/apktool_2.4.1.jar b /Users/rootbakar/Desktop/ANDROID_PENTEST/APK/diva-beta.apk-decompiled --use-aapt2
I: Using Apktool 2.4.1
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
W: brut.util.Jar_13481419790690433474.tmp W 07-26 13:02:16 11320 133729 ApkAssets.cpp:137] resources.arsc in APK '/Users/rootbakar/Library/apktool/framework/1.apk' is compressed.
I: Copying libs... (/lib)
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
Process exited with code 0.
```

Session 2 (keytool generation):

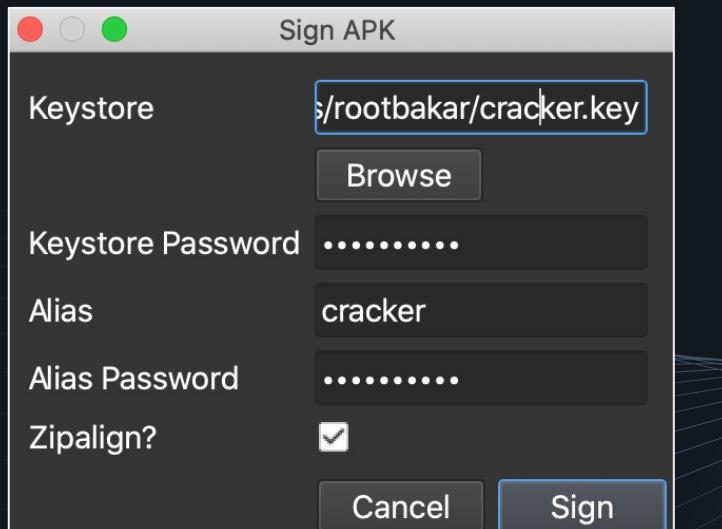
```
roholesitalaohu@rootbakar ~ % keytool -genkey -v -keystore rootbakarnow.key -alias rootbakarnow -keyalg RSA -keysize 2048 -validity 365
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: Root Bakar
What is the name of your organizational unit?
[Unknown]: IT Security
What is the name of your organization?
[Unknown]: Progress28
What is the name of your City or Locality?
[Unknown]: Jakarta
What is the name of your State or Province?
[Unknown]: DKI Jakarta
What is the two-letter country code for this unit?
[Unknown]: ID
Is CN=Root Bakar, OU=IT Security, O=Progress28, L=Jakarta, ST=DKI Jakarta, C=ID correct?
[no]: yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 365 days
for: CN=Root Bakar, OU=IT Security, O=Progress28, L=Jakarta, ST=DKI Jakarta, C=ID
[Storing rootbakarnow.key]
roholesitalaohu@rootbakar ~ %
```

SignIn

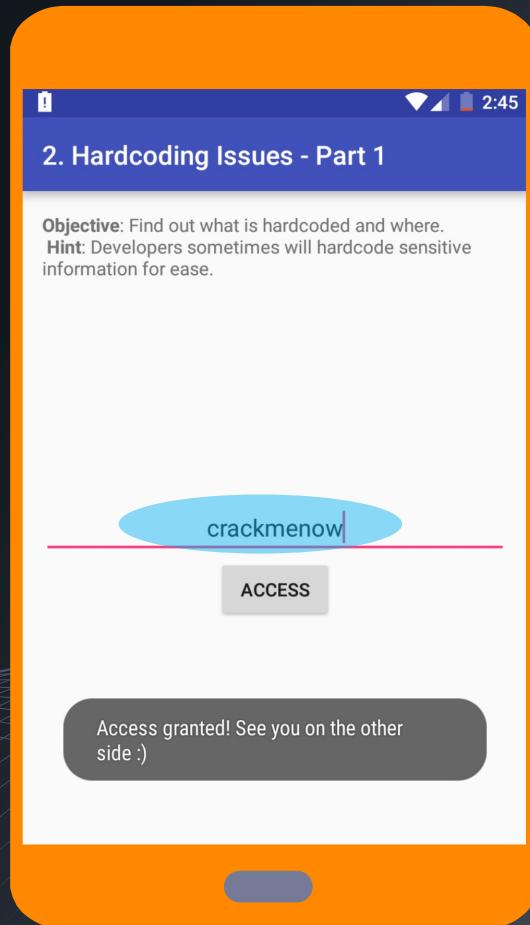
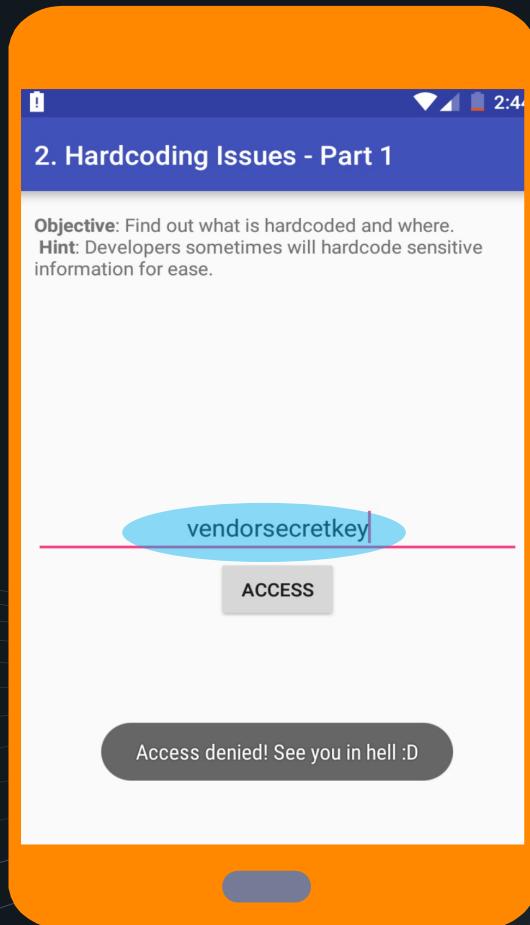
File baru dengan nama **cracker.key** akan terbuat. Kita buka kembali APK Studio > klik project > Sign / Export. Isi sesuai dengan data yang sudah kalian masukkan sebelumnya.

Tekan Sign untuk melanjutkan proses penandatanganan file APK. Jika proses ini sudah dilakukan file APK akan disimpan di folder **dist**.



