

بسم خدا

نام: سید حجت حسینی

وبسایت: <https://hojat.net/>

شهریور ماه 1397

Android Application Penetration Testing

تست نفوذ برنامه های اندروید

مقدمه

این دوره آموزشی درباره تست های امنیتی بر روی برنامه های اندروید می باشد که ما در نهایت به شما آموزش میدهیم چگونه هر برنامه ی اندرویدی را تست آسیب پذیری بگیریید و در نهایت راه کارهای امنیتی را بکار ببرید. دوره تست آسیب پذیری برنامه های اندرویدی برای برنامه نویسان و Pen tester ها و علاقه مندان به امنیت یک دوره ضروری می باشد ضمن آنکه دوره فوق از سطح خیلی مبتدی تا سطح پیشرفته آموزش داده می شود و آسیب پذیری های real-time پوشش داده شده است.

آنچه یاد خواهید گرفت

- Android Architecture and Permission model
- Android App Components
- Setting up an testing lab
- Reversing application using jadx, apktool, dex2jar etc...
- Various vulnerabilities
- Insecure Logging
- Leaking content provider
- Insecure Data Storage
- Client Side Injection – SQLi
- Malware Analysis
- DoS (Denial of Service)
- Pen testing DIVA – Damn Insecure Vulnerable App
- Drozer

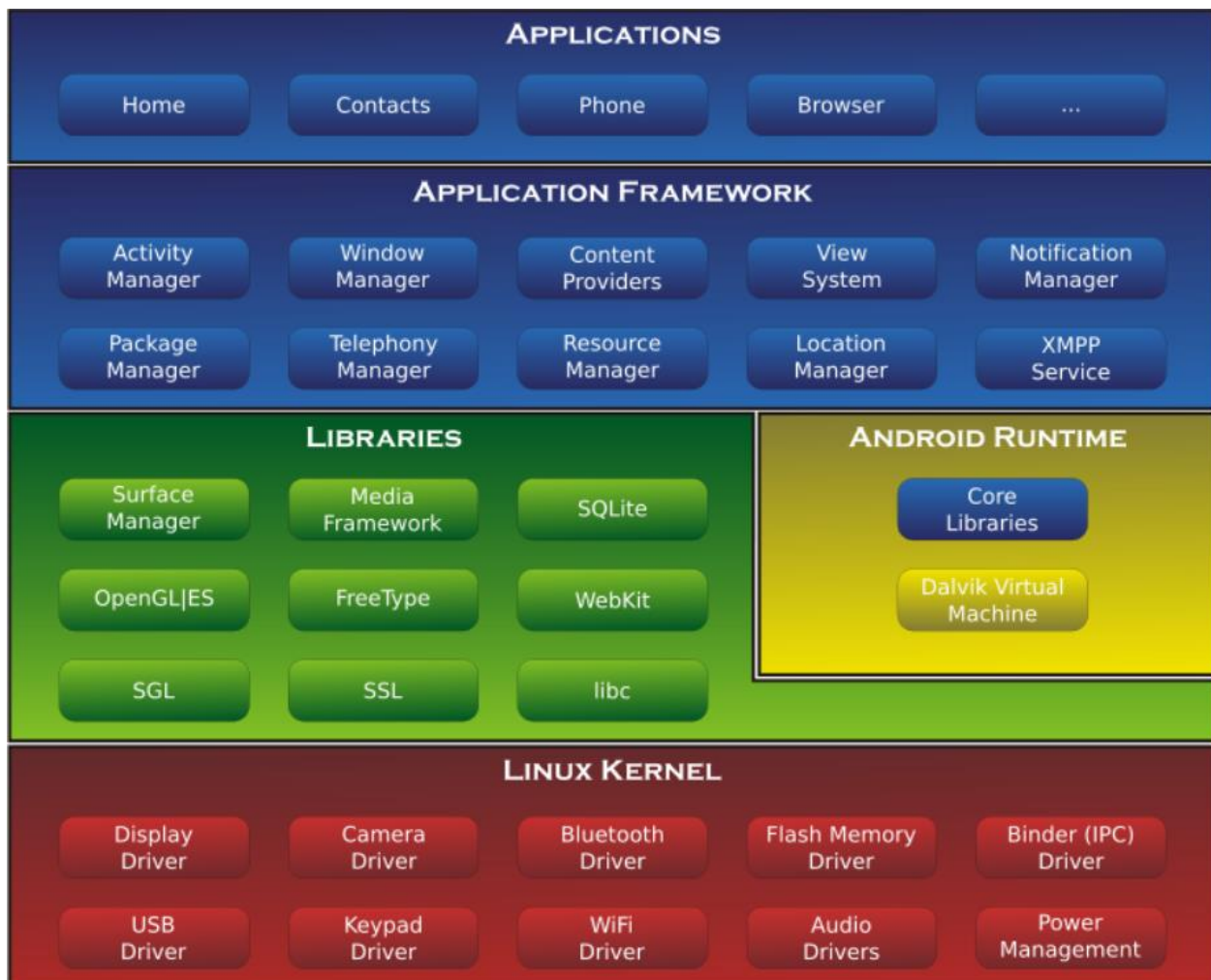
Android Architecture

معماری اندروید

قبل از اینکه در مورد چیزای دیگر صحبت شود بهتر است اول بدانیم اندروید چیست؟ بسیار خوب، اندروید محبوبترین سیستم عامل اوپن سورس موبایل می باشد و اولین انتخاب برنامه نویسان و مصرف کنندگان است و شما میتوانید از لینک زیر سورس اندروید را دریافت کنید و دانش خود را در زمینه اندروید بالا ببرید.

<https://source.android.com/>

معماری اندروید به چهار بخش کلی زیر تقسیم می شود.



- در لایه Application تمام برنامه هایی که شما در رابط گرافیکی شما وجود دارد مانند مرورگر، فیس بوک و... قرار میگیرند
- در لایه Application Framework تمامی کامپوننت ها و سرویس ها و فعالیتها قرار میگیرند
- لایه بعدی به دو بخش Libraries و Android Runtime تقسیم می شود.
 - در لایه Libraries مجموعه کتابخانه ها به زبان C و C++ قرار دارد که به عنوان کتابخانه های خارجی استفاده میشود مانند کتابخانه SQLite که برای پایگاه می باشد و کتابخانه SSL و مابقی کتابخانه ها که برای استفاده برنامه های کاربردی اندروید مورد استفاده قرار میگیرد.
 - در لایه Android Runtime مجموعه کتابخانه اصلی و توابع جاوا و ماشین مجازی dalvik قرار دارد که در ادامه توضیحات تکمیلی در مورد این لایه داده میشود.
- لایه Linux Kernel واسط سخت افزار و لایه های بالاتر می باشد

Android Security Architecture

در این بخش قصد داریم در مورد امنیت معماری اندروید صحبت کنیم و چون اندروید بر مبنای لینوکس می باشد و در بالاترین قسمت لینوکس اجرا می شود بنابراین به طور کلی به دو بخش Linux base و Android security تقسیم می شود.

در بخش Linux base ما دو موضوع زیر را بررسی می کنیم

- Privilege control
- Allocate separate PID to each app

و در بخش Android security به موضوع کنترل سطح دسترسی برنامه ها می پردازیم که در زمان نصب به برنامه بر روی گوشی که دسترسی هایی را لازم دارد.

در بخش Linux base Privilege Control هر اپلیکیشنی که PID و UID جداگانه گرفته است بنابراین اگر مالک User ID همان اپلیکیشن است Presses ID هم باید مربوط به همان UID باشد

```
root@vbox86p:/data/data/jakhar.aseem.diva # ls -la
drwxrwx--x u0_all u0_all 2017-08-23 05:56 app_webview
drwxrwx--x u0_all u0_all 2017-08-22 07:45 cache
drwxrwx--x u0_all u0_all 2017-08-23 06:29 databases
lrwxrwxrwx install install 2017-08-29 13:47 lib -> /data/app-lib/jakhar.aseem.diva-l
drwxrwx--x u0_all u0_all 2017-08-24 04:12 shared_prefs
-rw----- u0_all u0_all 19 2017-08-23 05:45 uinfo588328638tmp
```

```
u0_a5 5054 277 562896 35324 ffffffff b773807b S android.process.media
root 5179 68 1556 736 c014c89e b7636716 S /system/bin/sh
u0_all 5192 277 584348 41820 ffffffff b773807b S jakhar.aseem.diva
root 5222 5179 1868 504 00000000 b76be146 R ps
```

یکی از مهمترین فایل های هر برنامه ی اندروید، فایل AndroidManifest.xml می باشد که ویژگی های خاص اپلیکیشن های اندرویدی را تعیین می کند و تمام Permission هایی که برنامه نویس مشخص می کند در این فایل قرار می گیرد.

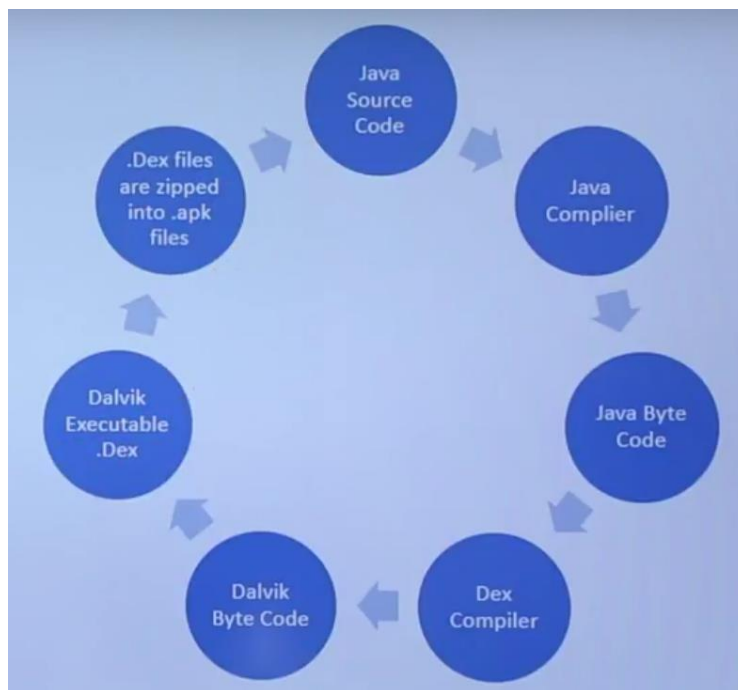
```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-feature android:name="android.hardware.camera"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
```

Android Application Development Cycle

چرخه توسعه برنامه اندروید

یک برنامه اندروید از زمان طراحی تا انتشار 7 مرحله ی زیر را طی می کند

- Java source code
- Java compiler
- Java byte code
- Dex compiler
- Delvik byte code
- Delvik executable .dex
- .dex files are zipped into .apk files



در این قسمت برنامه نویسی سورس برنامه نوشته شده به زبان جاوا را با کامپایلر جاوا تبدیل به java byte code می کند و بعد از آن با Dex کامپایلر تبدیل به Dalvik Byte code می کند که اگر بیاد بیاروید قبلا توضیح داده شد که در اندروید ماشین مجازی برای جاوا طراحی شده است که در بخش بعدش آنرا قابل اجرا می کند و Dalvik Executable یا DEX می شود و در نهایت تمامی فایل ها داخل یک فایل apk فشرده می شوند که شما میتوانید از آن استفاده و نصب کنید.

