Ders 4 - Insecure Data Storage > Part 2

Dersin Hedefi

Diva uygulamasındaki “4. Insecure Data Storage - Part 2” seçeneğine tıklayın. Bu seçenek ile açılan kayıt sayfasında üçüncü taraf servise kaydolmak için girilen kullanıcı adı ve parola bilgilerinin nerede ve ne şekilde depolandığını keşfedin. Ardından bu kayıt sayfasındaki girilen hesap bilgilerini güvensiz depolayan kod satırlarını belirtin. Not: Bir önceki makaleye göre bu sefer farklı bir yöntemle üçüncü taraf servise kayıt bilgileri depolanmaktadır.

Dersin Açıklaması

*Uyarı: Bir önceki Insecure Data Storage > Part 1 makalesini okuduysanız bu makaledeki **Dersin Açıklaması** başlığını atlayabilirsiniz. Çünkü bir önceki makaledeki aynı bilgiler bu başlık altında yer almaktadır.*

Android uygulamalarda android uygulama verilerini depolama ikiye ayrılmaktadır: Yerel veri depolama ve uzak veri depolama. Yerel veri depolama birbirinden amaç ve kapsam açısından farklılık arzeden birçok tekniği barındırmaktadır. Uzak veri depolama ise uzakta çalışan bulut veritabanının teknolojisine göre değişiklik gösterir.

Bu “Insecure Data Storage > Part 1-4” (Güvensiz Veri Depolama > Part 1-4) mini serisinde yerel veri depolama yöntemleri arasından birbirinden farklı, fakat diva zafiyetli mobil uygulamasında üçüncü taraf bir servise kayıt olma sayfasındaki girilen bilgileri (kullanıcı adı ve şifre bilgilerini) depolama noktasında ortak işlev görecek dört farklı yöntem üzerinden gidilecektir. Bunlar; “Shared Preferences” (Paylaşımlı Tercihler), “SQLite Database” (SQLite İlişkisel Veritabanı), “Internal File Storage” (Dahili Dosya Depolama) ve “External File Storage” (Harici Dosya Depolama) şeklindedir. Bu yerel depolama yöntemlerine ilave olarak “Saved Instance State”, “Internal Cache Files”, “Realm Database”,... şeklinde listeye eklemeler yapılabilir. Fakat zafiyetli diva uygulamamızı ilgilendirmediğinden başka yöntemlere girilmeyecektir.

Yerel depolama yöntemi SharedPreferences (Paylaşımlı Tercihler) uygulama ayar / girdi / … tercihlerinin saklanmasında kullanılır. SQLite uygulamadaki kullanıcı taraflı girdilerin / verilerin veritabanı teknolojisi ile depolanmasında kullanılır. Internal File Storage uygulamadaki kullanıcı taraflı girdilerin / verilerin cihazın dahili depolama ünitesinde metin dosyası halinde depolanmasında kullanılır. External File Storage uygulamadaki kullanıcı taraflı girdilerin / verilerin cihazın harici depolama ünitesinde metin dosyası halinde depolanmasında kullanılır.

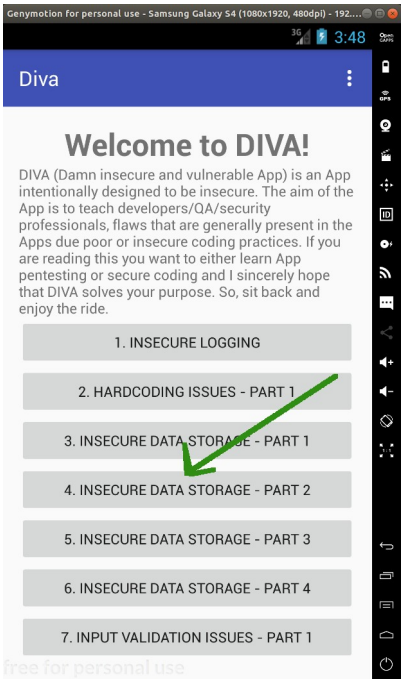
Android cihazlarda yerel veri depolama yöntemleri birbirlerinden **amaç ve kapsam** olarak farklıdırlar. Örneğin bu mini seride bahsedeceğimiz “Shared Preferences” tekniği sadece string, float, integer verileri depolar (çünkü amacı kullanıcı uygulama tercihlerini depolamaktır ve bu nedenle xml formatında veri depolar), resim-ses-video gibi verileri depolamaz, örneğin bu mini seride değinmeyeceğimiz “Saved Instance State” yerel depolama yöntemi sadece uygulama nesnelerinde yapılan değişiklikleri anlık kaydeder ve kazara sayfada geri gitme gibi durum olduğunda mevcut durumun / verilerin korunurluğunu sürdürür, veya bu mini seride değinmeyeceğimiz bir diğer yerel depolama yöntemi “Internal Cache Files” sadece kısıtlı süreli veri tutmaya yarar (çünkü amacı cihaz depolamasını verimli kullanmaktır)... gibi.

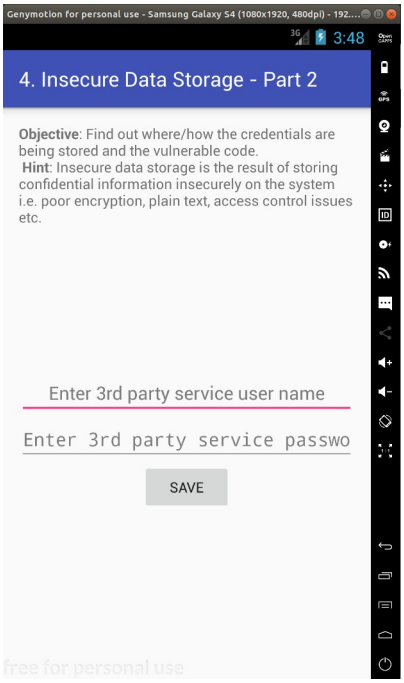
Bu güvensiz veri depolama mini (part 1 - 4) serisinde dört farklı yerel depolama yönteminin yanlış şekilde kullanımı nedeniyle kullanıcı hassas verilerinin (diva uygulamasındaki üçüncü taraf servis kullanıcı hesap bilgilerinin) ele geçirilebileceği gösterilecektir.