```
$ /usr/bin/java -Xmx256m -jar /Users/rootbakar/Desktop/ANDROID_PENTEST/TOOLS/uber-apk-signer-1.1.0.jar -a /Users/rootbakar/Desktop/ANDROID_PENTEST/APK/diva-beta.apk-decompiled/dist/diva-beta.apk -keystore /Users/rootbakar/Desktop/ANDROID_PENTEST/APK/diva-beta.apk-decompiled/dist/diva-beta.apk -alias cracker -password cracker -storepass cracker -v
zipalign location: BUILT_IN
/var/folders/vl/41g588jn3vq2x2v3dz79w8th0000gn/T/uapksigner-3312686111168965857/mac-zipalign-29_0_24124476859071547683.tmp
keystore:
[0] 286e50ab /Users/rootbakar/cracker.key (RELEASE_CUSTOM)
01. diva-beta.apk
SIGN
file: /Users/rootbakar/Desktop/ANDROID_PENTEST/APK/diva-beta.apk-decompiled/dist/diva-beta.apk (1.38 MiB)
checksum: f6b1a4e268d75650a40e04e7350d17179954b6b9f88d8f217f8e607a0072ef53 (sha256)
- zipalign success
- sign success
VERIFY
file: /Users/rootbakar/Desktop/ANDROID_PENTEST/APK/diva-beta.apk-decompiled/dist/diva-beta.apk (1.41 MiB)
checksum: cb2e614b552900ddbba0f6037a1dfb9bd9fc6c0cf3b8088b7046e61b25e4497 (sha256)
- zipalign verified
- signature verified [v1, v2, v3]
    Subject: CN=Root Bakar, OU=IT Staff, O=Progress28, L=Jakarta, ST=DKI Jakarta, C=ID