Android Application Components

کامپوننت های برنامه های اندروید

در این قسمت ما با انواع کامپوننت های مهم برنامه اندروید آشنا می شویم و اینکه چطور در هر برنامه ی اندرویدی استفاده می شوند. که یک برنامه اندرویدی شامل کامپوننت های زیر می باشد که در مورد هر کدام توضیحات لازم داده می شود



- Activity: یک اپلیکیشن ایمیل می تواند دارای فعالیت های زیر باشد: فعالیتی که لیستی از ایمیل های جدید را نمایش می دهد، یک فعالیت برای نوشتن یک ایمیل و یک فعالیت دیگر برای خواندن ایمیل ها
- Intents: مولفه های پیام ها را به هم متصل می کند
- Services: یک سرویس می تواند در پس زمینه موسیقی پخش کند، درحالیکه کاربر در اپلیکیشن دیگری است
- Broadcast Receivers: اپلیکیشن ها می توانند پیام های برودکستی را برای آگاهی سایر برنامه ها بفرستند مبنی بر اینکه داده ی موردنظر دانلود شده و آماده ی استفاده است
- Content Providers: در صورت درخواست داده های یک اپلیکیشن را به سایر اپلیکیشن ها عرضه می کند

Setting up Android Testing Lab

راه اندازی لابراتور تست امنیت اندروید

در این بخش با راه اندازی آزمایشگاه تست آسیب پذیری برنامه های اندروید و ابزارهایی که برای آن نیاز داریم آشنا می شوید.

همانطور که میدانید لینوکس توزیع های مختلفی برای هر زمینه دارد و برای تست نفوذ موبایل توزیع SANTOKU ارائه شده است که بر روی سه بخش mobile forensics و mobile malware و mobile security تمرکز کرده است و علاوه بر رایگان بودن به صورت Open Source می باشد که از لینک زیر قابل دریافت می باشد.



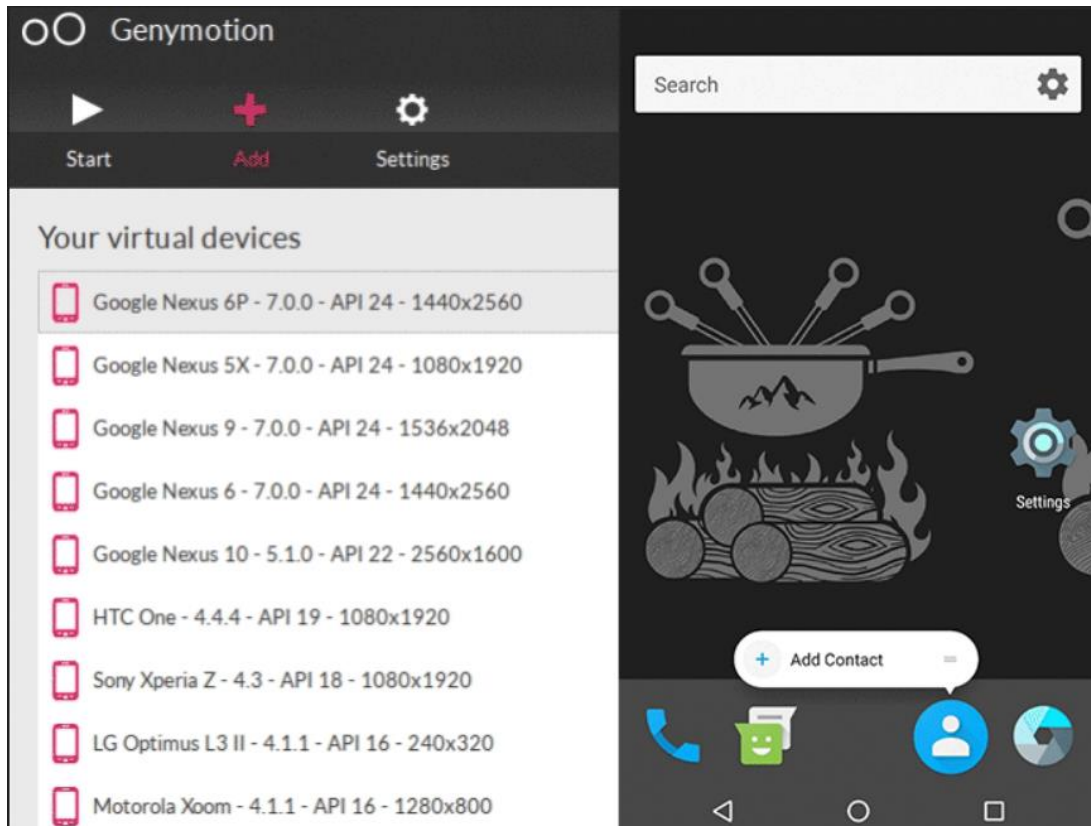
<https://santoku-linux.com/>

دومین ابزاری که به آن نیاز داریم ما Virtual Box می باشد که مجازی سازی استفاده می شود و از لینک زیر قابل دریافت می باشد



<https://www.virtualbox.org/>

و در نهایت سومین ابزاری که به آن نیاز پیدا میکنیم Genymotion است. این ابزار یک شبیه ساز اندروید می باشد که میتوانیم نسخه های مختلف اندروید را بر روی آن نصب کنیم و همانند موبایل از آن استفاده کنیم که میتوانید از لینک ابزار فوق را دریافت کنید



<https://genymotion.com/>

Android Debug Bridge (ADB)

آشنایی با دستورات ADB

یک رابط کنسولی و کامندی برای اتصال به اندروید می باشد که بدون نیاز به روت کردن دستگاه می توان عملیاتی مانند نصب یا حذف برنامه و دستورات دسترسی Shell انجام داد. در جدول زیر به مجموعه دستورات پر کاربرد آن می پردازیم.

دستور	توضیحات
Adb connect IP address:5555	اتصال به دستگاه اندروید
Adb devices	نمایش دستگاه های متصل
Adb shell	گرفتن دسترسی Shell
Adb -s (serial id) shell	سوئیچ -s برای انتخاب دیوایس می باشد
Adb install app name	نصب برنامه
Adb uninstall app name	پاک کردن برنامه
Adb push file-name directory-to-be-sent	دانلود فایل از دستگاه اندروید به کامپیوتر
Adb pull file-name	آپلود فایل بر روی دستگاه اندروید
Adb logcat	مانیتورینگ و کیچرینگ لاگ های برنامه

Android Startup Process

فرایند راه اندازی اندروید

سیستم عامل اندروید به طور کلی پنج مرحله اصلی را باید طی کند تا به صورت کامل راه اندازی شود که در زیر به تفکیک هر مرحله توضیح داده شده است.

- **Bootloader**

یک برنامه کوچکی است که قبل از سیستم عامل اندروید میشود و سازنده محدودیت در آن تعریف میکند

- **Init process**

پروسه init در بکگراند اجرا میشود و برای بارگزاری کانفیگ ها و کامپوننت ها استفاده می شود و در واقع اولین فرایند اندروید به شمار میرود و دو مسئولیت مهم دارد

1. نصب دایرکتوری مانند `/sys` ، `/dev` ، `/proc`

2. اجرای اسکریپت `init.rc`

- **Zygote process**

در واقع فرایند zygote به عنوان یکی از زیرفرایندهای init شناخته میشود که در boot اجرا می شود و هر فرآیندی که کامپوننت را داخل پوشه خودش بارگزاری کند یه نوع فرآیند zygote است که توسط init اجرا شده است و در مقابل ماشین مجازی Dalvik پاسخگو میباشد.

- **Dalvik Virtual Machine**

ماشین مجازی Dalvik برای اجرای برنامه های جاوا می باشد که توسط فرآیند zygote برای جلوگیری از پر شدن حافظه و حداقل زمان راه اندازی مدیریت میشود

- **Boot completed broadcast**

زمانی که تمام فرآیندهای قبلی و ماشین مجازی Dalvik بدون مشکل و به طور کامل اجرا شده اند، به تمام کامپوننت ها و برنامه ها یک پیام همگانی ارسال میشود که بوت به طور کامل انجام شد

Unzipping Android Application

Unzip کردن برنامه اندروید

یک برنامه اندروید در واقع یک فایل بایگانی است که شامل تمامی فایل های برنامه میباشد که با پسوند apk شناخته می شود و زمانی که یک فایل apk را از حالت فشرده خارج میشود، میتوان تمام فایل ها پوشه های برنامه را مشاهده کرد. ولی باید توجه داشت که نتیجه ی از حالت فشرده خارج کردن فایلها توسط unzip با دیکامپایل کردن برنامه متفاوت میباشد

به طور کلی بعد از unzip کردن یک فایل apk موارد زیر قابل مشاهده می باشد

- Classes.dex (file)
- AndroidManifest.xml (file)
- META-INF (folder)
- Res (folder)
- Assets (folder)
- lib (folder)

Reversing Android Application

دیکامپایل کردن برنامه های اندروید

ابزارهای زیادی برای دیکامپایل کردن برنامه های اندروید وجود دارد که سریعترین و کاربردی ترین آنها برنامه android-apktool می باشد و دومین ابزار برای دیکامپایل کردن فایل های apk استفاده از برنامه Jadx میباشد.