Dersin Çözümü

Uyarı: Bir önceki Insecure Data Storage > Part 1 makalesini okuduysanız bu makaledeki ***Dersin Çözümü*** *başlığı çoğunlukla aynı metne sahiptir. Ancak diva uygulamasındaki Insecure Data Storage > Part 2 dersinin sunduğu metot gereği bir noktada ayrışmaktadır.*

Diva uygulamasının "4. Insecure Data storage - Part 2" sayfasında senaryo gereği bizi uygulama içi üçüncü taraf bir servise kaydolma sayfası karşılamaktadır. Bu kayıt işlemi ile diva uygulaması içerisinden üçüncü taraf bir servisin işlevlerini / özelliklerini kullanabilir hale geleceğiz.

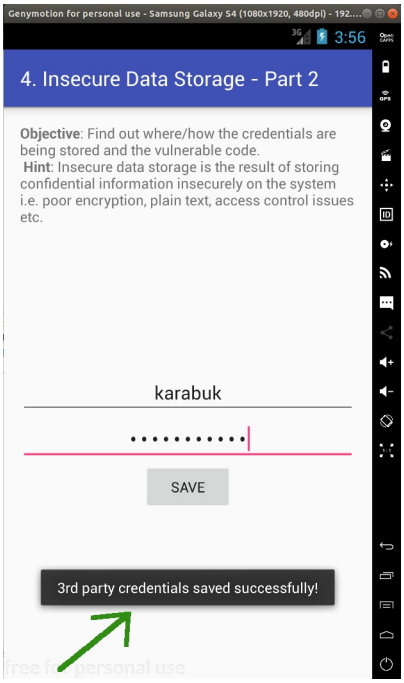




Bu sayfaya gireceğimiz kayıt bilgileri normal şartlarda uzak sunucuda kayıt altına alınacaktır. Fakat ders sayfasındaki senaryoya göre uygulamayı başka zamanlarda kullanırken üçüncü taraf servise her defasında elle giriş yapmayalım diye kayıtlı hesabımızın mobil cihazda yerel olarak depolanması söz konusudur. Bu şekilde diva uygulamasını kullanırken üçüncü taraf servise her daim bağlı halde kalmış olacağız ve uygulamayı kullanırken her daim üçüncü taraf servisin işlevlerini / özelliklerini kullanabilir halde olacağız.

Bizim amacımız ise yerel android sistemde güvensiz şekilde depolanan üçüncü taraf servis hesabımızın nerede ve ne şekilde depolandığını tespit etmektir.

Şimdi uygulama sayfasında üçüncü taraf servise kayıt olmak için bir kullanıcı adı ve şifre girelim;
Kullanıcı adı: karabuk, Şifre: password123



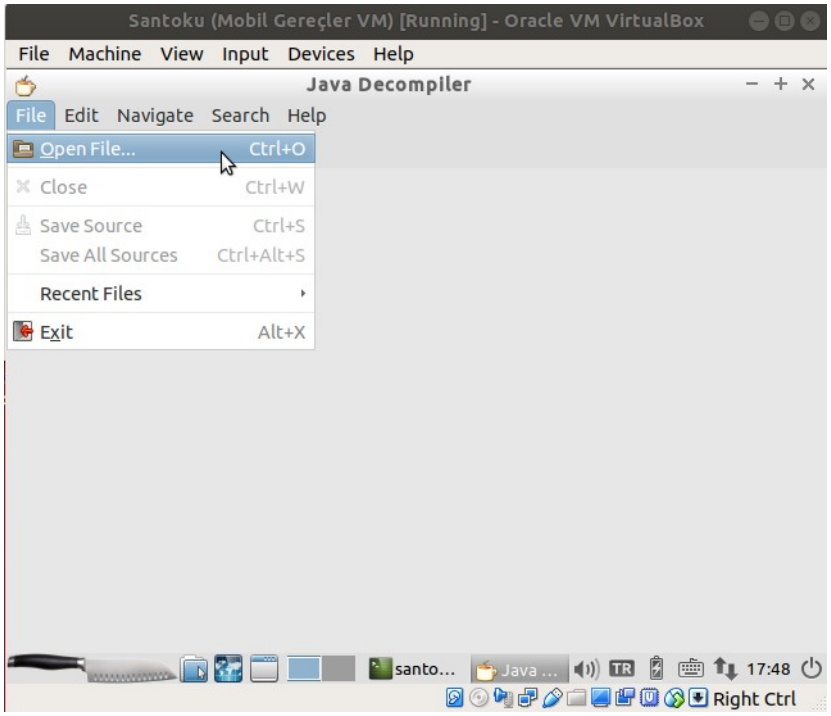
Bu bilgiler uzak sunucuda bir tür veritabanına kaydolur. Yerel mobil cihazda ise sonradan kullanım için bir dosya halinde saklanır / kaydedilir.