    SHA256: 93d60e1200f738d6c387b08757e3afbd93249a6a4adb4788d4da01c1fb569c / SHA256withRSA
    Expires: Mon Jul 26 13:38:50 WIB 2021
[Sun Jul 26 13:43:03 WIB 2020][v1.1.0]
Successfully processed 1 APKs and 0 errors in 1.84 seconds.
Process exited with code 0.
```

Running APK

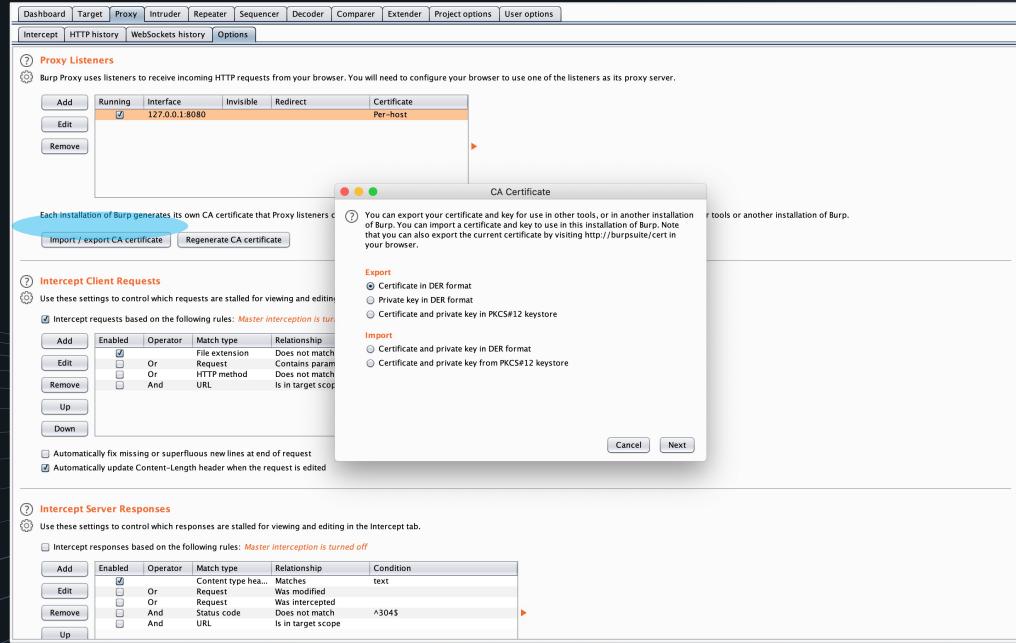


INTERCEPT DATA REQUEST ANDROID WITH BURP

BurpSuite

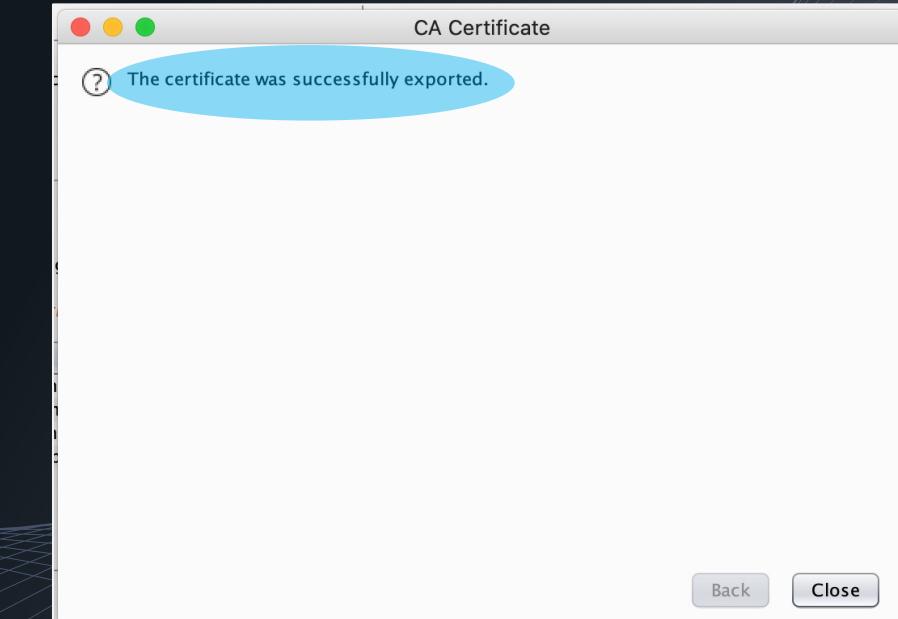
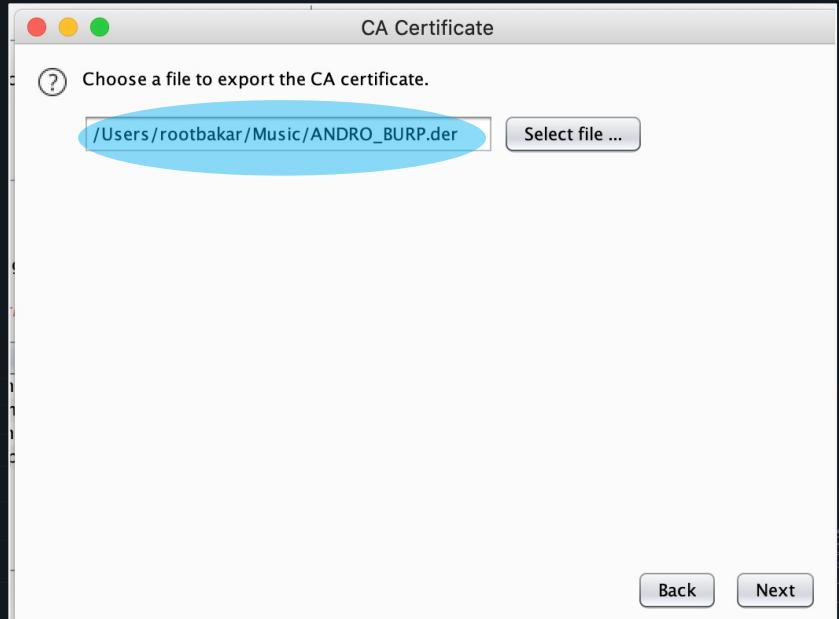
Langkah pertama kita perlu membuat sertifikat secara manual. Hal ini diperlukan agar setiap lalu lintas data yang terjadi di perangkat virtual device android milik kita tercapture oleh **Burp Suite** dengan sertifikat palsu yang telah kita generate. namun sertifikat tersebut masih bisa terbaca dan dianggap sebagai sertifikat milik perangkat virtual device android milik kita yang terverifikasi dan terpercaya.

Dengan **intercept** ini seorang **security engineer** dan **penetration tester** akan mampu melihat permintaan dan respon dari server dan aplikasi.



BurpSuite

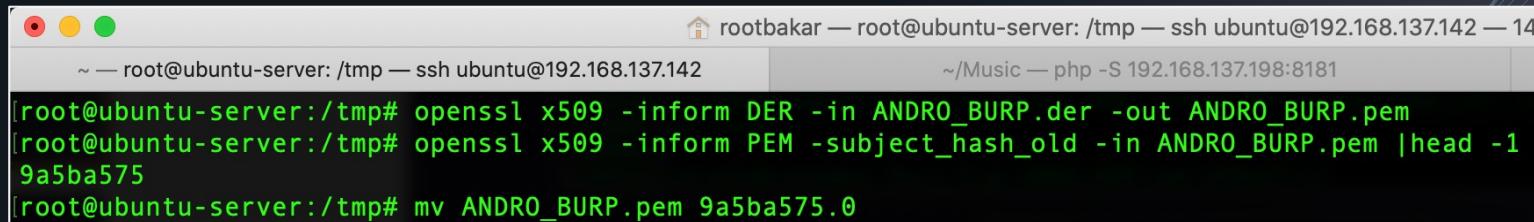
Simpan file dengan extensi .der dan sesuaikan dengan folder yang rekan-rekan kehendaki



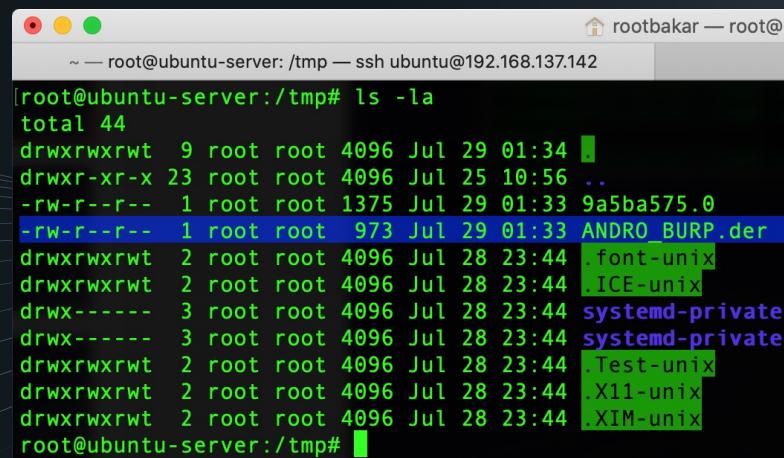
BurpSuite

Command :