نحوه ی استفاده از هر دو ابزار به شکل زیر می باشد

android-apktool d sample.apk •

```
I: Using Apktool 2.2.4 on diva-beta.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/beast/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

JaDX -d output-folder sample.apk •

Application signing and Building

ایجاد امضا برای برنامه اندروید و کامپایل کردن آن

در قسمت قبلی با نحوه دیکامپایل کردن برنامه های اندروید و دیدن سورس کدها، مجوزها و... آشنا شدید و در این بخش میخواهیم بعد از اعمال تغییرات خود در برنامه مجدد فایل apk را بسازیم که با برنامه apk-tool میتوان پوشه برنامه را کامپایل کرد و فایل apk ساخته میشود.

- apktool b applicationfolder

برای انتشار برنامه ی ساخته در play store و... میبایست برنامه ی خود را دارای امضای کنید. با استفاده از امضای برنامه یا Application signing تمام پکت ها به صورت رمز شده تبدیل میشود که امضای برنامه به صورت یونیک در اندروید استفاده می شود و برنامه بعد از ساخته شدن میتواند در playstore قرار گیرد که به دونوع مکانیزم کلی زیر تقسیم میشود.

- Certificate Authority (CA)
- Self-signing certificate

Analyzing Dex Files

آنالیز کردن فایل های Dex

فایل های dex یا Dalvik Executable File شامل اطلاعات زیادی در مورد برنامه می باشد که در برنامه های اندروید با فایلی با نام classes.dex وجود دارد که در ماشین مجازی Dalvik اجرا می شود و به دو روش می توان آنالیز کرد

- با استفاده از ابزار کامپایل و تبدیل به فایل jar

روش اول با استفاده از ابزار

- Hex Dump

از ابزار hexdump میتوان برای خواندن و آنالیز فرمت dex در قالب زبان hex استفاده کرد.

```
0000000 6564 0a78 3330 0035 27e4 81ee 4003 55d9
0000010 6ac5 37f0 c750 6efd 01e6 8442 1a34 048f
0000020 8a28 002c 0070 0000 5678 1234 0000 0000
0000030 0000 0000 9ea8 0007 4dca 0000 0070 0000
0000040 096e 0000 3798 0001 0dbf 0000 5d50 0001
0000050 26d6 0000 0244 0002 4b4b 0000 38f4 0003
0000060 06ff 0000 934c 0005 eb80 0024 9ea8 0007
0000070 6d44 001a 6d46 001a 6d49 001a 6d4c 001a
0000080 6d50 001a 6d55 001a 6d5b 001a 6d60 001a
0000090 6d6f 001a 6d86 001a 6db9 001a 6de1 001a
00000a0 6df1 001a 6e14 001a 6e2e 001a 6e4b 001a
00000b0 6e6c 001a 6e84 001a 6eab 001a 6ed3 001a
00000c0 6efa 001a 6f22 001a 6f43 001a 6f4b 001a
00000d0 6f68 001a 6f8b 001a 6fa8 001a 6fb6 001a
00000e0 6fc5 001a 6fd3 001a 6fe1 001a 7002 001a
00000f0 7015 001a 7023 001a 7032 001a 7040 001a
0000100 7053 001a 705d 001a 7074 001a 708c 001a
0000110 70a7 001a 70b3 001a 70c2 001a 70ce 001a
0000120 70e1 001a 70f7 001a 710d 001a 7124 001a
0000130 712f 001a 7145 001a 7149 001a 714d 001a
```

- Dex Dump

با ابزار dexdump میتوان فایل های dex را به دو فرمت xml و plaint text تبدیل آنالیز کرد

- Dexdump -l xml classes.dex
- Dexdump -l plain classes.dex

```
<class name="AnimRes"
  extends="java.lang.Object"
  abstract="true"
  static="false"
  final="false"
  visibility="public"
>
<implements name="java.lang.annotation.Annotation">
</implements>
</class>
<class name="AnimatorRes"
  extends="java.lang.Object"
  abstract="true"
  static="false"
  final="false"
  visibility="public"
>
```

- 010 Editor

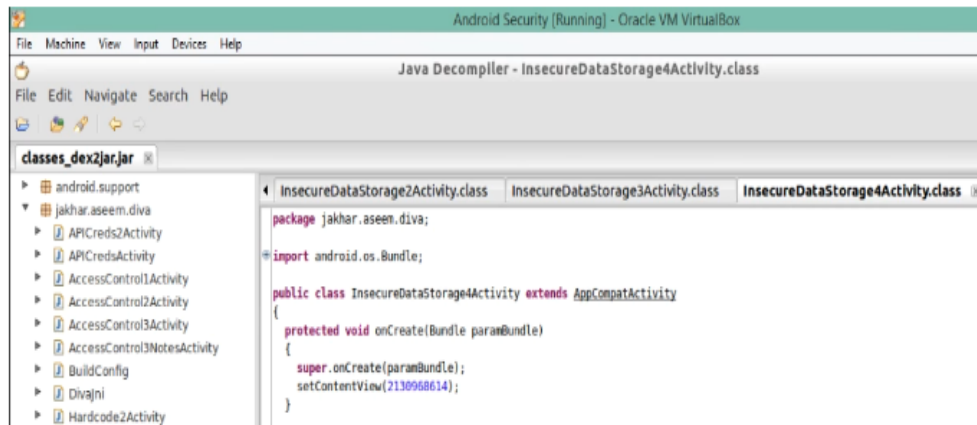
ابزاری برای خواندن و آنالیز فایل ها که از فرمت dex نیز پشتیبانی میکند

روش دوم با استفاده از کامپایل و تبدیل به فرمت jar

با استفاده از ابزار d2j-dex2jar فایل classes.dex را به فرمت جاوا تبدیل میکنیم و در نهایت با استفاده از ابزار JD-GUI فایل جاوا را آنالیز میکنیم

- Dex2jar classes.dex
- jd-gui classes_dex2jar.jar

```
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar classes.dex -> classes_dex2jar.jar
Done.
```



Owasp Mobile Top 10

استاندارد امنیتی وب اپلیکیشن

در زیر لیست 10 ریسک امنیتی موبایل طبق استاندارد OWASP نمایش داده شده است

- M1 - Improper Platform Usage
 - سواستفاده از پلتفرم یا عدم کنترل امنیتی در پلتفرم مانند TouchID , Keychain
- M2 - Insecure Data Storage
 - ذخیره سازی داده های ناامن و نشت داده های ناخواسته می شود.
- M3 - Insecure Communication
 - شامل نسخه های ضعیف SSL و ارتباط ناامن اطلاعات حساس
- M4 - Insecure Authentication
 - شامل عدم پشتیبانی از اطلاعات هویتی کاربر و ضعف در مدیریت جلسه
- M5 - Insufficient Cryptography
 - این دسته برای مسائلی است که در آن رمزگذاری انجام شد، اما به درستی انجام نشد.
- M6 - Insecure Authorization
 - این دسته مربوط به خطاهای authorization است
- M7 - Client Code Quality
 - این دسته مربوط به کنترل ورودی برای جلوگیری از آسیب پذیری های BoF و غیر می باشد
- M8 - Code Tampering
 - این دسته شامل حملات method hooking and swizzling, dynamic memory modification
- M9 - Reverse Engineering
 - این دسته شامل تجزیه و تحلیل باینری برای سورس کد، کتابخانه ها، و الگوریتم ها میباشد
- M10 - Extraneous Functionality
 - این دسته شامل اشتباهات برنامه نویسی میباشد مانند غیرفعال کردن تایید دو مرحله ای در زمان تست برنامه

Android Traffic Analysis & Interception

تجزیه و تحلیل ترافیک اندروید

تجزیه و تحلیل ترافیک اندروید به دو بخش کلی پسو و اکتیو تقسیم میشود که در زیر به جزئیات آن اشاره شده است.

تجزیه و تحلیل پسو

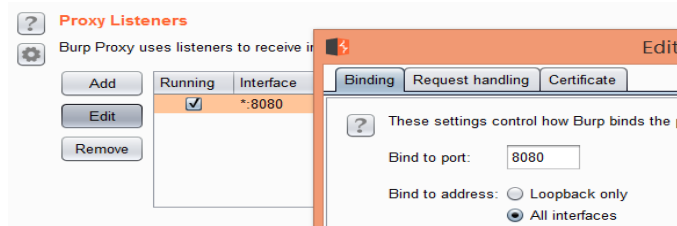
- ترافیک مخفیانه جمع شده است
- در ابتدا داده ها کپچر می شوند و بعدا آنالیز بر روی آنها انجام میشود
- استفاده از ابزارهایی مانند tcpdump

تجزیه و تحلیل اکتیو

- ترافیک به طور فعال جمع آوری شده و یا متوقف می شود
- ارتباطات به طور فعال بررسی میشود
- استفاده از ابزارهایی مانند burpsuite

برای اضافه کردن burpsuite certificate به اندروید مراحل زیر را به ترتیب انجام دهید.

- ابزار burpsuite را اجرا کنید و در قسمت proxy listeners به شکل زیر تنظیم کنید



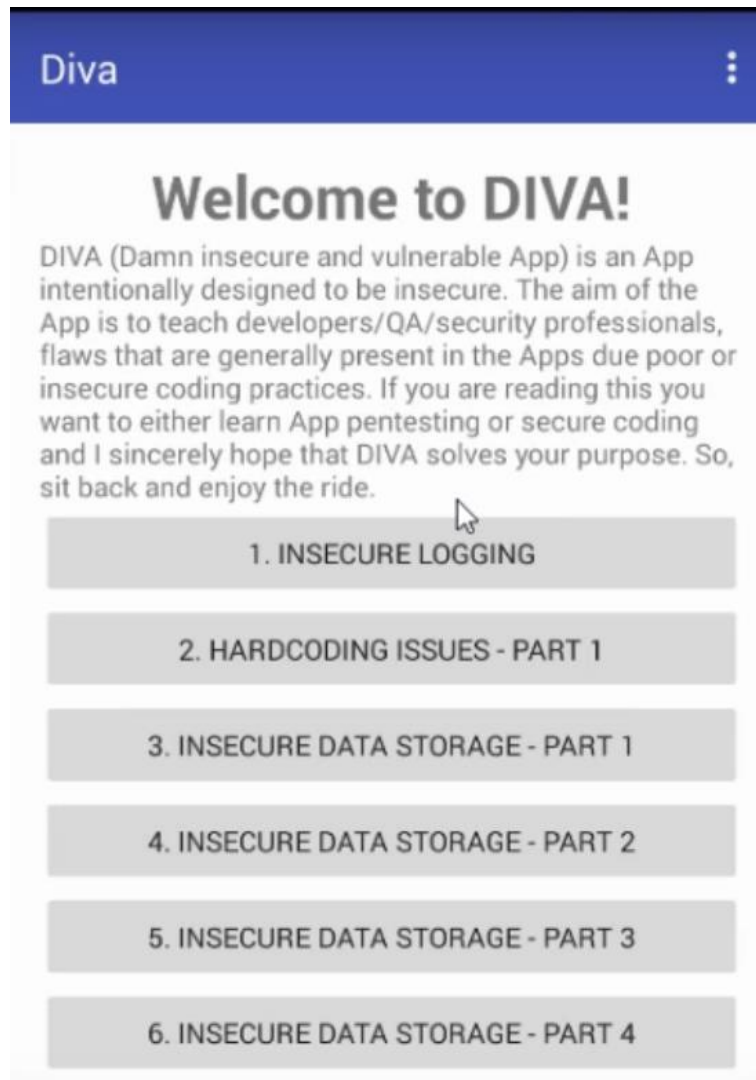
- در موبایل یا شبیه ساز اندروید، IP و Port مربوط به سیستم burpsuite را وارد کنید
- در موبایل یا شبیه ساز اندروید، مرورگر را باز کنید و وارد لینک <http://burp> شوید
- برای دریافت Cacert.der بر روی داندلود کلیک کنید
- با ابزار ADB به محیط Shell اندروید وصل شده
- وارد قسمت پوشه داندلود شود (در صورت استفاده از شبیه ساز اندروید وارد مسیر زیر شود)
 - `cd /mnt/sdcard/Download`
- برای تغییر پسوند فایل داندلود شده دستور زیر را وارد کنید
 - `mv cacert.der cacert.crt`
- در موبایل یا برنامه شبیه ساز اندروید، وارد منوی تنظیمات شده و بر روی گزینه Security کلیک کنید
- سپس بر روی گزینه install from SD card و در نهایت فایل cacert.cer را انتخاب نمایید

Damn Insecure & Vulnerable Application

نرم افزار ناامن و آسیب پذیر DIVA

نرم افزار DIVA برنامه ای است که مجموع ای از آسیب پذیری ها را دارا می باشد و برای آشنایی و تحلیل آسیب پذیری ها مناسب است که از لینک زیر قابل دریافت می باشد.

<http://payatu.com/damn-insecure-and-vulnerable-app>



Insecure Logging

آسیب پذیری ورود به سیستم

برنامه های اندروید ممکن است بعضی از داده ها را به عنوان لاگ ذخیره کند و LOGها در یه مکان مشخص و مرکزی ذخیره می شود که هر برنامه اندرویدی دیگری میتواند به آن دسترسی داشته باشد و این خطرناک است چون میتواند در لاگ ها نام کاربری و رمز عبور و کوکی و اطلاعات حساس دیگری یافت. اما در نسخه های بالاتر اندروید سطح دسترسی برای آن در نظر گرفته شده است.

در اینجا برنامه DIVA در اندروید خود اجرا کنید و PID برنامه را با دستور زیر بدست میاوریم

