|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

РТУ МИРЭА г. Москва

Кафедра Информационно-аналитические системы кибербезопасности (КБ - 2)

**ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИКИ № 2**

по дисциплине

«Проектирование и разработка безопасного программного обеспечения информационно-аналитических систем»

|  |  |
| --- | --- |
| Выполнил студент группы БИСО-03-19 | *Иванов Е. А.* |
| Принял | *доцент*  *Латыпов И. Т.* |

|  |  |
| --- | --- |
| Работа выполнена | «\_\_»\_\_\_\_\_\_\_2024г. |
|  |  |
| «Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_» | «\_\_»\_\_\_\_\_\_\_2024г. |

Москва 2024

InsecureDataStorage3Activity.java:

public class InsecureDataStorage3Activity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_insecure\_data\_storage3);  
 }  
  
 public void saveCredentials(View view) {  
 EditText usr = (EditText) findViewById(R.id.ids3Usr);  
 EditText pwd = (EditText) findViewById(R.id.ids3Pwd);  
  
 File ddir = new File(getApplicationInfo().dataDir);  
  
 try {  
 File uinfo = File.*createTempFile*("uinfo", "tmp", ddir);  
 uinfo.setReadable(true);  
 uinfo.setWritable(true);  
 FileWriter fw = new FileWriter(uinfo);  
 fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");  
 fw.close();  
 Toast.makeText(this, "3rd party credentials saved successfully!", Toast.*LENGTH\_SHORT*).show();  
 // Now you can read the temporary file where ever the credentials are required.  
 }  
 catch (Exception e) {  
 Toast.makeText(this, "File error occurred", Toast.*LENGTH\_SHORT*).show();  
 Log.*d*("Diva", "File error: " + e.getMessage());  
 }  
 }  
}

Приложение сохраняет введенные пользователем логин и пароль во временный файл в каталоге данных приложения.

1. Имеет место **CWE-276: Incorrect Default Permissions** (некорректные разрешения по умолчанию).

Установка атрибутов файла

uinfo.setReadable(true);  
uinfo.setWritable(true);

позволяет файлу потенциально быть прочитанным и измененным другими приложениями или компонентами системы, имеющими на это разрешения.

Каталог приложения обычно недоступен для других приложений и пользователей, однако это может не помешать злоумышленнику получить оттуда данные. Например,

* методом «рутирования» и, соответственно, получением полного доступа к операционной системе и файловой системе устройства, включая системные файлы и файлы других приложений
* методом запросов разрешений на доступ к хранилищу пользователю

Данные методы хорошо работали на старых версиях Android:

* До Android 6.0 приложения могли запрашивать намного больше разрешений при установке и принимать их все одновременно.
* До Android 10 не было динамических разделов - Dynamic Partitions, используемых для проверки целостности системы и усложняющих процесс её модификации и, следовательно, рутирования.
* Также в 10 версии была представлена новая модель управления файлами и данными на устройствах - Scoped Storage, ограничивающая прямой доступ к файловой системе устройства без явного согласия пользователя и в которой каждое приложение имеет доступ только к своему собственному уникальному пространству файлов.

Решение: создавать файл без явного указания разрешений на его чтение и запись в него, использовать новые версии Android.

public void saveCredentials(View view) {  
 EditText usr = findViewById(R.id.ids3Usr);  
 EditText pwd = findViewById(R.id.ids3Pwd);  
  
 File ddir = new File(getApplicationInfo().dataDir);  
  
 try {  
 File uinfo = File.createTempFile("uinfo", "tmp", ddir);

// удалили вызовы setReadable и setWritable

FileWriter fw = new FileWriter(uinfo);  
 fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");  
 fw.close();

...  
}

1. Введенные конфиденциальные данные хранятся в незашифрованном виде.

Это **CWE-312: Cleartext Storage of Sensitive Information** (хранение конфиденциальной информации в открытом виде).

Такие данные, как логин и пароль, стоит хранить в зашифрованном виде для предотвращения их прочтения при несанкционированном доступе.

Решение: перед сохранением данных шифровать их.