```
openssl x509 -inform DER -in ANDRO_BURP.der -out ANDRO_BURP.pem  
openssl x509 -inform PEM -subject_hash_old -in ANDRO_BURP.pem |head -1  
mv ANDRO_BURP.pem <hash>.0
```



```
root@ubuntu-server:/tmp# openssl x509 -inform DER -in ANDRO_BURP.der -out ANDRO_BURP.pem  
root@ubuntu-server:/tmp# openssl x509 -inform PEM -subject_hash_old -in ANDRO_BURP.pem |head -1  
9a5ba575  
root@ubuntu-server:/tmp# mv ANDRO_BURP.pem 9a5ba575.0
```

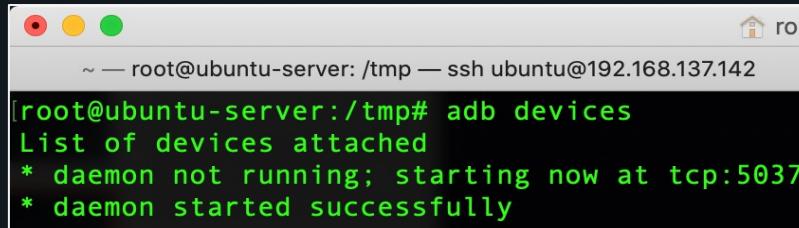


```
root@ubuntu-server:/tmp# ls -la  
total 44  
drwxrwxrwt 9 root root 4096 Jul 29 01:34 .  
drwxr-xr-x 23 root root 4096 Jul 25 10:56 ..  
-rw-r--r-- 1 root root 1375 Jul 29 01:33 9a5ba575.0  
-rw-r--r-- 1 root root 973 Jul 29 01:33 ANDRO_BURP.der  
drwxrwxrwt 2 root root 4096 Jul 28 23:44 .font-unix  
drwxrwxrwt 2 root root 4096 Jul 28 23:44 .ICE-unix  
drwx----- 3 root root 4096 Jul 28 23:44 systemd-private-  
drwx----- 3 root root 4096 Jul 28 23:44 systemd-private-  
drwxrwxrwt 2 root root 4096 Jul 28 23:44 .Test-unix  
drwxrwxrwt 2 root root 4096 Jul 28 23:44 .X11-unix  
drwxrwxrwt 2 root root 4096 Jul 28 23:44 .XIM-unix  
root@ubuntu-server:/tmp#
```

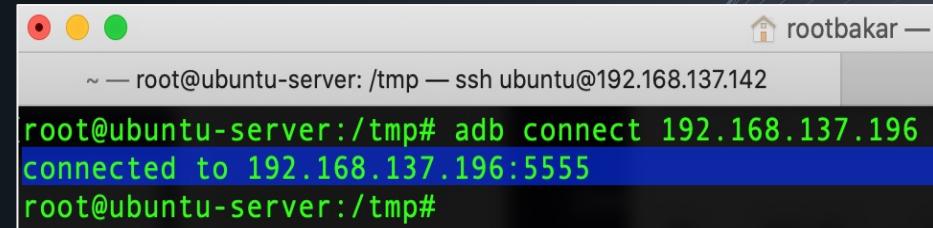
BurpSuite

Command :

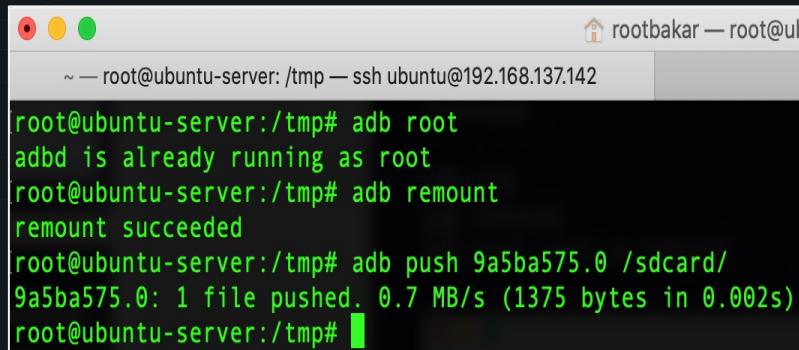
```
adb root  
adb remount  
adb push <cert>.0 /sdcard/
```



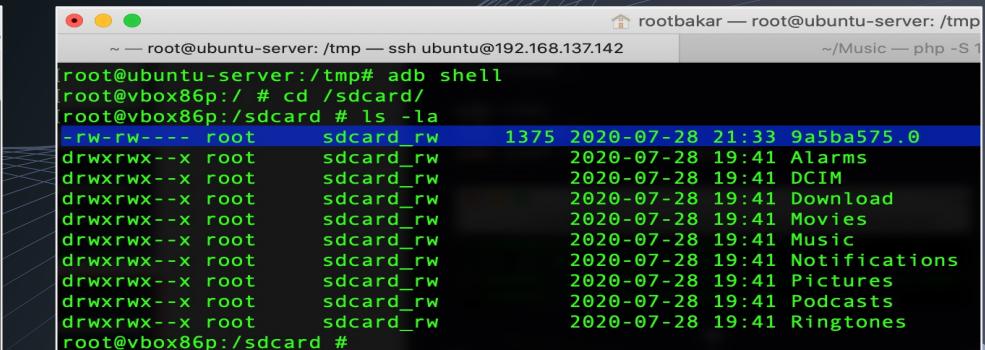
```
root@ubuntu-server:/tmp# adb devices  
List of devices attached  
* daemon not running; starting now at tcp:5037  
* daemon started successfully
```