```
adb shell ps | grep "diva" •
```

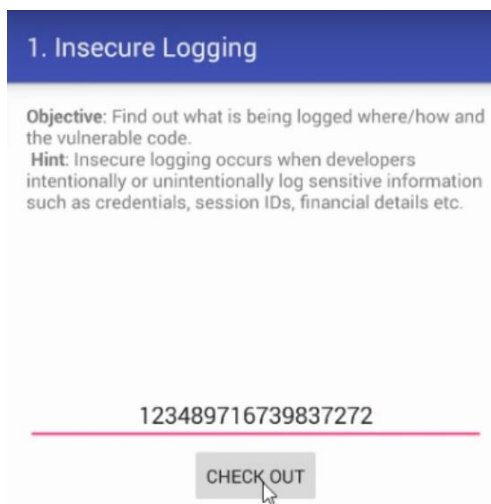
```
u0 a11 2516 324 585264 45192 ffffffff b771507b S jakhar.aseem.diva
```

در مرحله بعد اطلاعات لاگ را با دستور زیر برای برنامه diva کپچر میکنیم

```
adb logcat | grep 2516 •
```

```
W/EGL_genymotion( 2516): eglSurfaceAttrib not implemented  
I/ActivityManager( 662): START u0 {cmp=jakhar.aseem.diva/.LogActivity} from pid  
2516
```

در مرحله بعدی از داخل برنامه diva گزینه اول insecure logging را کلیک کنید و در قسمت شماره کارت، اعدادی را وارد نمایید و بر روی checkout کلیک کنید



و در نهایت شما در دو مرحله قبل که از دستور logcat استفاده کردید با عبارت زیر مواجه روبرو میشوید که شماره کارت را به عنوان لاگ ذخیره کرده است

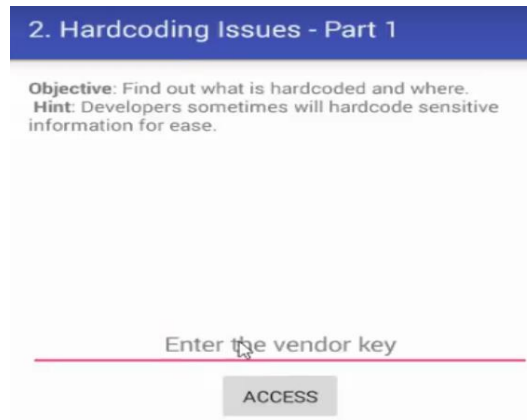
```
W/EGL_genymotion( 2516): eglSurfaceAttrib not implemented  
E/diva-log( 2516): Error while processing transaction with credit card: 123489716739837272
```

Hardcoding Issues

آسیب پذیری Hardcoding قسمت اول


گاهی اوقات برنامه نویسان به اشتباه در سورس کد برنامه های اندروید ثابت هایی را تعریف میکنند که دارای مقدارهای با ارزشی می باشند که به این گونه داده ها hardcoded گفته میشود و شامل رمزعبور، دسترسی توکن، رشته های سطح دسترسی و ... می باشد که هکر میتواند با دیکامپایل کردن برنامه اندروید به اطلاعات حساس دسترسی پیدا کند.

برای بررسی این آسیب پذیری برنامه diva را در اندروید اجرا کنید و گزینه دوم hardcoding issues part1 را انتخاب کنید



در کادر متن هر عبارتی که میخواهید وارد کنید و بروی access کلیک کنید که در نهایت پیام عدم دسترسی برای شما نمایش داده میشود.

در این مرحله برنامه diva را دیکامپایل میکنیم و سورس کد آن را بررسی میکنیم که پس از بررسی سورس با کلمه عبور vendorsecretkey در سورس برنامه به عنوان پسورد روبرو میشوید



```
classes_dex2jar.jar
├── android.support
├── jakhar.aseem.diva
│   ├── APICreds2Activity
│   ├── APICredsActivity
│   ├── AccessControl1Activity
│   ├── AccessControl2Activity
│   ├── AccessControl3Activity
│   ├── AccessControl3NotesActivity
│   ├── BuildConfig
│   ├── Divajni
│   ├── Hardcode2Activity
│   └── HardcodeActivity
│       ├── InputValidation2URISchemeAct
│       ├── InputValidation3Activity
│       ├── InsecureDataStorage1Activity
│       ├── InsecureDataStorage2Activity
│       ├── InsecureDataStorage3Activity
│       ├── InsecureDataStorage4Activity
│       └── LoginActivity
└── ...

HardcodeActivity.class
package jakhar.aseem.diva;

import android.os.Bundle;

public class HardcodeActivity extends AppCompatActivity
{
    public void access(View paramView)
    {
        if (((EditText)findViewById(2131492987)).getText().toString().equals("vendorsecretkey"))
        {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
            return;
        }
        Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
    }

    protected void onCreate(Bundle paramBundle)
    {
        super.onCreate(paramBundle);
        setContentView(2130968607);
    }
}
```

Insecure Data Storage

آسیب پذیری Insecure Data Storage

گاهی اوقات برنامه نویسان اندروید اطلاعات حساس را بدون رمزنگاری ذخیره می کنند که این موضوع می تواند خطرناک باشد. معمولاً در چهار مکان مختلف اطلاعات به صورت مشترک ذخیره میشوند که در زیر به آن اشاره شده است.

- /data/data/package name/shared_preferences
- Databases
- Temporary files
- External storage

برای بررسی آسیب پذیری فوق برنامه diva را اجرا کنید و وارد گزینه سوم insecure data storage part1 شوید و هر نام کاربری و رمز عبوری که میخواهید وارد کنید.

3. Insecure Data Storage - Par...

Objective: Find out where/how the credentials are being stored and the vulnerable code.
Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

SECRET

.....

SAVE

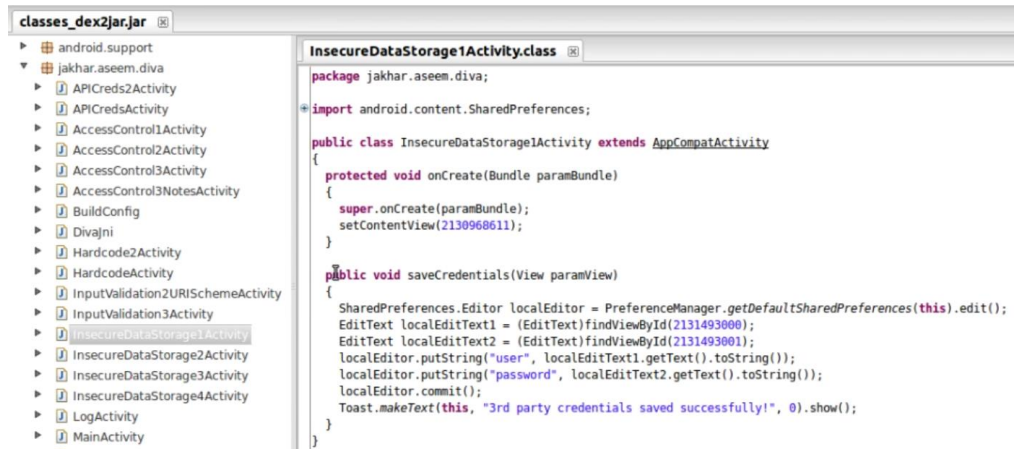
سپس با ابزار abd به اندروید متصل شوید و به مسیر زیر بروید

- `cd /data/data/jakhar.aseem.diva/shared_prefs`

در مسیر فوق میتوانید فایل xml را مشاهده کنید که شامل اطلاعات وارد شده است و با دستور cat میتوانید آن را به صورت plain مشاهده کنید که هیچگونه رمزنگاری برای انتقال اطلاعات یا ذخیره اطلاعات استفاده نشده است.

```
root@vbox86p:/data/data/jakhar.aseem.diva/shared_prefs # ls
jakhar.aseem.diva_preferences.xml
rences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="user">admin</string>
  <string name="password">secretpassword</string>
</map>
root@vbox86p:/data/data/jakhar.aseem.diva/shared_prefs #
```

اگر برنامه diva را دیکامپایل کنید، با مشاهده سورس کد متوجه plain text ذخیره شدن نام کاربری و رمزعبور خواهید شد.