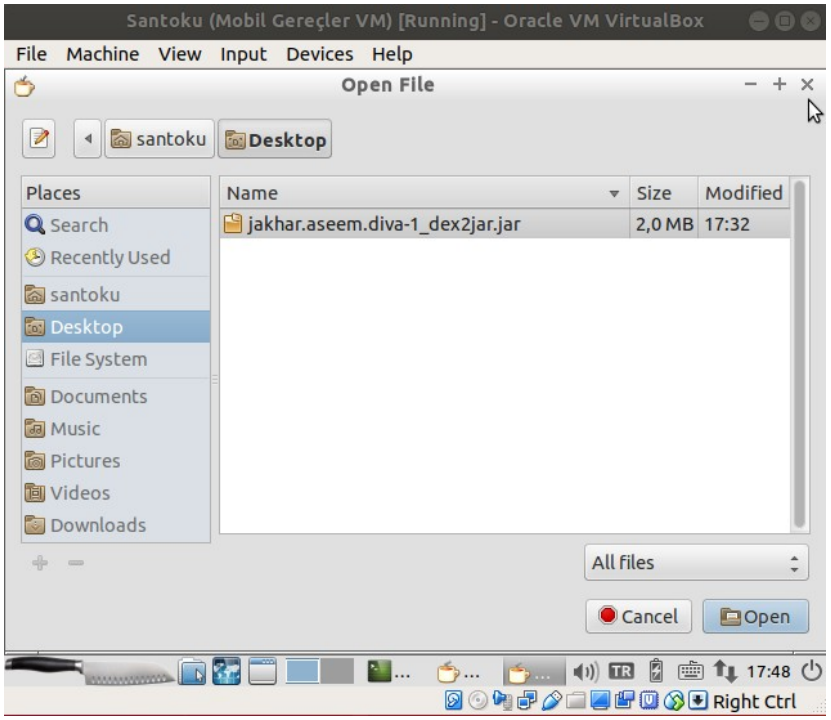
Şimdi bu makale dizisinin en başında yaptığımız uygulama binary dosyasını (.apk dosyasını) bilgisayara çekme, üzerinde tersine mühendislik yapma ve uygulamanın okunabilir java dosyalarını elde etme işini tekrarladığımızı varsayalım. Uygulamanın okunabilir java dosyalarını JD-GUI editörü ile inceleyerek bu uygulama sayfasını (“Insecure Data Storage - Part 1” sayfasını) sunan / kontrol eden java dosyasını tespit edelim. Bahsedilen işlemlerin ayrıntısı için bkz. Başlangıç: Android Uygulamalarda Tersine Mühendislik ile Okunabilir Java Kaynak Kodları Elde Etme (Dex2Jar, JD-GUI, ApkTool)

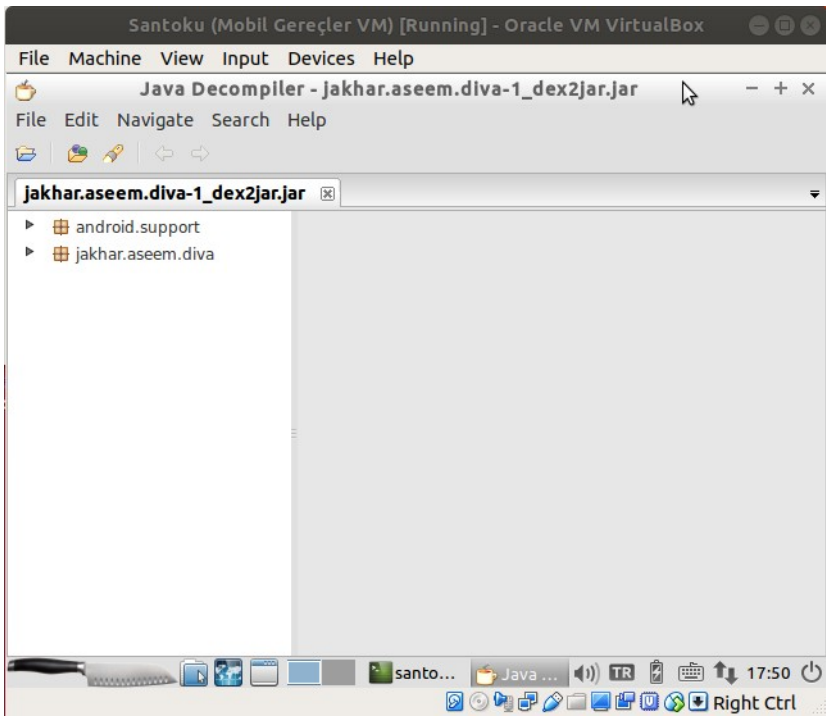
Santoku Linux Terminal:

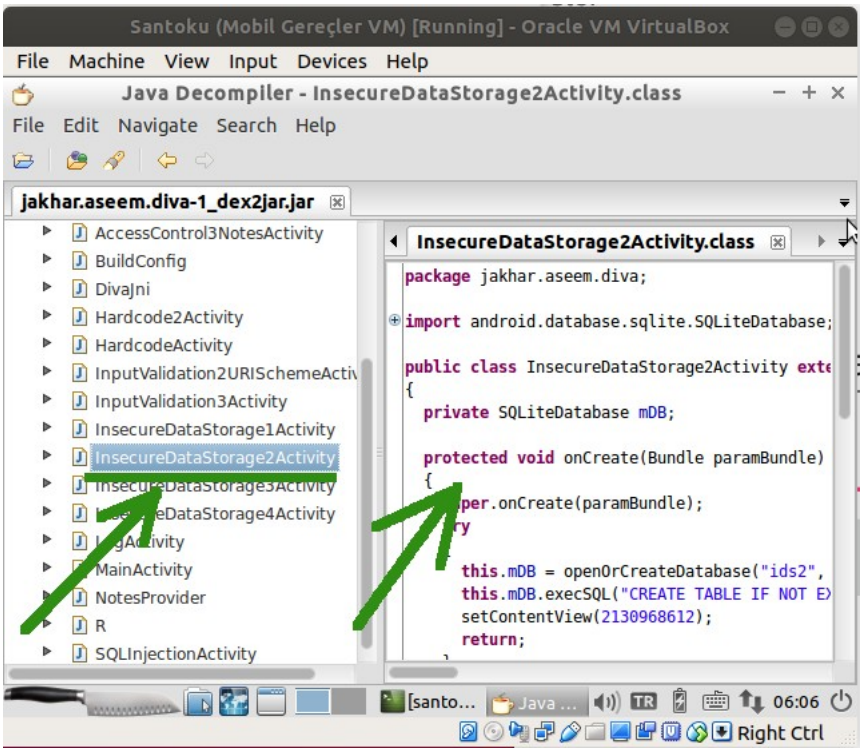
> jd-gui

Çıktı:


Santoku (Mobil Gereçler VM) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Java Decompiler
File Edit Navigate Search Help
Open File... Ctrl+O
Close Ctrl+W
Save Source Ctrl+S
Save All Sources Ctrl+Alt+S
Recent Files
Exit Alt+X
santo...
Java ...
TR
17:48
Right Ctrl