Добавим необходимые импорты и константы:

import android.security.keystore.KeyGenParameterSpec;  
import android.security.keystore.KeyProperties;  
  
import java.io.FileOutputStream;  
import java.security.KeyStore;  
import java.security.KeyStore.SecretKeyEntry;  
import javax.crypto.Cipher;  
import javax.crypto.KeyGenerator;  
import javax.crypto.SecretKey;  
import javax.crypto.spec.GCMParameterSpec;

private static final String *KEY\_ALIAS* = "MyKeyAlias"; //ключ  
private static final String *ANDROID\_KEYSTORE* = "AndroidKeyStore"; //хранилище ключей  
private static final String *TRANSFORMATION* = "AES/GCM/NoPadding"; //параметры шифрования

Определим методы для генерации ключа и шифрования данных:

private void generateKey() {  
 try {  
 KeyGenerator keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY\_ALGORITHM\_AES, ANDROID\_KEYSTORE);  
 keyGenerator.init(new KeyGenParameterSpec.Builder(KEY\_ALIAS,  
 KeyProperties.PURPOSE\_ENCRYPT | KeyProperties.PURPOSE\_DECRYPT)  
 .setBlockModes(KeyProperties.BLOCK\_MODE\_GCM)  
 .setEncryptionPaddings(KeyProperties.ENCRYPTION\_PADDING\_NONE)  
 .build());  
 keyGenerator.generateKey();  
 } catch (Exception e) {  
 throw new RuntimeException("Failed to generate key", e);  
 }  
}

private byte[] encryptData(String data) {  
 try {  
 KeyStore keyStore = KeyStore.getInstance(ANDROID\_KEYSTORE);  
 keyStore.load(null);  
 SecretKey secretKey = ((SecretKeyEntry) keyStore.getEntry(KEY\_ALIAS, null)).getSecretKey();  
  
 Cipher cipher = Cipher.getInstance(TRANSFORMATION);  
 cipher.init(Cipher.ENCRYPT\_MODE, secretKey);  
 byte[] iv = cipher.getIV();  
 byte[] encryptedData = cipher.doFinal(data.getBytes());  
  
 byte[] combined = new byte[iv.length + encryptedData.length];  
 System.*arraycopy*(iv, 0, combined, 0, iv.length);  
 System.*arraycopy*(encryptedData, 0, combined, iv.length, encryptedData.length);  
  
 return combined;  
 } catch (Exception e) {  
 throw new RuntimeException("Failed to encrypt data", e);  
 }  
}

В методе onCreate сгенерируем ключ:

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_secure\_data\_storage);  
 generateKey();  
}

В методе saveCredentials зашифруем данные и поместим их в файл:

public void saveCredentials(View view) {  
 EditText usr = findViewById(R.id.ids3Usr);  
 EditText pwd = findViewById(R.id.ids3Pwd);

String credentials = usr.getText().toString() + ":" + pwd.getText().toString();  
 byte[] encryptedData = encryptData(credentials);

File ddir = new File(getApplicationInfo().dataDir);  
  
 try {  
 File uinfo = File.createTempFile("uinfo", "tmp", ddir);  
 // uinfo.setReadable(true);   
 // uinfo.setWritable(true);

try (FileOutputStream fos = new FileOutputStream(uinfo)) {

// используем FileOutputStream, так как данные шифруются в массив байт  
 fos.write(encryptedData);  
 }

...

}

SQLInjectionActivity.java:

public class SQLInjectionActivity extends AppCompatActivity {  
  
 private SQLiteDatabase mDB;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 try {  
 mDB = openOrCreateDatabase("sqli", MODE\_PRIVATE, null);  
 mDB.execSQL("DROP TABLE IF EXISTS sqliuser;");  
 mDB.execSQL("CREATE TABLE IF NOT EXISTS sqliuser(user VARCHAR, password VARCHAR, credit\_card VARCHAR);");  
 mDB.execSQL("INSERT INTO sqliuser VALUES ('admin', 'passwd123', '1234567812345678');");  
 mDB.execSQL("INSERT INTO sqliuser VALUES ('diva', 'p@ssword', '1111222233334444');");  
 mDB.execSQL("INSERT INTO sqliuser VALUES ('john', 'password123', '5555666677778888');");  
  
 }  
 catch(Exception e) {  
 Log.*d*("Diva-sqli", "Error occurred while creating database for SQLI: " + e.getMessage());  
 }  
 setContentView(R.layout.activity\_sqlinjection);  
 }  
  
 public void search(View view) {  
 EditText srchtxt = (EditText) findViewById(R.id.ivi1search);  
 Cursor cr = null;  
 try {  
 cr = mDB.rawQuery("SELECT \* FROM sqliuser WHERE user = '" + srchtxt.getText().toString() + "'", null);  
 StringBuilder strb = new StringBuilder("");  
 if ((cr != null) && (cr.getCount() > 0)) {  
 cr.moveToFirst();  
  
 do {  
 strb.append("User: (" + cr.getString(0) + ") pass: (" + cr.getString(1) + ") Credit card: (" + cr.getString(2) + ")\n");  
 } while (cr.moveToNext());  
 }  
 else {  
 strb.append("User: (" + srchtxt.getText().toString() +") not found");  
 }  
 Toast.makeText(this, strb.toString(), Toast.*LENGTH\_SHORT*).show();  
 }  
 catch(Exception e) {  
 Log.*d*("Diva-sqli", "Error occurred while searching in database: " + e.getMessage());  
 }  
 }  
}

Приложение представляет собой простейший поиск информации о пользователе в базе данных.