```
classes_dex2jar.jar
├── android.support
└── jakhar.aseem.diva
    ├── APICreds2Activity
    ├── APICredsActivity
    ├── AccessControl1Activity
    ├── AccessControl2Activity
    ├── AccessControl3Activity
    ├── AccessControl3NotesActivity
    ├── BuildConfig
    ├── Divajni
    ├── Hardcode2Activity
    ├── HardcodeActivity
    ├── InsecureDataStorage1Activity
    ├── InsecureDataStorage2Activity
    ├── InsecureDataStorage3Activity
    ├── InsecureDataStorage4Activity
    ├── LogActivity
    └── MainActivity

InsecureDataStorage1Activity.class
package jakhar.aseem.diva;

import android.content.SharedPreferences;

public class InsecureDataStorage1Activity extends AppCompatActivity
{
    protected void onCreate(Bundle paramBundle)
    {
        super.onCreate(paramBundle);
        setContentView(2130968611);
    }

    public void saveCredentials(View paramView)
    {
        SharedPreferences.Editor localEditor = PreferenceManager.getDefaultSharedPreferences(this).edit();
        EditText localEditText1 = (EditText)findViewById(2131493806);
        EditText localEditText2 = (EditText)findViewById(2131493801);
        localEditor.putString("user", localEditText1.getText().toString());
        localEditor.putString("password", localEditText2.getText().toString());
        localEditor.commit();
        Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
    }
}
```

Database Insecure Storage

آسیب پذیری Database Insecure Storage

همانطور که قبلا گفته شد گاهی اوقات برنامه نویسان اندروید اطلاعات حساس را بدون رمزنگاری ذخیره می کنند که این موضوع می تواند خطرناک باشد. و معمولا در چهار مکان مختلف اطلاعات به صورت مشترک ذخیره میشوند که در این قسمت به ذخیره اطلاعات در پایگاه داده می پردازیم. برای بررسی آسیب پذیری فوق برنامه diva را اجرا کنید و وارد گزینه چهارم insecure data storage part2 شوید و هر نام کاربری و رمز عبوری که میخواهید وارد کنید.



سپس با ابزار abd به اندروید متصل شوید و به مسیر زیر رفته و ادامه دستورات را بزنید

- `cd /data/data/jakhar.aseem.diva/databases`
- `sqlite3 ids2`
- `.tables`
- `Select * from myuser;`

در تصویر زیر میتوانید نتیجه ی مربوطه را مشاهده کنید که شامل اطلاعات وارد شده است و با دستور select میتوانید آن را به صورت plain text مشاهده کنید که هیچگونه رمزنگاری برای انتقال اطلاعات یا ذخیره اطلاعات استفاده نشده است. نکته قابل توجه نام دیتابیس ids2 از داخل سورس کد فایل بعد از دیکامپایل بدست آمده است

```
root@vbox86p:/data/data/jakhar.aseem.diva/databases # sqlite3 ids2
ids2      ids2-journal
root@vbox86p:/data/data/jakhar.aseem.diva/databases # sqlite3 ids2
SQLite version 3.7.11 2012-03-20 11:35:50
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .tables
android_metadata  myuser
sqlite> select * from myuser;
demo|secretpassword
```


Temporary File Insecure Storage

آسیب پذیری Temporary File Insecure Storage

همانطور که قبلاً گفته شد گاهی اوقات برنامه نویسان اندروید اطلاعات حساس را بدون رمزنگاری ذخیره می کنند که این موضوع می تواند خطرناک باشد. و معمولاً در چهار مکان مختلف اطلاعات به صورت مشترک ذخیره میشوند که در این قسمت به ذخیره اطلاعات در فایل‌های موقت می پردازیم. برای بررسی آسیب پذیری فوق برنامه diva را اجرا کنید و وارد گزینه پنجم insecure data storage part3 شوید و هر نام کاربری و رمز عبوری که میخواهید وارد کنید.

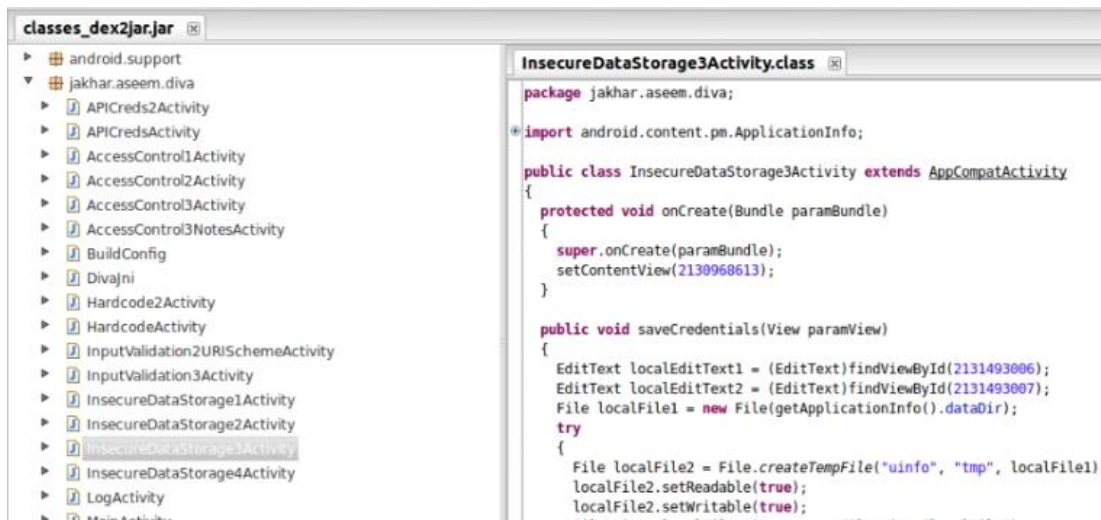
5. Insecure Data Storage - Part 3

Objective: Find out where/how the credentials are being stored and the vulnerable code.

Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.



بعد از دیکامپایل برنامه و مشاهده سورس کد متوجه می شوید که اطلاعات در فایل با نام unif0 به صورت plain text ذخیره می شود و تصویر زیر این موضوع را نشان میدهد.



```

classes_dex2jar.jar
├── android.support
├── jakhar.aseem.diva
│   ├── APICreds2Activity
│   ├── APICredsActivity
│   ├── AccessControl1Activity
│   ├── AccessControl2Activity
│   ├── AccessControl3Activity
│   ├── AccessControl3NotesActivity
│   ├── BuildConfig
│   ├── Divajni
│   ├── Hardcode2Activity
│   ├── HardcodeActivity
│   ├── InputValidation2URISchemeActivity
│   ├── InputValidation3Activity
│   ├── InsecureDataStorage1Activity
│   ├── InsecureDataStorage2Activity
│   ├── InsecureDataStorage3Activity
│   ├── InsecureDataStorage4Activity
│   ├── LogActivity
│   └── MainActivity
└── InsecureDataStorage3Activity.class
    package jakhar.aseem.diva;
    import android.content.pm.ApplicationInfo;
    public class InsecureDataStorage3Activity extends AppCompatActivity
    {
        protected void onCreate(Bundle paramBundle)
        {
            super.onCreate(paramBundle);
            setContentView(2130968613);
        }
        public void saveCredentials(View paramView)
        {
            EditText localEditText1 = (EditText)findViewById(2131493006);
            EditText localEditText2 = (EditText)findViewById(2131493007);
            File localFile1 = new File(getApplicationInfo().dataDir);
            try
            {
                File localFile2 = File.createTempFile("uinfo", "tmp", localFile1);
                localFile2.setReadable(true);
                localFile2.setWritable(true);
            }
        }
    }
    
```

سپس با ابزار abdf به اندروید متصل شوید و به مسیر زیر رفته و ادامه دستورات را بزنید

```
cd /data/data/jakhar.aseem.diva/ •  
ls •  
cat uinfo-1050553955tmp •
```

در تصویر زیر میتوانید نتیجه ی مربوطه را مشاهده کنید که فایل موقت ایجاد شده است و در نهایت با دستور cat میتوانید آن را به صورت plain text مشاهده کنید که هیچگونه رمزنگاری برای انتقال اطلاعات یا ذخیره اطلاعات استفاده نشده است.

```
root@vbox86p:/data/data/jakhar.aseem.diva # ls  
cache  
databases  
lib  
shared_prefs  
uinfo-1050553955tmp
```

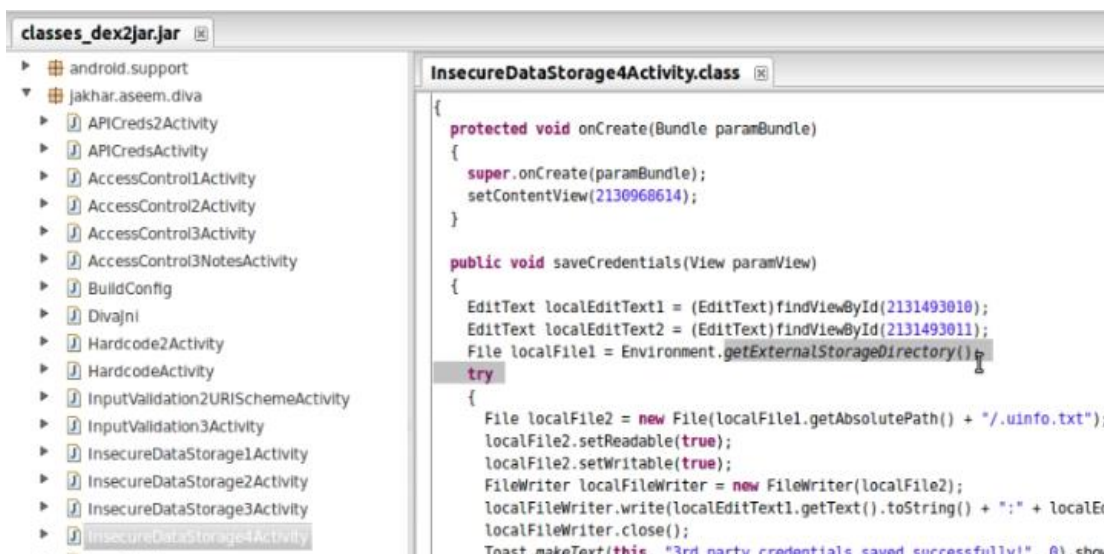
External Insecure Data Storage

آسیب پذیری External Insecure Data Storage

همانطور که قبلا گفته شد گاهی اوقات برنامه نویسان اندروید اطلاعات حساس را بدون رمزنگاری ذخیره می کنند که این موضوع می تواند خطرناک باشد. و معمولا در چهار مکان مختلف اطلاعات به صورت مشترک ذخیره میشوند که در این قسمت به آخرین محل ذخیره سازی یعنی ذخیره اطلاعات در حافظه خارجی می پردازیم. برای بررسی آسیب پذیری فوق برنامه diva را اجرا کنید و وارد گزینه ششم insecure data storage part4 شوید و هر نام کاربری و رمز عبوری که میخواهید وارد کنید.



بعد از دیکامپایل برنامه و مشاهده سورس کد متوجه می شوید که اطلاعات در حافظه خارجی داخل فایلی با نام unifo.txt به صورت plain text ذخیره می شود که تصویر زیر این موضوع را نشان میدهد.