```
root@ubuntu-server:/tmp# adb connect 192.168.137.196  
connected to 192.168.137.196:5555  
root@ubuntu-server:/tmp#
```



```
root@ubuntu-server:/tmp# adb root  
adb is already running as root  
root@ubuntu-server:/tmp# adb remount  
remount succeeded  
root@ubuntu-server:/tmp# adb push 9a5ba575.0 /sdcard/  
9a5ba575.0: 1 file pushed. 0.7 MB/s (1375 bytes in 0.002s)  
root@ubuntu-server:/tmp#
```

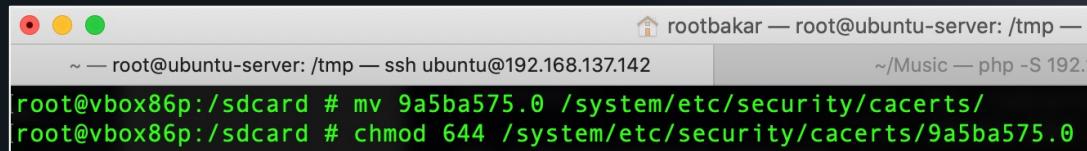


```
root@ubuntu-server:/tmp# adb shell  
root@vbox86p:/ # cd /sdcard/  
root@vbox86p:/sdcard # ls -la  
-rw-rw--- root sdcard_rw 1375 2020-07-28 21:33 9a5ba575.0  
drwxrwx--x root sdcard_rw 2020-07-28 19:41 Alarms  
drwxrwx--x root sdcard_rw 2020-07-28 19:41 DCIM  
drwxrwx--x root sdcard_rw 2020-07-28 19:41 Download  
drwxrwx--x root sdcard_rw 2020-07-28 19:41 Movies  
drwxrwx--x root sdcard_rw 2020-07-28 19:41 Music  
drwxrwx--x root sdcard_rw 2020-07-28 19:41 Notifications  
drwxrwx--x root sdcard_rw 2020-07-28 19:41 Pictures  
drwxrwx--x root sdcard_rw 2020-07-28 19:41 Podcasts  
drwxrwx--x root sdcard_rw 2020-07-28 19:41 Ringtones  
root@vbox86p:/sdcard #
```

BurpSuite

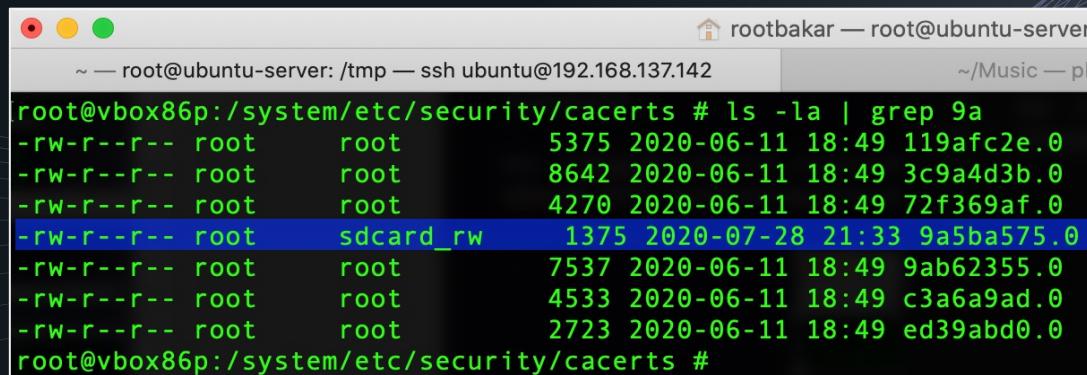
Command :