Santoku (Mobil Gereçler VM) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Open File
santoku
Desktop
Places
Search
Recently Used
santoku
Desktop
File System
Documents
Music
Pictures
Videos
Downloads
Name
Size
Modified
jakhar.aseem.diva-1_dex2jar.jar
2,0 MB
17:32
All files
Cancel
Open
TR
17:48
Right Ctrl

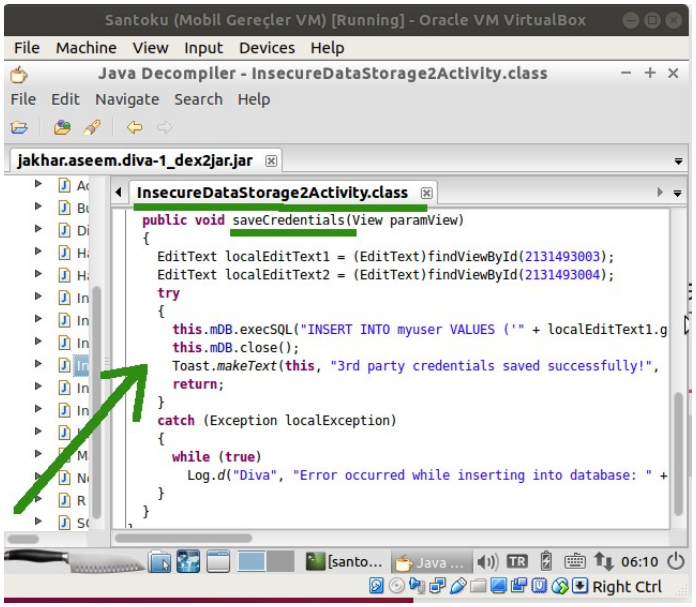




Java kaynak kod dosyaları incelendiğinde uygulamada görüntülüyor olduğumuz ekranın hangi java dosyası tarafından sunulduğu / kontrol edildiği yukarıdaki gibi görülebilir. Bu örnek için java dosyasını bulmak daha önceki makalelerde de bahsedildiği üzere uygulama ekranındaki sayfa ismi ile benzerliği dolayısıyla kolay olmuştur ama gerçek bir senaryoda java dosyaları arasında inceleme yapmak ve doğru java dosyasını bulmak için okumalar yapmak gerekecektir.

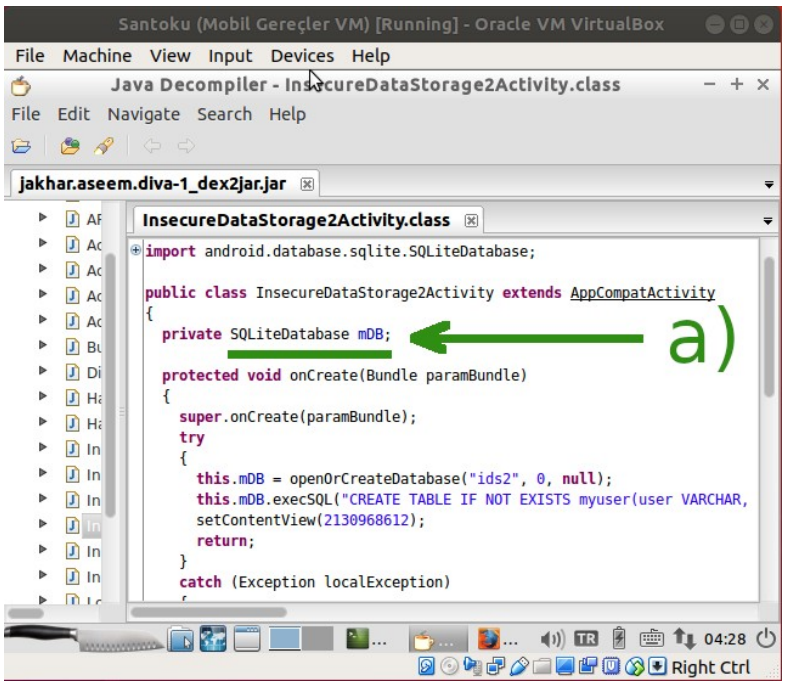
Üçüncü taraf servise kaydolduğumuz mobil uygulama sayfasını sunan / kontrol eden java dosyası yukarıdaki gibi InsecureDataStorage2Activity.class'tır. Bu java dosyası incelendiğinde

saveCredentials() isimli (yani Türkçe ifadeyle hesap bilgilerini kaydet isimli) bir metot kullanımı görünmektedir.

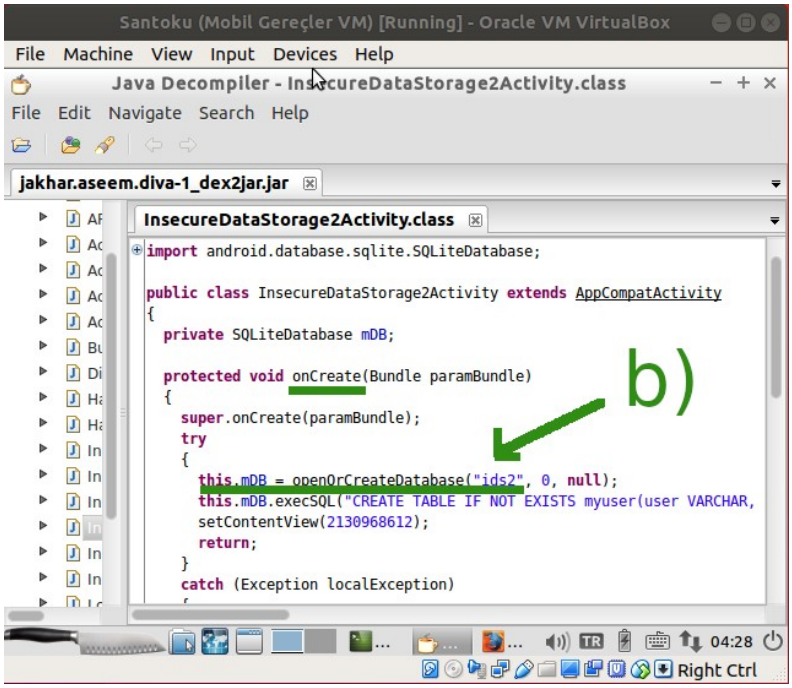


Metot içerisi incelendiğinde uygulama sayfasındaki metin kutuları nesneler halinde, içerdikleri verilerle beraber localEditText1 ve localEditText2'e atanmaktadırlar. Yani bu nesnelerden biri kullanıcı adını, diğeri parolayı kelime halinde tutmaktadır. Sonra try isimli bir bloğa girilmektedir. Bu blokta daha önceden tanımlanmış bir nesnenin (this.mDB'nin) execSQL() isimli metodu çağırılmaktadır ve bir sql sorgusu çalıştırılmaktadır. Bundan hareketle kullanılan depolama yönteminin yerel sistemde bir veritabanı çözümü olduğu tespitini yapabiliriz.

