Ex140-MobileAppTest

Jigar Patel

2023-12-05

Contents

1 Technical Report

1.1 Finding: *Plaintext Database Credentials in "Arts Tailor News" App*

Severity Rating

CVSS Base Severity Rating: 8.3 AV:N AC:L PR:N UI:L S:C C:L I:L A:L

Vulnerability Description

On decompiling "Arts Tailor News" App, an attacker is able to find the database credentials used to make queries to **db.artstailor.com**. The attacker can then use these credentials to log in to db.artstailor.com and exfiltrate data (including database data and configurations) not meant to be access publicly.

Confirmation method

Using a Dex to Java decompiler like **jdax-gui**, one can view the APK source code and find the base64 encoded database credentials in **ItemListActivity.java**. These credentials can be decoded using online tools or the base64 command.

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

    Async() {
    }

    /* JADX INFO: Access modifiers changed from: protected */
    @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.getStrin
            }
            return null;
        } catch (Exception e) {
            this.error = e.toString();
            return null;
        }
```

Mitigation or Resolution Strategy

1. Leveraging Android keystores or other native APIs that encrypt and store sensitive data like database credentials is a must.

2. Code obfuscation tools such as ProGuard should be used to make reverse engineering harder and time-consuming.

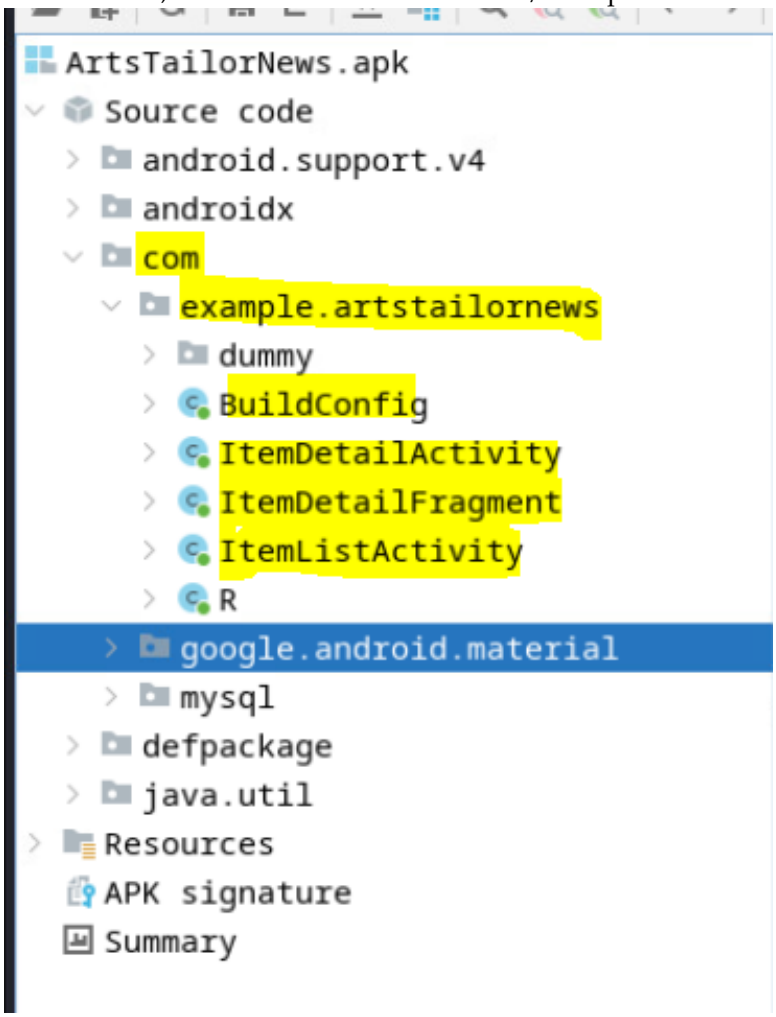
3. Code audits should be done frequently to check for bad practices and compliance violations.

4. Additionally, the backend database should be encrypting or using tokenization (or both) to protect sensitive data like credit cards details, so that even if the database credentials are leaked, such sensitive data is not readily accessible.

2 Attack Narrative - Finding Secrets in "Arts Tailor News" App

1. We download the APK from *http://www.artstailor.com/apps/ArtsTailorNews.apk* using *wget*.

2. Next, we open the APK in jadx-gui. The APK has the following directory structure and, we find files of interest in com/example.artstailor.



3. jadx-gui decompiles the Dalvik byte code and shows us the Java code. However, to find the entry point and permission of the application, we first look at **manifest.xml** in **Resources**.

The following observations can be made:

(a) We see that the parent activity (also the same activity captures MAIN and LAUNCHER intents) of the application is **ItemListActivity**.

(b) Also, the application asks for only **android.permissions.INTERNET**. No extra permissions are asked by the application, which is a best practice.

4. Looking at the **ItemListActivity**, we find that the application is making **Select query** to db.artstailor.com using hard-coded username and password.