```
mv /sdcard/<cert>.0 /system/etc/security/cacerts/  
chmod 644 /system/etc/security/cacerts/<cert>.0
```



A terminal window titled "rootbakar — root@ubuntu-server: /tmp — s". The title bar also shows "rootbakar — root@ubuntu-server: /tmp — s" and "~/Music — php -S 192.168.137.142". The command history shows:

```
root@vbox86p:/sdcard # mv 9a5ba575.0 /system/etc/security/cacerts/  
root@vbox86p:/sdcard # chmod 644 /system/etc/security/cacerts/9a5ba575.0
```



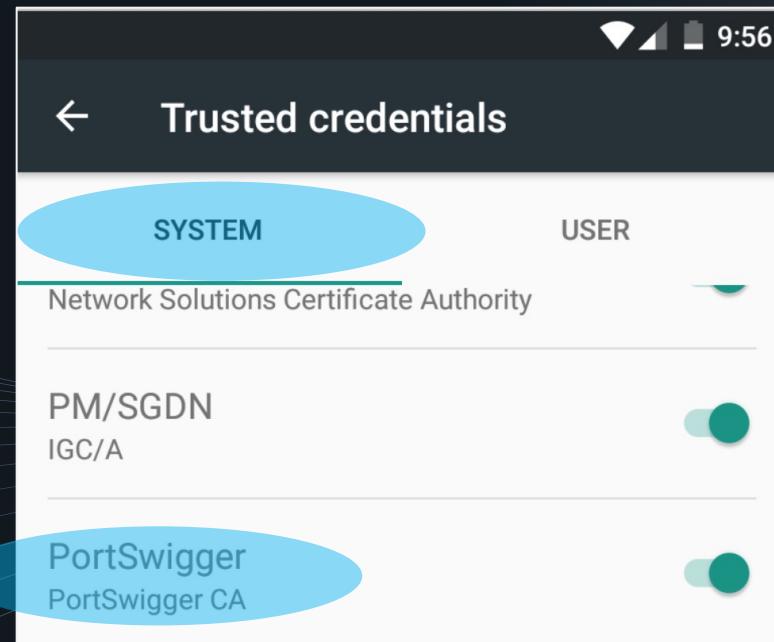
A terminal window titled "rootbakar — root@ubuntu-server". The title bar also shows "rootbakar — root@ubuntu-server: /tmp — s" and "~/Music — php -S 192.168.137.142". The command history shows:

```
root@vbox86p:/system/etc/security/cacerts # ls -la | grep 9a  
-rw-r--r-- root root 5375 2020-06-11 18:49 119afc2e.0  
-rw-r--r-- root root 8642 2020-06-11 18:49 3c9a4d3b.0  
-rw-r--r-- root root 4270 2020-06-11 18:49 72f369af.0  
-rw-r--r-- root sdcard_rw 1375 2020-07-28 21:33 9a5ba575.0  
-rw-r--r-- root root 7537 2020-06-11 18:49 9ab62355.0  
-rw-r--r-- root root 4533 2020-06-11 18:49 c3a6a9ad.0  
-rw-r--r-- root root 2723 2020-06-11 18:49 ed39abd0.0  
root@vbox86p:/system/etc/security/cacerts #
```

BurpSuite

Lakukan pengecekan sertifikat Burp Suite yang sudah kita install pada perangkat virtual android milik kita. Pastikan apakah sertifikat tersebut sudah terdapat pada directory installasi sertifikat yang ada di perangkat virtual android.

Masuk ke menu SETTINGS -> SECURITY -> CREDENTIAL STORAGE (TRUSTED CREDENTIALS)



BurpSuite

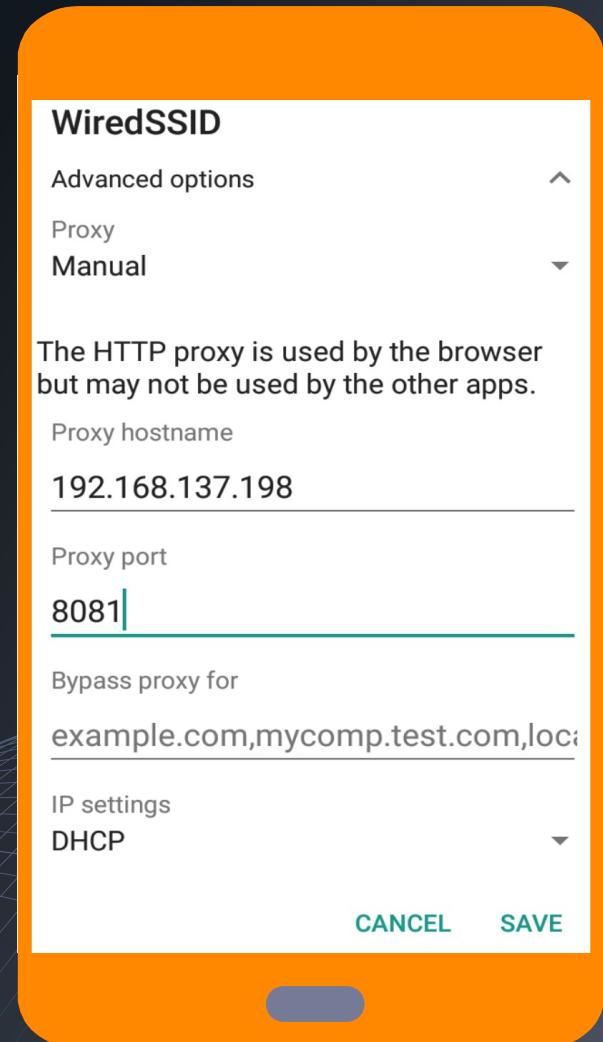
Mengaktifkan Proxy Burp di Android

Langkah terakhir diperlukan agar semua lalu lintas data akan dibelokkan ke aplikasi Burp agar kita bisa melakukan analisa lebih lanjut.

Klik SETTINGS -> WIFI

Jika sudah masuk ke menu WIFI tekan lama pada tulisan **WiredSSID** (sesuaikan dengan nama wifi di emulator masing-masing). Kemudian pilih **Modify Network**

Breakdown pada **Advanced options** dan isi seperti pada gambar dibawah ini.



BurpSuite

Cek intercept dengan Burp Suite.....



The interface shows a request to `http://p1.progress28.com:80`. The 'Intercept is on' button is highlighted with a blue oval. The request details are as follows:

```
1 GET /vulnweb HTTP/1.1
2 Host: p1.progress28.com
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Linux; Android 6.0; SAMSUNG_A50 Build/LOK49F) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.116 Mobile Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 X-Requested-With: com.android.browser
9 Connection: close
```

FRIDA + SSL PINNING TWITTER

SSL Pining using Frida

Sampai tahap ini pasti kalian sudah mengetahui ternyata enkripsi bisa dipatahkan dengan mudah di materi sebelumnya. Hal ini menjadi PR besar seorang security engineer untuk mengatasi teknik intersepsi sertifikat yang ternyata bisa disabotase dengan memasang sertifikat root CA pada perangkat. Akhirnya user bisa tahu permintaan dan respon yang terjadi antara aplikasi dan server.

Pengamanan permintaan dan respon dari API Server adalah salah satu pertahanan awal dimana jika penyerang bisa mendapatkan informasi tersebut akan memudahkan mereka dalam mengidentifikasi kerentanan pada aplikasi mobile ataupun aplikasi web yang melayani API.

Untuk itulah konsep SSL Pinning muncul. **SSL Pinning** merupakan teknik untuk validasi sertifikat disisi aplikasi mobile.

Untuk teknis penerapan ada banyak sekali metode. misalkan saja verifikasi signature hingga menanamkan sertifikat kunci publik dalam aplikasi android. Imbasnya dengan teknik SSL Pinning lalu lintas data yang tadinya muncul di Burp menjadi tidak tertangkap lagi.

SSL Pinning using Frida

Download script SSL Pinning di <https://codeshare.frida.re/@pcipolloni/universal-android-ssl-pinning-bypass-with-frida/>

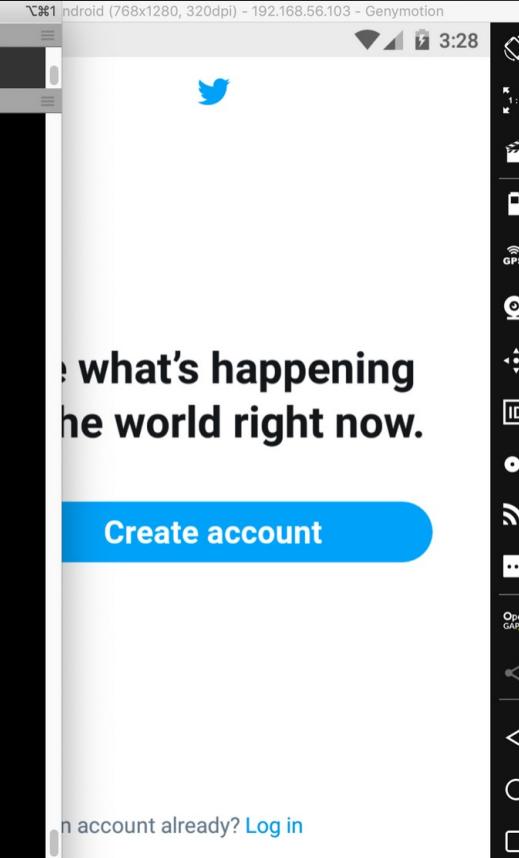
```
[root@ubuntu-server:~# adb push sslfrida.js /data/local/tmp]
```

```
[root@ubuntu-server:~# adb shell  
[root@vbox86p:/ # cd /data/local/tmp  
[root@vbox86p:/data/local/tmp # ls  
frida-server-12.11.4-android-x86  
root@vbox86p:/data/local/tmp # ]
```