```

classes_dex2jar.jar
├── android.support
├── jakhar.aseem.diva
│   ├── APICreds2Activity
│   ├── APICredsActivity
│   ├── AccessControl1Activity
│   ├── AccessControl2Activity
│   ├── AccessControl3Activity
│   ├── AccessControl3NotesActivity
│   ├── BuildConfig
│   ├── Divajni
│   ├── Hardcode2Activity
│   ├── HardcodeActivity
│   ├── InputValidation2URISchemeActivity
│   ├── InputValidation3Activity
│   ├── InsecureDataStorage1Activity
│   ├── InsecureDataStorage2Activity
│   ├── InsecureDataStorage3Activity
│   └── InsecureDataStorage4Activity
└── ...
    
```

```

InsecureDataStorage4Activity.class
{
    protected void onCreate(Bundle paramBundle)
    {
        super.onCreate(paramBundle);
        setContentView(2130968614);
    }

    public void saveCredentials(View paramView)
    {
        EditText localEditText1 = (EditText)findViewById(2131493010);
        EditText localEditText2 = (EditText)findViewById(2131493011);
        File localFile1 = Environment.getExternalStorageDirectory();
        try
        {
            File localFile2 = new File(localFile1.getAbsolutePath() + "/.unifo.txt");
            localFile2.setReadable(true);
            localFile2.setWritable(true);
            FileWriter localFileWriter = new FileWriter(localFile2);
            localFileWriter.write(localEditText1.getText().toString() + ":" + localEdit
            localFileWriter.close();
            Toast.makeText(this, "3rd party credentials saved successfully!", 0).show(
    
```

سپس با ابزار abd به اندروید متصل شوید و به مسیر زیر رفته و ادامه دستورات را بزنید

- `cd /mnt/sdcard`
- `ls -a`
- `cat .uinfo.txt`

در تصویر زیر میتوانید نتیجه ی مربوطه را مشاهده کنید که فایل داخل حافظه ی خارجی ایجاد شده است و در نهایت با دستور cat میتوانید آن را به صورت plaintext مشاهده کنید که هیچگونه رمزنگاری برای انتقال اطلاعات یا ذخیره اطلاعات استفاده نشده است.

```
root@vbox86p:/mnt/sdcard # ls -a
.uinfo.txt
Alarms
Android
DCIM
Download
Movies
Music
Notifications
Pictures
Podcasts
Ringtones
Statements_dinesh.html
storage
root@vbox86p:/mnt/sdcard # cat .uinfo.txt
demo:password
```

SQL Injection (Input Validation)

آسیب پذیری (SQL Injection (Input Validation)

آسیب پذیری SQLi جز خطرناکترین آسیب پذیری ها به شمار می رود و زمانی رخ میدهد که برنامه نمیتواند ورودی را پاکسازی کند که در نتیجه منجر به حمله به سرور از سمت کلاینت می شود.

برای بررسی این موضوع از برنامه diva گزینه ی هفتم input validation issues – part1 استفاده میکنیم که قسمتی مربوط به جستجوی کاربر وجود دارد و فیلد مربوطه به SQLi آسیب پذیر می باشد که هکر میتواند دستورات مخرب خود را وارد کند.

7. Input Validation Issues - Part 1

Objective: Try to access all user data without knowing any user name. There are three users by default and your task is to output data of all the three users with a single malicious search.

Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. For ease of testing there are three users already present in the database, for example one of them is admin, you can try searching for admin to test the output.

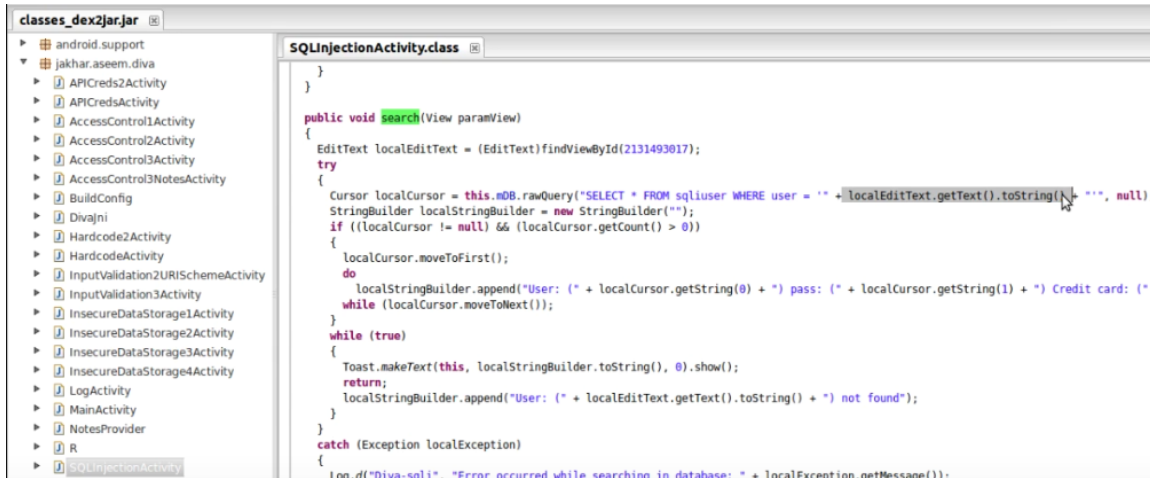
Enter user name to search

SEARCH

در زیر تعدادی از دستورات برای حمله ی SQLi آورده شده است:

- 1'or'1'='1'
- Admin' or '1'='1'
- Admin' or '1'='1'--
- Admin' or '1'='1'#
- Admin' or '1'='1'/*
- Admin' or 1=1 or '='
- Admin' or 1=1
- Admin' or 1=1--
- Admin' or 1=1#
- Admin' or 1=1/*
- Admin') or ('1'='1

بعد از دیکامپایل برنامه و مشاهده سورس کد متوجه می شوید که ورودی کاربر بدون هیچ اعتبارسنجی دریافت می شود و هکر میتواند با نمونه پیلودهایی که در بالا به آن اشاره شد کوئری مخرب خود را وارد کند و اطلاعات دیگری را از پایگاه داده فراخوانی کند. که تصویر زیر این موضوع را نشان میدهد.



```

SQLInjectionActivity.class
}
}
public void search(View paramView)
{
    EditText localEditText = (EditText)findViewById(2131493017);
    try
    {
        Cursor localCursor = this.mDB.rawQuery("SELECT * FROM sqluser WHERE user = '" + localEditText.getText().toString() + "'", null);
        StringBuilder localStringBuilder = new StringBuilder("");
        if ((localCursor != null) && (localCursor.getCount() > 0))
        {
            localCursor.moveToFirst();
            do
            {
                localStringBuilder.append("User: (" + localCursor.getString(0) + ") pass: (" + localCursor.getString(1) + ") Credit card: (" + localCursor.getString(2) + ")");
                while (localCursor.moveToNext());
            }
            while (true)
            {
                Toast.makeText(this, localStringBuilder.toString(), 0).show();
                return;
                localStringBuilder.append("User: (" + localEditText.getText().toString() + ") not found");
            }
        }
    }
    catch (Exception localException)
    {
        Log.d("Diva-sali", "Error occurred while searching in database: " + localException.getMessage());
    }
}
    
```

در تصویر زیر هکر به پایگاه داده میگوید اطلاعاتی را در خروجی چاپ کن که یکی از دو شرط زیر برقرار باشد.

- آیا نام کاربری diva در پایگاه داده وجود دارد؟
- آیا $1 = 1$ است؟

چون شرط دوم همیشه درست می باشد پس دستور مخرب هکر تمامی اطلاعات پایگاه داده را نمایش میدهد.

7. Input Validation Issues - Part 1

Objective: Try to access all user data without knowing any user name. There are three users by default and your task is to output data of all the three users with a single malicious search.

Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. For ease of testing there are three users already present in the database, for example one of them is admin, you can try searching for admin to test the output.

`diva'or'1='1`

```

User: (admin) pass: (passwd123) Credit card:
(1234567812345678)
User: (diva) pass: (p@ssword) Credit card:
(1111222233334444)
User: (john) pass: (password123) Credit card:
(5555666677778888)
    