```
/* loaded from: classes.dex */
class Async extends AsyncTask<Void, Void, Void> {
    String records = "";
    String error = "";
    String b64username = "ZGJfdXNlc190b2tlbgo=";
    String b64password = "S[redacted]o=";

    Async() {
    }

    /* JADX INFO: Access modifiers changed from: protected */
    @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.getStrin
            }
            return null;
        } catch (Exception e) {
            this.error = e.toString();
            return null;
        }
```

5. Decoding the base64 username (which turns out to be **db_user_token**) and password, we find the credentials we could use to make queries to **db.artstailor.com** running mysql.

2.1 MITRE ATT&CK Framework TTPs

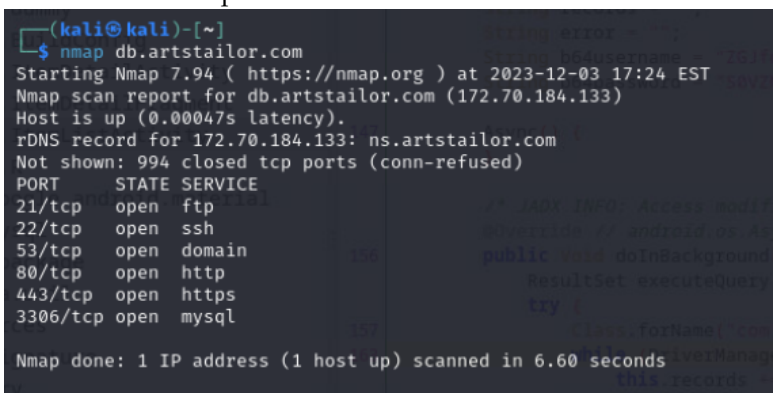
TA0001: Initial Access

T1190: Exploit Public-Facing Application

NA: NA
TA0007: Discovery
T1087: Account Discovery
.001: Local Account
TA0007: Discovery
T1046: Network Service Discovery
NA: NA

3 Attack Narrative - Discovering Plaintext Sensitive Data in db.artstailor.com

1. We first run a nmap scan on **db.artstailor.com**.



   We see mysql running on its default port 3306. Which is actually very much expected since the connection string **jdbc:mysql:://db.artstailor.com/android** specifies no ports.

2. Next, we use the base64 decoded username and password to log in to the mysql server.

3. We use mysql commands **show databases** and **show tables** to enumerate databases and tables, respectively.

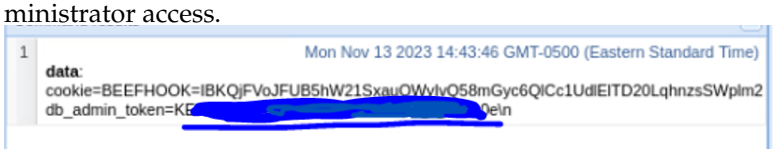
```
MariaDB [customerdb]> show databases;
+--------------------+
| Database           |
+--------------------+
| customerdb         |
| information_schema |
+--------------------+
2 rows in set (0.001 sec)
```

4. We find customerdb interesting and query it using **select**.

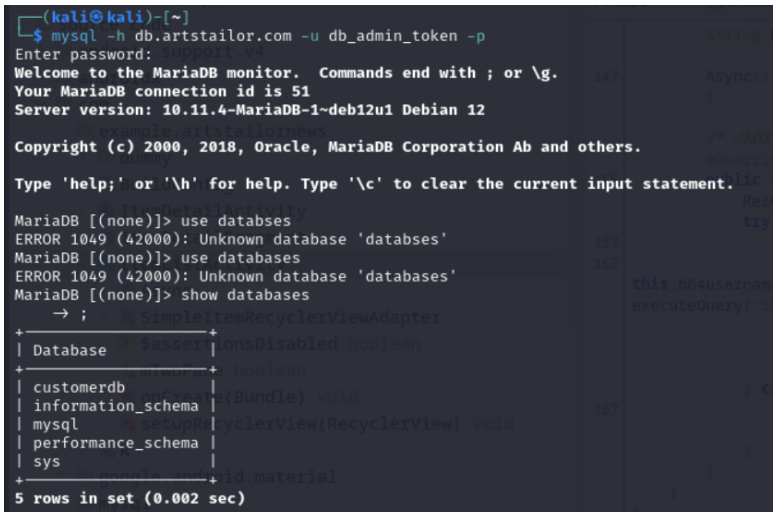
```
MariaDB [customerdb]> use customerdb
Database changed
MariaDB [customerdb]> show tables
    -> ;
+--------------------+
| Tables_in_customerdb |
+--------------------+
| news               |
+--------------------+
1 row in set (0.001 sec)