Aynı java dosyasındaki saveCredentials() isimli hesap kaydet metodunun yukarısında yer alan kodlar incelendiğinde ise saveCredentials() isimli metodun içerisinde kullanılan mDB nesnesinin bir SQLite nesnesi olarak tanımlandığını görmekteyiz:



Buna ilaveten hemen aşağısında yer alan onCreate() metodu ile (bu metot android dünyasında sayfa çalışırken otomatik çağırılan / çalıştırılan metot anlamına gelir) *ids2* isimli bir veritabanı dizini oluşturulduğunu görmekteyiz.



Yani bu bilgilerden hareketle kullanılan yerel depolama yönteminin ilişkisel veritabanı SQLite olduğu tespitini yapabiliriz. Çünkü java dosyasındaki saveCredentials() metodunda (Türkçe ifadeyle hesap kaydet metodunda),

```
public void saveCredentials(View paramView)
{
        EditText localEditText1 = (EditText)findViewById(2131493003);
        EditText localEditText2 = (EditText)findViewById(2131493004);

        try
        {
                this.mDB.execSQL("INSERT INTO myuser VALUES ('" +
localEditText1.getText().toString() + "', '" + localEditText2.getText().toString() + "');");
                this.mDB.close();
                Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
                return;
        }
        catch (Exception localException)
        {
                while (true)
                        Log.d("Diva", "Error occurred while inserting into database: " +
                localException.getMessage());
        }
}
```

bir sqlite nesnesi ile sql sorgusu çalıştırılmaktadır ve sorguya göre ekrandaki birinci metin kutusundan gelen veri ve sonra ikinci metin kutusundan gelen veri *myuser* adlı bir veritabanı tablosuna yazılmaktadır / eklenmektedir. Yani kullanıcı hesap bilgileri yerel depolamada veritabanı yöntemi ile saklanmaktadır. Veritabanı olarak SQLite kullanılmaktadır. Veritabanı dizini olarak *ids2* oluşturulmaktadır ve kullanıcı servis hesap bilgileri *ids2* veritabanı dizini içerisindeki *myuser* tablosuna yazılmaktadır / eklenmektedir.

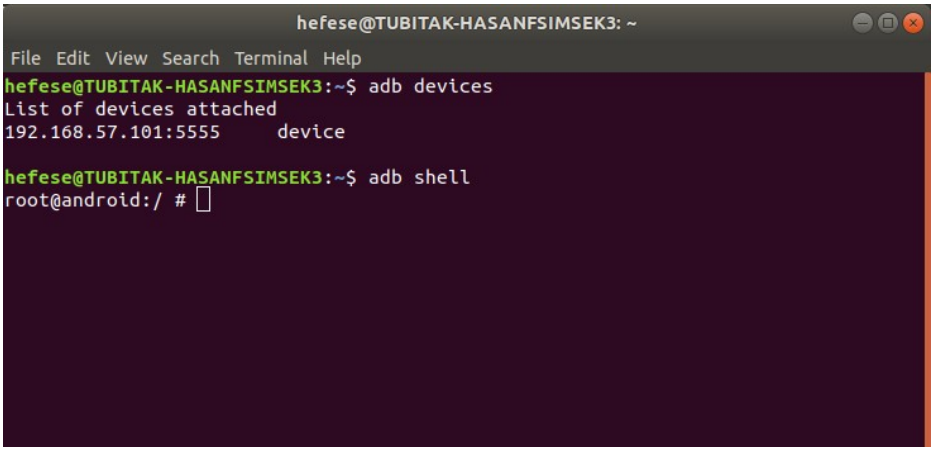
Uygulama java kaynak kodunu inceleyerek kullanıcı hassas verilerinin nerede tutulduğu bilgisine (SQLite veritabanı tablosunda tutulduğu bilgisine) eriştik. Şimdi nasıl tutulduğu bilgisine erişmek için bilgisayardan mobil cihaza bağlanalım ve kullanıcı hassas verilerinin tutulduğu SQLite ilgili veritabanı dosyasını linux bilgisayara çekelim. Ardından linux bilgisayarda veritabanı dosyasını açalım ve içindeki servis hesap bilgilerini görüntüleyelim.

Öncelikle linux bilgisayardan mobil cihaza resmi Android SDK aracı adb ile bağlanalım ve shell komutu ile mobil cihazın komut satırını alalım.

Ubuntu 18.04 LTS Linux Terminal:

> adb devices
> adb shell

Çıktı:



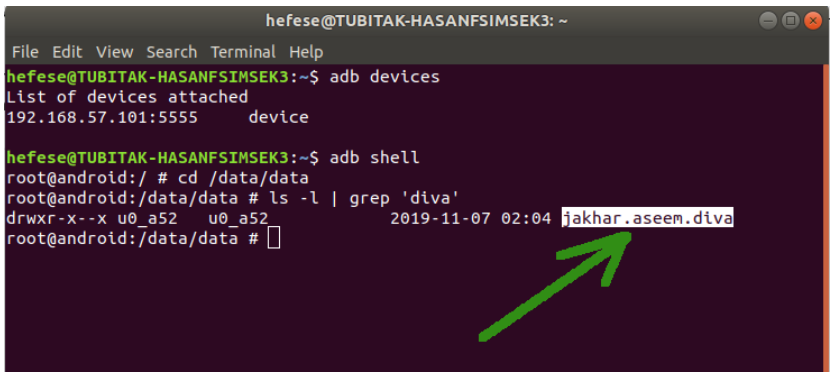
Ardından yapılacak şey mobil cihazdaki diva uygulamasının android sistemdeki ismini öğrenmektir, sonra android sistemdeki /data/data dizini altında öğrendiğimiz paket ismindeki dizinin içerisine girmektir. Bu şekilde uygulama paket ismindeki klasör altında database/ adlı bir klasör ekrana yansıyacaktır ve klasörün içerisinde ise veritabanı dosyası listelenecektir.