```

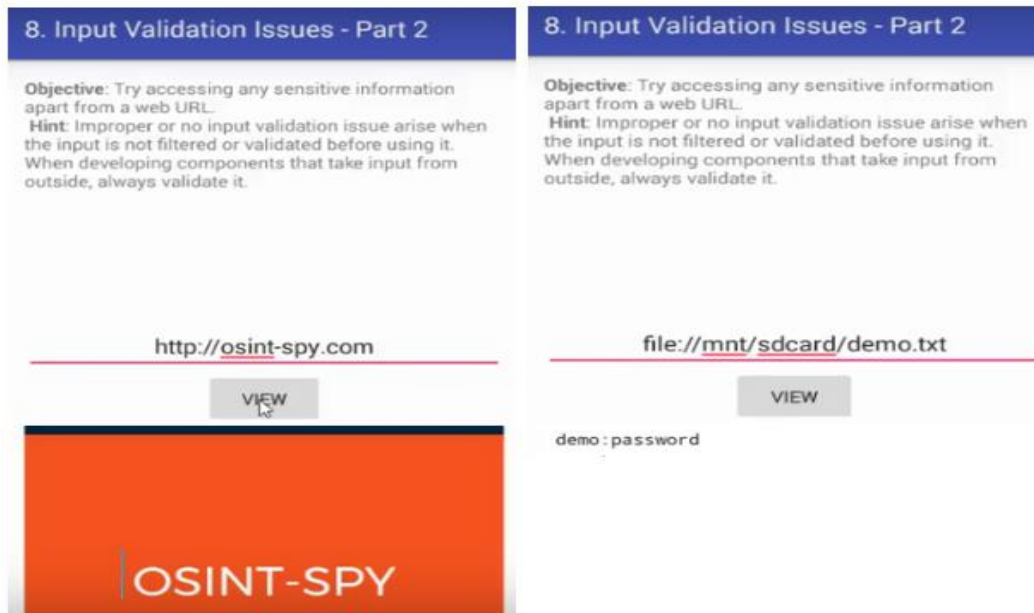
Abuse Web View

آسیب پذیری های Web View

در واقع web view یک مرورگر قابل حمل کوچک است که در اپلیکیشن ما وجود دارد و کار نمایش صفحات وب را انجام میدهد و پروتکول هایی که توسط مرورگر پشتیبانی می شوند به شرح زیر است

- http
- https
- ftp
- file
- smtp

توسط برنامه ی Diva سناریو مربوط به این آسیب پذیری را اجرا می کنیم که برای این منظور وارد گزینه هشتم برنامه – part2 – input validation issues میشویم، که در کادر متن برای نمایش وبسایت میبایست آدرس وبسایت وارد شود ولی هکر قصد دارد با توجه به پشتیبانی مرورگر از پروتکول file محتویات یکی از فایل های داخل اندروید را مشاهده کند. که دو تصویر زیر این موضوع را نشان می دهد.



Access Control

آسیب پذیری Access Control

آسیب پذیری کنترل دسترسی زمانی رخ میدهد که برنامه کاربر را نتواند تایید کند و بررسی مجوز کاربر با اختلال مواجه شده است. و در نهایت هکر با همین مجوز میتواند منابع دیگر را بررسی و سواستفاده کند. به طور مثال شما برنامه ای دارید که اطلاعات حساس را در خود دارد و با این آسیب پذیری میتوانید به اطلاعات حساس آن برنامه دسترسی پیدا کنید.

برای این منظور سناریویی را با نرم افزار diva انجام داده می شود که میبایست وارد گزینه ی نهم access control issues part 1 شوید. و زمانی که بر روی دکمه ی view api کلیک میکنید اطلاعات مربوط به کاربری که حق دسترسی دارد را نمایش داده میشود.



در این سناریو ما قصد داریم بدون وارد شدن به برنامه، به اطلاعات کاربر دسترسی پیدا کنیم. این عملیات توسط یک اکتیویتی انجام میشود و برای بررسی آن باید اول برنامه را دیکامپایل کرد و وارد فایل Manifest.xml شد که نام اکتیویتی مربوطه پیدا میشود و اکتیویتی مربوطه توسط init-filter استفاده میشود که در قسمت action نمایش اطلاعات credential برای آن تنظیم شده است که در واقع این قسمت مسئول نمایش اطلاعات کاربر می باشد.

```
<activity android:label="@string/d9" android:name="jakhar.aseem.diva.AccessControll1Activity" />
<activity android:label="@string/apic_label" android:name="jakhar.aseem.diva.APICredsActivity">
  <intent-filter>
    <action android:name="jakhar.aseem.diva.action.VIEW_CREDS" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```


حال از برنامه diva خارج شده و با ابزار adb به اندروید متصل شوید و توسط activity manager برای نمایش اطلاعات کاربر، اکتیویتهی مربوطه را اجرا می کنیم.

```
adb shell •  
am start -a jakhar.aseem.diva.action.VIEW_CREDS •
```

تصویر نتیجه خروجی دستورات بالا رو نمایش میدهد.

```
root@vbox86p:/ # am start -a jakhar.aseem.diva.action.VIEW_CREDS  
Starting: Intent { act=jakhar.aseem.diva.action.VIEW_CREDS }
```

Vendor API Credentials

```
API Key: 123secretapikey123  
API User name: diva  
API Password: p@ssword
```

Authentication Based Access Control Issues

آسیب پذیری Authentication Based Access Control

همانند مبحث قبلی میخواهیم اطلاعات مربوط به کاربر را بدون وارد شدن به برنامه مشاهده کنیم ولی تفاوتی که وجود دارد در این است که کاربر برای ثبتنام نیاز به pin-code میباشد که از طرف وبسایت به آن داده میشود. که در سناریو زیر ما میخواهیم pic-code را بایمس کنیم. در ابتدا وارد برنامه diva شده و گزینه ی دهم access control issues – part 2 را انتخاب میکنیم

10. Access Control Issues - Part 2

Objective: You are able to access the Third Party app TVEETER API credentials after you have registered with Tveeter. The App requests you to register online and the vendor gives you a pin, which you can use to register with the app. Now, try to access the API credentials from outside the app without knowing the PIN. This is a business logic problem so you may need to see the code.

Hint: Components of an app can be accessed from other apps or users if they are not properly protected and some may also accept external inputs. Components such as activities, services, content providers are prone to this.

Register Now. Already Registered.

VIEW TVEETER API CREDENTIALS

پس از بررسی فایل Manifest.xml نام اکتیویتهی مربوطه را پیدا کرده و سپس سورس فایل جاوا مربوط به اکتیویتهی را باز میکنیم و دنبال متغیر چک کردن pin-code میگردیم

```
<activity android:label="@string/d10" android:name="jakhar.aseem.diva.AccessControl2Activity" />
<activity android:label="@string/apic2_label" android:name="jakhar.aseem.diva.APICredits2Activity">
  <intent-filter>
    <action android:name="jakhar.aseem.diva.action.VIEW_CREDS2" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```

```

import android.util.Log;
import android.view.View;
import android.widget.RadioButton;
import android.widget.Toast;

public class AccessControl2Activity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView((int) R.layout.activity_access_control2);
    }

    public void viewAPICredentials(View view) {
        RadioButton rbregnow = (RadioButton) findViewById(R.id.aci2rbregnow);
        Intent i = new Intent();
        boolean chk_pin = rbregnow.isChecked();
        i.setAction("jakhar.aseem.diva.action.VIEW_CREDS2");
        i.putExtra(getString(R.string.chk_pin), chk_pin);
        if (i.resolveActivity(getPackageManager()) != null) {
            startActivity(i);
            return;
        }
        Toast.makeText(this, "Error while getting Tveeter API details", 0).show();
        Log.e("Diva-ac11", "Couldn't resolve the Intent VIEW_CREDS2 to our activity");
    }
}
    
```

با بررسی دقیق تر کد متوجه میشوید که نوع متغیر Boolean بوده و مقدار true یا false میگیرد که در سورس کد آن با true استفاده شده است و ما میبایست اکتیویته را به طور مستقیم و غیرفعال بودن chk_pin اجرا کنیم، برای این منظور از سویچ ez- در activity manager استفاده میکنیم.

حال از برنامه diva خارج شده و از طریق adb دستور زیر را وارد کنید:

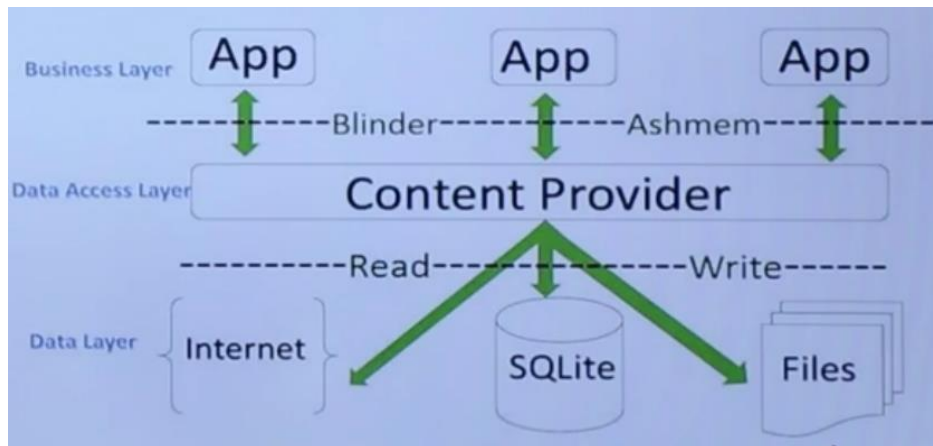
- adb shell
- am start -a jakhar.aseem.diva.action.VIEW_CREDS2 --ez "check_pin" false



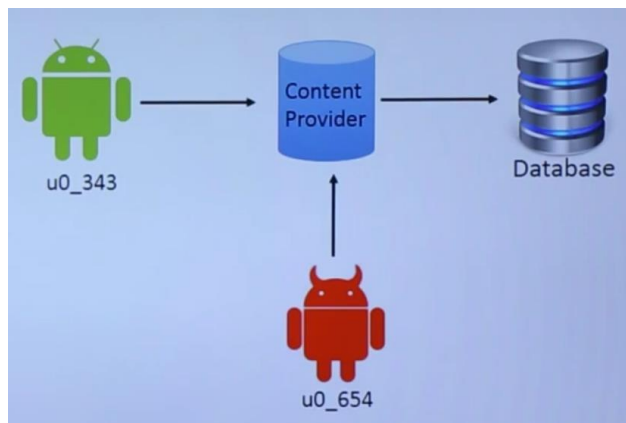
Leaking Content Provider

آسیب پذیری Leaking Content Provider

در واقع content providerها برای ذخیره سازی و کوئری زدن داده در اپلیکیشن استفاده میشود که قابلیت read/write را دارند و تمام content providerها یک URI یونیک دارند که با content:// شروع میشوند.



در سناریویی که قرار است توسط برنامه diva اجرا شود، هکر به طور مستقیم به لایه میانی یعنی content provider حمله میکند و از طریق آن به پایگاه داده حمله میکند و دستورات مخرب خود را وارد میکند. برای این منظور از برنامه diva گزینه یازدهم - part3 access control issues را انتخاب میکنیم. در کادر اول یک پین 4 رقمی باید بسازیم که برای مثال پین کد 1234 را وارد میکنیم و بر روی ساخت پین کلیک کنید و بعد از آن بر روی دکمه ی go to private notes کلیک کرده که از شما پین کد وارد شده را میخواهد و مجدد 1234 را وارد میکنیم



Diva Private Notes	
Exercise	Alternate days running
Expense	Spent too much on home theater
Weekend	b333333333333r
holiday	Either Goa or Amsterdam
home	Buy toys for baby, Order dinner
office	10 Meetings, 5 Calls, Lunch with CEO

حال ما می‌خواهیم بدون وارد شدن به برنامه اطلاعات مربوط به private note را مشاهده کنیم. در ابتدا ما می‌بایست URI مربوط به این content provider را پیدا کنیم بعد از آن می‌توانیم کوئری‌های دلخواه خود را وارد کنیم برای این منظور ما نیاز داریم فایل manifest.xml را تجزیه و تحلیل کنیم. و اکتیویتهی مربوطه (NoteProvider.java) را یافت کرده سپس فایل را برای یافتن URI بررسی می‌کنیم.