MariaDB [customerdb]> select * from news
    -> ;
+---------+-----------------------------------------------------------------------------------------------------------------------------------+
| item_id | text                                                                                                                              |
+---------+-----------------------------------------------------------------------------------------------------------------------------------+
|       1 | Art's Tailor Shoppe, now featuring indestructable prom dresses. Get yours before they are all gone! |
+---------+-----------------------------------------------------------------------------------------------------------------------------------+
```

However, we don't find anything of interest in these tables.

5. Previously, we had found a database token by using the BeEf Framework. The token was similarly named db_admin_token but probably with administrator access.

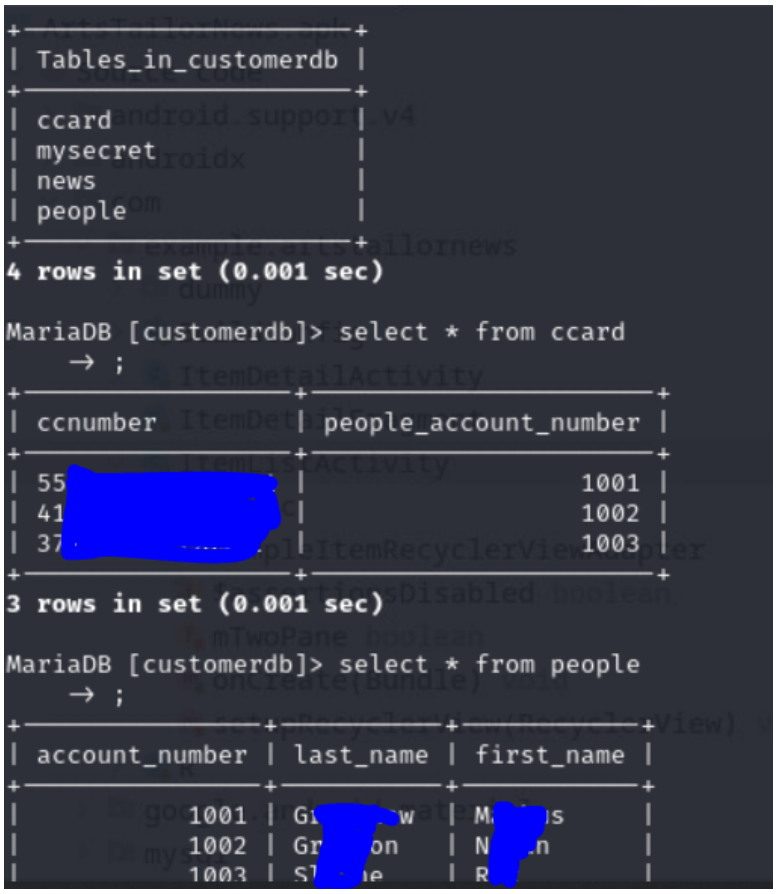


6. Using these credentials, we again log in to mysql and enumerate databases.



We observe that we are able to query more databases, including sys.

7. Using customerdb, we again list the contents of the tables using **show tables** and **select** statements.



We observe the following:

(a) We have two tables of interest ccard and people.

(b) **ccard** contains credit card numbers of various people indexed by their database ID.

(c) **people** contains the mapping of people's name to their database ID.

(d) The table **mysecret** contains the string "Browser's don't lie...".

3.1 MITRE ATT&CK Framework TTPs

TA0001: Initial Access
T1078: Valid Accounts
.003: Local Accounts
TA0009: Collection
T1005: Data from Local System
NA: NA