Linux bilgisayardan mobil cihazla olan komut satırı oturumuzda diva uygulamasının android sistemdeki paket ismini öğrenelim.

Ubuntu 18.04 LTS Terminal:

root@android:/ # cd /data/data/
root@android:/data/data # ls -l | grep 'diva'

Çıktı:

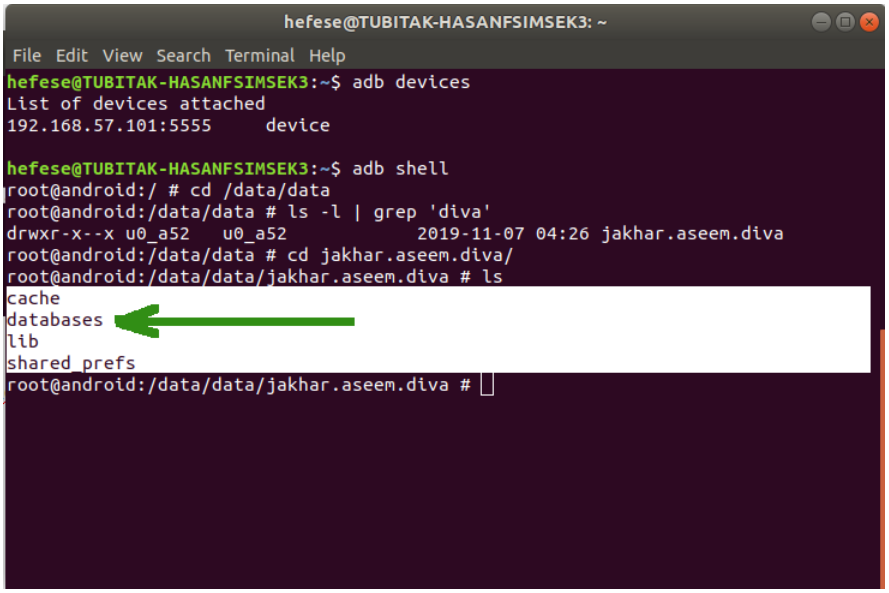


Ardından /data/data dizini altındaki diva ugulamasının android sistemdeki paket isminde olan klasöre girelim ve databases/ klasörünü görüntüleyelim.

Ubuntu 18.04 LTS Terminal:

root@android:/data/data # cd jakhar.aseem.diva/
root@android:/data/data/jakhar.aseem.diva # ls

Çıktı:

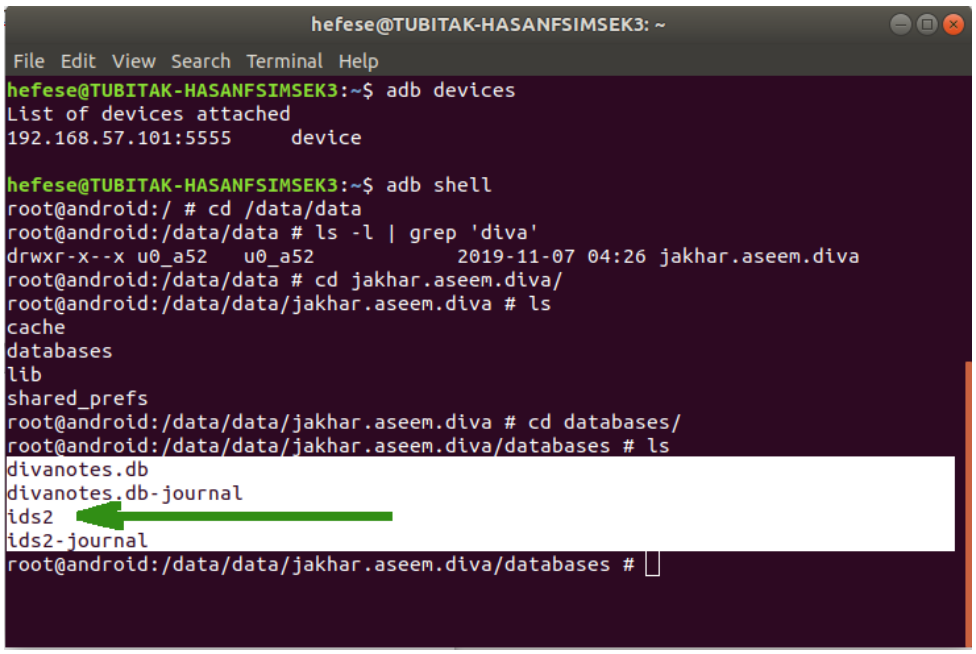


databases/ klasörü içerisinde veritabanı dosyaları yer alacaktır. Veritabanı dosyalarını sıralayalım.

Ubuntu 18.04 LTS Terminal:

root@android:/data/data/jakhar.aseem.diva # cd databases/
root@android:/data/data/jakhar.aseem.diva/shared_prefs # ls

Çıktı:

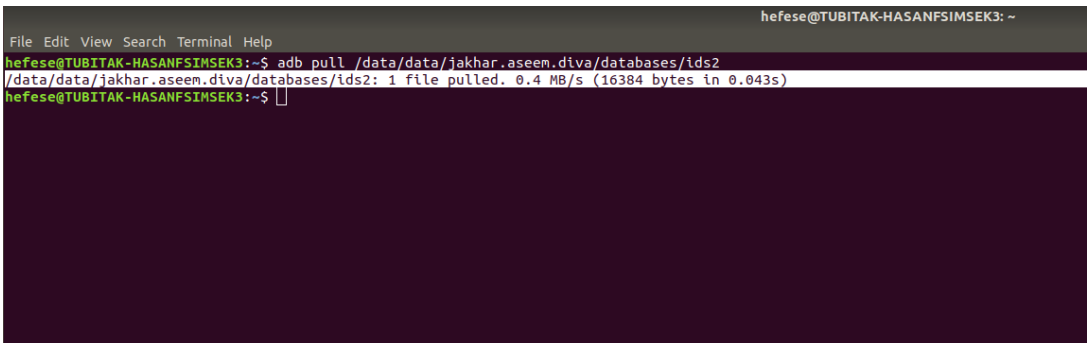


Diva uygulamasına ait SQLite'ın içerdiği veritabanı dosyalarını görüntülemekteyiz. Uygulama ekranını sunan / kontrol eden java dosyasındaki kaynak kodlara baktığımız sıralar değindiğimiz üzere *ids2* veritabanı dosyası oluşturulmaktaydı ve bu veritabanı dosyası içerisindeki *myuser* adlı tabloya kullanıcı servis hesap bilgileri yazılmaktaydı. Bu nedenle sıralanan veritabanı dosyaları içerisinden *ids2* veritabanı dosyasını adb aracının pull parametresi ile linux bilgisayara çekelim.

Ubuntu 18.04 LTS Terminal:

> adb pull /data/data/jakhar.aseem.diva/databases/ids2

Çıktı:

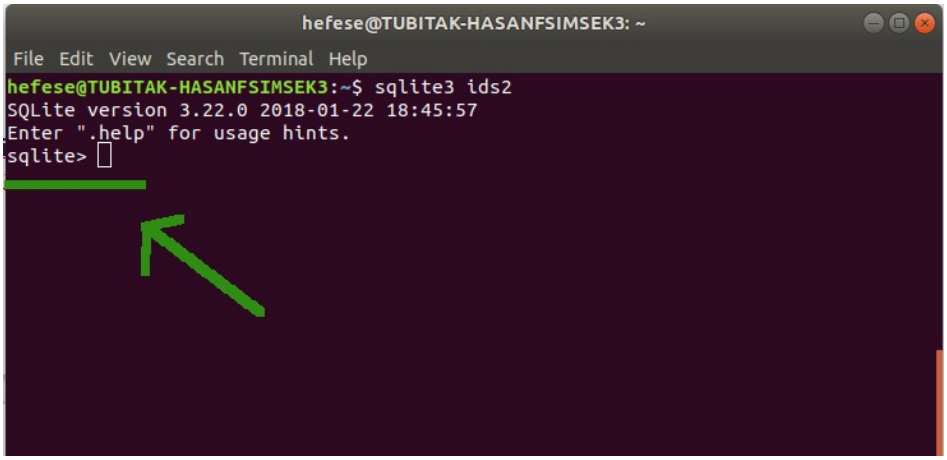


Ardından resmi android sdk aracı sqlite3 ile *ids2* veritabanı dosyasını açalım.

Ubuntu 18.04 LTS Terminal:

> sudo apt-get install sqlite3 // Debian linux sisteme sqlite3 aracı yüklenir
> sqlite3 ids2 // Sqlite3 ids2 isimli veritabanı dosyası açılır.

Çıktı:

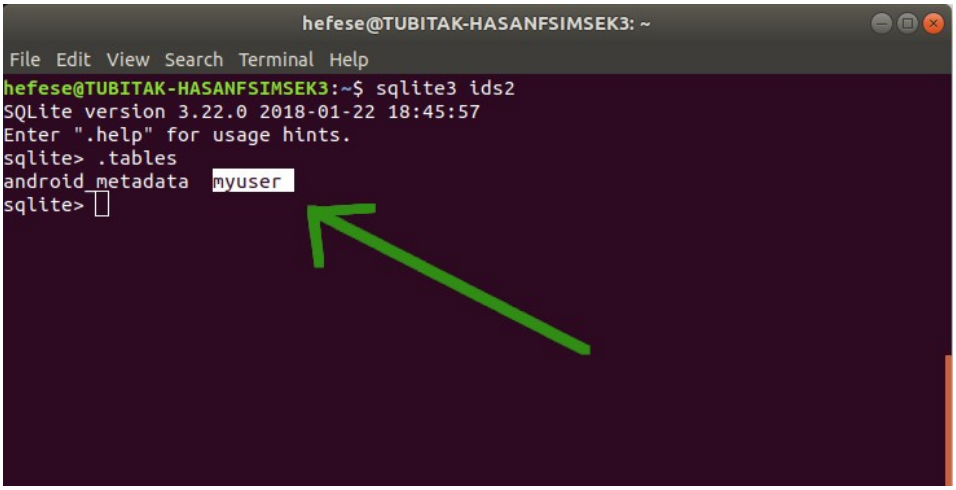


Ardından Sqlite komut satırına .tables komutunu girerek *ids2* veritabanı dosyasında yüklü tabloları sıralayalım.

Ubuntu 18.04 LTS Terminal:

sqlite> .tables

Çıktı:

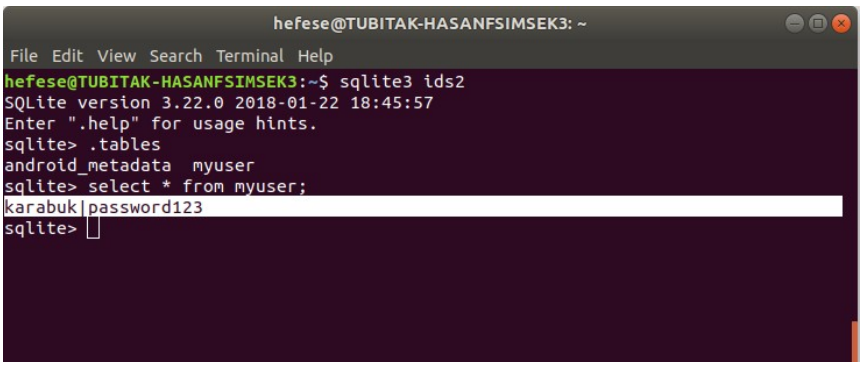


Son olarak ise daha önce ilgili uygulama sayfası java kaynak kodlarındaki sql sorgusundan edindiğimiz bilgiden hareketle kullanıcının kayıt için girdiği servis hesap bilgileri (kullanıcı adı ve şifre bilgisi) *myuser* tablosuna yazılmaktaydı. Bu nedenle *myuser* tablosu verilerini ekrana basalım.