```
<provider android:name="jakhar.aseem.diva.NotesProvider" android:enabled="true" android:exported="true"
android:authorities="jakhar.aseem.diva.provider.notesprovider" />
<activity android:label="@string/d11" android:name="jakhar.aseem.diva.AccessControl3Activity" />
<activity android:label="@string/d12" android:name="jakhar.aseem.diva.Hardcode2Activity" />
<activity android:label="@string/pnotes" android:name="jakhar.aseem.diva.AccessControl3NotesActivity" />
<activity android:label="@string/d13" android:name="jakhar.aseem.diva.InputValidation3Activity" />
</application>
```

```
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;

public class NotesProvider extends ContentProvider {
    static final String AUTHORITY = "jakhar.aseem.diva.provider.notesprovider";
    static final Uri CONTENT_URI = Uri.parse("content://jakhar.aseem.diva.provider.notesprovider/notes");
    static final String CREATE_TBL_QRY = "CREATE TABLE notes (_id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT
NOT NULL, note TEXT NOT NULL);";
    static final String C_ID = "_id";
    static final String C_NOTE = "note";
    static final String C_TITLE = "title";
```

حال توسط ADB برای نمایش تمام اطلاعات private note دستورات زیر را وارد کنید.

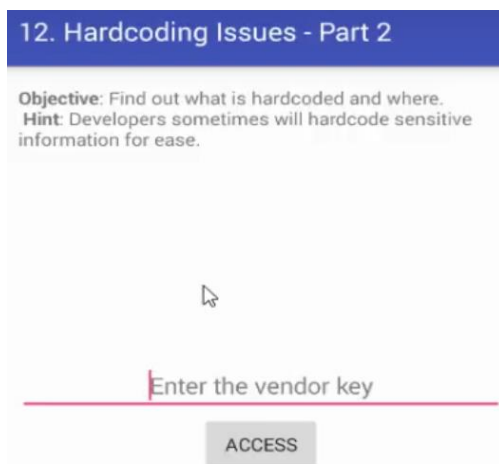
- adb shell
- content query -uri content://jakhar.aseem.diva.notesprovider/notes

```
Row: 0 _id=5, title=Exercise, note=Alternate days running
Row: 1 _id=4, title=Expense, note=Spent too much on home theater
Row: 2 _id=6, title=Weekend, note=b3333333333333r
Row: 3 _id=3, title=holiday, note=Either Goa or Amsterdam
Row: 4 _id=2, title=home, note=Buy toys for baby, Order dinner
Row: 5 _id=1, title=office, note=10 Meetings. 5 Calls. Lunch with CEO
```

Hardcoding Issues Using JNI

آسیب پذیری Hardcoding Issues Using JNI

JNI یک چارچوب برنامه نویسی است که کد جاوا را در JVM فعال می کند تا با دیگر کتابخانه ها ارتباط برقرار کند، همچنین زبان های برنامه نویسی دیگر مثل C, C++ هم میتوانند با کد جاوا از طریق JNI در ارتباط باشند. برای این منظور سناریویی را توسط برنامه diva انجام میدهیم که بعد از وارد شدن به برنامه گزینه دوازدهم hardcoding issues – part2 را انتخاب کنید و همانطور که قبلا گفته شد گاهی اوقات برنامه نویسان به اشتباه در سورس کد برنامه های اندروید ثابت هایی را تعریف میکنند که دارای مقدرهای با ارزشی می باشند که به این گونه داده ها hardcoded گفته میشود.



پس در این مرحله همانند مباحث قبلی ما نیاز داریم سورس کد را بررسی کنیم، پس سورس کد جاوا hardcode2Activity.java را باز میکنیم که با دقت به سورس متوجه میشویم که از JNI برای ارتباط با دیگر زبان های برنامه استفاده شده است.

```
package jakhar.aseem.diva;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class Hardcode2Activity extends AppCompatActivity {
    private DivaJni djni;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView((int) R.layout.activity_hardcode2);
        this.djni = new DivaJni();
    }

    public void access(View view) {
        if (this.djni.access(((EditText) findViewById(R.id.hc2Key)).getText().toString()) != 0) {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
        } else {
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }
}
```

پس برای بررسی JNI از طریق ADB به مسیر زیر رفته و دستورات را وارد کنید

- adb shell
- cd /data/data/jakhar.aseem.diva/lib

```
root@vbox86p:/data/data/jakhar.aseem.diva # ls
app_webview
cache
databases
lib
shared_prefs
test.txt
root@vbox86p:/data/data/jakhar.aseem.diva # cd lib/
root@vbox86p:/data/data/jakhar.aseem.diva/lib # ls
libdivajni.so
root@vbox86p:/data/data/jakhar.aseem.diva/lib #
```

برای خواندن فایل JNI باید از دستور strings استفاده کنیم و در نهایت میتوانیم از محتویات آن برای یافتن رمزعبور استفاده کرد.

- strings libdivajni.so



The image shows a split-screen view. On the left is an ADB terminal window with the following output:

```
__stack_chk_fail
Java_jakhar_aseem_diva_DivaJni_access
Java_jakhar_aseem_diva_DivaJni_initiateLaunchS
strcpy
JNI_OnLoad
edata
_bss_start
end
libstdc++.so
libm.so
libc.so
libdl.so
libdivajni.so
d$0[
olsdfgad;lh
.dotdot
;*2$"
GCC: (GNU) 4.8
gold 1.11
.shstrtab
.dynsym
.dynstr
```

On the right is a web interface titled "12. Hardcoding Issues - Part 2". It contains the following text:

Objective: Find out what is hardcoded and where.
Hint: Developers sometimes will hardcode sensitive information for ease.

Below this text is a search bar containing the string "olsdfgad;lh". A red horizontal line is drawn across the search bar. Below the search bar is a button labeled "ACCESS". Below the button is a small icon of a person. At the bottom of the interface is a message box that says "Access granted! See you on the other side :)"

DOS Attack in Android

DOS Attack in Android آسیب پذیری

خطای حافظه زمانی رخ میدهد که ورودی کاربر بیش از حافظه اختصاص داده شده باشد که خطای غیرمنتظره ای در برنامه اتفاق می افتد و آن قطعه از حافظه مجدد بازنویسی میشود که اصطلاحاً سرریز بافر میگویند و در نهایت منجر به حمله ی DOS می شود.

برای این منظور توسط برنامه diva سناریویی طراحی میشود که منجر به DoS یا crash برنامه میشود. پس باز کردن برنامه وارد گزینه سیزدهم 3 part – input validation issues شوید در کادر ورودی هر عبارتی که میخواهید وارد کنید و بر روی دکمه کلیک کنید، سپس به تعداد کاراکترها اضافه کنید تا برنامه از با خطا مواجه شود. دو تصویر زیر سناریوی فوق را نشان میدهد.



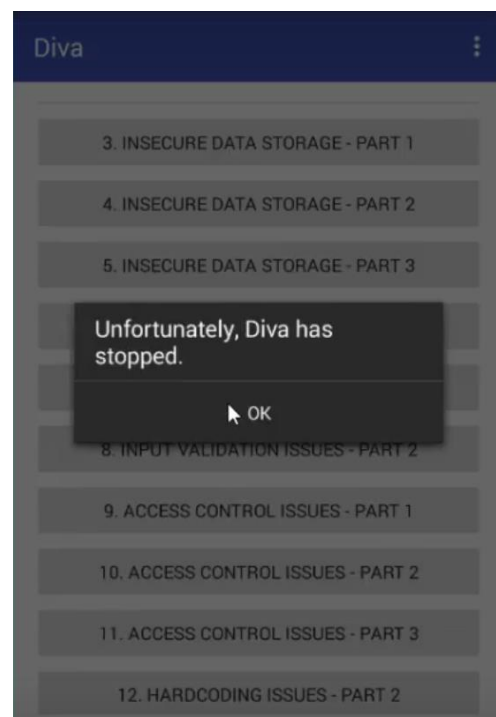
13. Input Validation Issues - Part 3

Objective: This is a Missile Launch App. Spread love not War! DOS the Damn thing! Your objective here is to NOT find the code and then launch the missiles, rather it is to crash the app (and then find the root cause the crash).

Hint: Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. This is a classic memory corruption vulnerability. If you can get code execution, I would love to hear from you. I dont expect anyone to go that far though.

launch123

PUSH THE RED BUTTON



Divia

3. INSECURE DATA STORAGE - PART 1

4. INSECURE DATA STORAGE - PART 2

5. INSECURE DATA STORAGE - PART 3

Unfortunately, Diva has stopped.

OK

8. INPUT VALIDATION ISSUES - PART 2

9. ACCESS CONTROL ISSUES - PART 1

10. ACCESS CONTROL ISSUES - PART 2

11. ACCESS CONTROL ISSUES - PART 3

12. HARDCODING ISSUES - PART 2

Drozer Security Framework

فریم ورک امنیتی Drozer

یک فریم ورک جامع امنیتی برای تست نفوذ اندروید می باشد که میتوان تست های آسیب پذیری مختلفی بر روی برنامه های اندروید را انجام داد که از لینک زیر قابل دریافت است. که میبایست نسخه سرور و نسخه agent آن را دریافت کنید.

<https://labs.mwrinfosecurity.com/tools/drozer>

برای استفاده از ابزار امنیتی Drozer مراحل را به ترتیب زیر طی کنید.

- Install Drozer server
- Install Drozer agent (بعد از نصب اجرا نشود)
- adb forward tcp: 31415 tcp:31415
- run agent Drozer (اجرا نسخه موبایل Drozer)
- drozer console connect (اجرای نسخه سرور)

دستورات پر کاربرد

دستور	توضیحات
ls	نمایش تمام دستورات drozer
run app.package.list	نمایش تمام برنامه های نصب شده در اندروید
run app.package.attacksurface [package name]	نمایش اطلاعات کلی در مورد content provider, activity, services, broadcast
run app.package.info [package name]	نمایش اطلاعات مانند نسخه و نام و... در مورد برنامه
run app.activity.info [package name]	نمایش activity و exported های آن
run scanner.provider.finduris -a [package name]	نمایش URI های برنامه مورد نظر
run scanner.provider.injection -a [package name]	تست آسیب پذیری injection بر روی برنامه
run scanner.provider.traversal -a [package name]	تست آسیب پذیری traversal بر روی برنامه

```

Selecting 17c9544deb31c1b8 (Genymotion Custom Phone - 4.4.4 - API 19 - 768x1280
4.4.4)
..
..O.. ..F..
..a.. ..nd
..ro..idsnemesisan..pr
..otectorandroidsne.
..sisandprotectorandroids+
..nemesisanprotectorandroidsn:
..emesisanprotectorandroidsnemes..
..isandp...rotectorandro...idsnem.
..isisandp...rotectorandroid...snemis.
..andprotectorandroidsnemisandprotec.
..torandroidsnemesisanprotectorandroid.
..snemisisanprotectorandroidsnemis:
..dprotectorandroidsnemesisanprotector.
I
drozer Console (v2.3.3)
dz>
    
```