```
[root@vbox86p:/data/local/tmp # ./frida-server-12.11.4-android-x86
```

```
Python
adb
126|vbox86p:/data/local/tmp # ./frida-server-12.10.4-android-x86
Python
[Android::com.twitter.android]-
[Android::com.twitter.android]-
[Android::com.twitter.android]-
[Android::com.twitter.android]-
[Android::com.twitter.android]- exit

Thank you for using Frida!
sh-3.2# frida -U -f com.twitter.android -l pinning_ssl1.js --no-pause
    / _ |  Frida 12.10.4 - A world-class dynamic instrumentation toolkit
  ! ( _ |
  > _ |  Commands:
  /-/ _ |    help      -> Displays the help system
  . . . .   object?   -> Display information about 'object'
  . . . .   exit/quit -> Exit
  . . . .
  . . . .   More info at https://www.frida.re/docs/home/
Spawned `com.twitter.android`. Resuming main thread!
[Android::com.twitter.android]-
===
* Injecting hooks into common certificate pinning methods *
===
* Setup custom trust manager
* Setup okhttp3 pinning
* Unable to hook into trustkit pinner
* Setup TrustManagerImpl pinning
* Unable to hook into Appcelerator pinning
! Intercepted trustmanager request
! Intercepted trustmanager request
! Intercepted okhttp3: api.twitter.com
! Intercepted okhttp3: api.twitter.com
! Intercepted okhttp3: api.twitter.com
! Intercepted okhttp3: api.twitter.com
[Android::com.twitter.android]- ! Intercepted okhttp3: api.twitter.com
```



The screenshot shows a Frida terminal window on the left and a Genymotion Android emulator window on the right. The Frida terminal displays the results of running the pinning script against the Twitter app. The Genymotion window shows the Twitter login screen with the text 'See what's happening in the world right now.' and a 'Create account' button.

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

Site map Scope Issue definitions

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Contents

Host	Method	URL	Params	Status
https://api.twitter.com	POST	/1.1/guest/activate.json		✓ 200
https://api.twitter.com	GET	/1.1/help/settings.json		✓ 200
https://api.twitter.com	POST	/1.1/jot/client_event		✓ 200
https://api.twitter.com	POST	/1.1/jot/client_event		✓ 200
https://api.twitter.com	POST	/1.1/jot/client_event		✓ 200
https://api.twitter.com	POST	/1.1/jot/client_event		✓ 200
https://api.twitter.com	POST	/1.1/jot/client_event		✓ 200
https://api.twitter.com	POST	/1.1/jot/client_event		✓ 200
https://api.twitter.com	POST	/1.1/jot/client_event		✓ 200
https://api.twitter.com	POST	/1.1/jot/client_event		✓ 200
https://api.twitter.com	GET	/1.1/help/settings.json		✓ 304
https://api.twitter.com	GET	/1.1		

Request Response

Raw Params Headers Hex

```

1 POST /1.1/guest/activate.json HTTP/1.1
2 TImezone: GMT
3 Optimize-Body: true
4 Accept: application/json
5 X-Twitter-Client: TwitterAndroid
6 User-Agent: TwitterAndroid/6.53.0-release.01 (18530001-r-1)
7 Android/8.0.0 (Google Edition/Android;Android;vbox86p;0;1;2013)
8 X-Twitter-Client-AppID: 02fd9e77-bd95-4ec8-9ffb-17a9561cf478
9 Accept-Encoding: gzip, deflate
10 X-Twitter-Client-Language: en-US
11 X-Client-UUID: f78d7c9a-3c43-4c8c-88d5-d2290f3181d8
12 X-Twitter-Client-DeviceID: 83832a9d77227076
13 Authorization: Bearer
14 AAAAAAAAAAAAAAAAFAFxAwAAAAAAAMHCxpeSDG1gLNLghVe8d74h1k4%
15 3DRUMF4xQLsbeHTSRRcIopJtxoGWeyHrb5tE2jpGsKWDFW82F
16 X-Twitter-Client-Version: 8.53.0-release.01
17 X-Play-Label: 3d7d59a4070975e9
18 X-Twitter-Client-Max-Ad-Tracking: 0
19 Accept-Language: en-US
20 X-Twitter-Client-Flavor:
21 Content-type: text/plain; charset=ISO-8859-1
22 Content-length: 0
23 Host: api.twitter.com
24 Connection: close
25 Cookie: personalization_id=v1_03bcT3mJ2fU33LrT7d515w==;
           guest_id=v1%3A159495625845738087

```

Issues

- Session token in URL
- Strict transport security not enforced
- Cookie scoped to parent domain [3]
- Cookie without HttpOnly flag set [3]
- Browser cross-site scripting filter disabled [5]

Advisory Request Response

Session token in URL

Issue: Session token in URL
Severity: Medium
Confidence: Firm
Host: https://api.twitter.com
Path: /1.1/help/settings.json

Issue detail
The URL in the request appears to contain a session token within the query string:

- https://api.twitter.com/1.1/help/settings.json?feature_set_token=f192a6b739500078e35b6a5447a54e9af58879c2&settings_version=4531d0449c529f19b2e00e0340e907b1

Issue background
Sensitive information within URLs may be logged in various locations, including the user's browser, the web server, and any forward or reverse

TERIMA KASIH