Ubuntu 18.04 LTS Terminal:

sqlite> select * from myuser; // Myuser tablosundaki tüm kolonların
// tuttuğu değerleri ekrana basar.

Çıktı:



Görüldüğü üzere üçüncü taraf servis hesap bilgileri mobil cihazda yerel olarak bir veritabanı tablosu içerisinde açık bir şekilde (güvensiz bir şekilde) depolanmıştır. Bu mobil cihaza veya başka söz gelimi diva uygulamasını kullanan mobil cihazlara sızmayı başaran saldırganlar diva uygulaması klasörü içerisinde kritik bir veri elde edebilir miyim diye inceleme yaptıklarında databases/ klasörü onlara kullanıcının diva uygulamasındaki bağlı olduğu üçüncü taraf servisin hesap bilgilerini verecektir.

Şimdi uygulama sayfasındaki kullanıcı hesap bilgilerini kaydetme metodundaki güvensiz veri depolayan kod satırlarını gösterelim ve başlığı kapatalım.

```
public void saveCredentials(View paramView)
{
        EditText localEditText1 = (EditText)findViewById(2131493003);
        EditText localEditText2 = (EditText)findViewById(2131493004);

        try
        {
                this.mDB.execSQL("INSERT INTO myuser VALUES ('" +
localEditText1.getText().toString() + "', '" + localEditText2.getText().toString() + "');");
                this.mDB.close();
                Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
                return;
        }
        catch (Exception localException)
        {
                while (true)
                        Log.d("Diva", "Error occurred while inserting into database: " +
                localException.getMessage());
        }
}
```

SQL sorgusu çalıştırılan satırda görüldüğü üzere uygulama sayfasındaki metin kutularından gelen veriler olduğu gibi sorguya eklenmişlerdir ve sorgu bu şekilde çalıştığından metin kutularından gelen veriler aynen oldukları gibi veritabanı tablosuna yazdırılmaktadır. Bu satırda normalde bir çeşit şifreleme metotlarıyla kullanıcı adı ve/veya parola şifrelenip o şekilde veritabanı tablosuna yazdırılmalıydı.

>> Güvensiz Veri Depolama Kod Satırları

```
this.mDB.execSQL("INSERT INTO myuser VALUES ('" +
localEditText1.getText().toString() + "', '" + localEditText2.getText().toString() + "');");
```

Bunun yerine direk olduğu gibi metin kutusundan gelen kullanıcı adı ve parola bilgileri veritabanı tablosuna yazdırıldığından yerel veritabanı depolama yöntemi kullanıcı adı ve şifreyi ifşa eder durumda kalmıştır.

Sonuç

Diva uygulamasının Insecure Data Storage - Part 2 sayfasındaki girilen üçüncü taraf servise kaydolma hesap bilgilerinin nerede ve ne şekilde kaydolduğu bilgisi tersine mühendislik ile elde edilen uygulama kaynak kodlarının incelenmesiyle tespit edilmiştir. Nerede olduğu bilgisi kaynak kodlarda kullanılan SQLiteDatabase java sınıfı isminden, ne şekilde tutulduğu bilgisi ise (yani şifreli mi, yoksa açık metin halinde mi bilgisi) android sistemdeki diva uygulaması klasörü altında yer alan databases/ klasörü içerisindeki veritabanı dosyasının barındırdığı tablo kayıtlarından görülerek anlaşılmıştır.

Bu derste vurgulanmak istenen açıklık diva uygulamasındaki kullanıcı servis hesap bilgilerinin açık metin halinde / şifrelenmeden yerel sistemdeki veritabanı dosyasında tutulmasıdır. Normal şartlarda hiçbir güvenlikçi perspektife sahip geliştirici veritabanında uygulama için veya kullanıcı için hassas nitelikte olan verileri (örn; kullanıcı adı ve/veya şifreyi) açık metin halinde tutmazlar. Bu hassas verileri erişimi kısıtlı veritabanı dosyasında tutsalar dahi şifreli halde tutarlar ki sonradan bir kaza meydana geldiğinde (yani uygulamada veya uygulamanın yer aldığı sistemde bir açıklık meydana geldiğinde) sızan saldırganların daha fazla zarar verememesi / daha fazla bilgi elde edememesi hedeflenir. Bu bir tür saldırganlara karşı frenleme önlemidir. Yani çok daha ileriye gidebileceklerken (çok daha kritik veri elde edip daha fazla zarar verebileceklerken) frenleyerek bir noktada tutma tedbiridir. Veritabanında kritik veriler şifreli halde tutulurlarsa saldırganlar veritabanı dosyasına ele geçirseler dahi dosyayı bu derste olduğu gibi açtıklarında kullanıcı parolasını şifrelenmiş halde göreceklerdir ve parolanın açık halini elde edebilmek için bir dünya güçlükle karşılaşmış olacaklardır. Bu nedenle veritabanı dosyalarında kritik mahiyette veri tutulacağı zaman bu veriler şifrelenmiş halde tutulmalıdırlar.

Sonuç olarak kullanıcı servis hesap bilgileri açık metin halinde / şifrelenmeden yerel sistemdeki veritabanı dosyasında tutulduğu için mobil cihaza ilerde sızacak bir saldırganın kullanıcı hassas verilerini okuyabilmesi ve elde edebilmesi yolu açık durumdadır. Dolayısıyla android uygulamaların yerel sistemde veritabanı ile saklayacakları hassas veriler şifrelenmek suretiyle tutulmalıdırlar.

Kaynak

https://www.loginworks.com/blogs/top-10-ways-know-store-data-android/
https://stackoverflow.com/questions/16691437/when-are-java-temporary-files-deleted
https://source.android.com/devices/tech/connect/third-party-call-apps
https://developer.android.com/guide/topics/data/data-storage
https://developer.android.com/training/data-storage/shared-preferences
https://www.tutorialspoint.com/android/android_shared_preferences.htm