Но, всё равно, необходимо знать точный username для успешного результата поиска.

Однако, например, путем подачи на вход строки **‘ OR ‘1’=’1** получим вместо запроса **SELECT \* FROM sqliuser WHERE user = ‘[usermane]’** запрос **SELECT \* FROM sqliuser WHERE user = ‘’ OR ‘1’=‘1’**, что приведет к выводу информации обо всех пользователях, игнорируя поиск имени пользователя.

Это **CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')** (нейтрализация спец. элементов в SQL-запросе).

Решение: удалять спец. символы при обработке введенной строки.

Определим метод removeSpecialCharacters, который будет удалять все символы, кроме букв и цифр:

private String removeSpecialCharacters(String input) {  
 String cleanInput = input.replaceAll("[^a-zA-Z0-9]", "");  
 return cleanInput;  
}

Затем в методе search:

public void search(View view) {

...

EditText srchtxt = (EditText) findViewById(R.id.ivi1search);  
String userInput = srchtxt.getText().toString();  
String cleanedInput = removeSpecialCharacters(userInput);

try {  
 cr = mDB.rawQuery("SELECT \* FROM sqliuser WHERE user = '" + cleanedInput + "'", null);

...

}

AccessControl3Activity.java и AccessControl3NotesActivity.java

public class AccessControl3Activity extends AppCompatActivity {  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_access\_control3);  
  
 SharedPreferences spref = PreferenceManager.*getDefaultSharedPreferences*(this);  
 String pin = spref.getString(getString(R.string.pkey), "");  
  
 if (!pin.isEmpty()) {  
 Button vbutton = (Button) findViewById(R.id.aci3viewbutton);  
 vbutton.setVisibility(View.*VISIBLE*);  
 }  
 }  
  
 public void addPin(View view) {  
 SharedPreferences spref = PreferenceManager.*getDefaultSharedPreferences*(this);  
 SharedPreferences.Editor spedit = spref.edit();  
 EditText pinTxt = (EditText) findViewById(R.id.aci3Pin);  
 String pin = pinTxt.getText().toString();  
  
 if (pin == null || pin.isEmpty()) {  
 Toast.makeText(this, "Please Enter a valid pin!", Toast.*LENGTH\_SHORT*).show();  
 }  
 else {  
 Button vbutton = (Button) findViewById(R.id.aci3viewbutton);  
 spedit.putString(getString(R.string.pkey), pin);  
 spedit.commit();  
 if (vbutton.getVisibility() != View.*VISIBLE*) {  
 vbutton.setVisibility(View.*VISIBLE*);  
 }  
  
 Toast.makeText(this, "PIN Created successfully. Private notes are now protected with PIN", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 public void goToNotes(View view) {  
 Intent i = new Intent(this, AccessControl3NotesActivity.class);  
 startActivity(i);  
 }  
}

public class AccessControl3NotesActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_access\_control3\_notes);  
 }  
  
 public void accessNotes(View view) {  
 EditText pinTxt = (EditText) findViewById(R.id.aci3notesPinText);  
 Button abutton = (Button) findViewById(R.id.aci3naccessbutton);  
 SharedPreferences spref = PreferenceManager.*getDefaultSharedPreferences*(this);  
 String pin = spref.getString(getString(R.string.pkey), "");  
 String userpin = pinTxt.getText().toString();  
  
 // XXX Easter Egg?  
 if (userpin.equals(pin)) {  
 // Display the private notes  
 ListView lview = (ListView) findViewById(R.id.aci3nlistView);  
 Cursor cr = getContentResolver().query(NotesProvider.*CONTENT\_URI*, new String[] {"\_id", "title", "note"}, null, null, null);  
 String[] columns = {NotesProvider.*C\_TITLE*, NotesProvider.*C\_NOTE*};  
 int [] fields = {R.id.title\_entry, R.id.note\_entry};  
 SimpleCursorAdapter adapter = new SimpleCursorAdapter(this, R.layout.notes\_entry ,cr, columns, fields, 0);  
 lview.setAdapter(adapter);  
 pinTxt.setVisibility(View.*INVISIBLE*);  
 abutton.setVisibility(View.*INVISIBLE*);  
 //cr.close();  
  
 }  
 else {  
 Toast.makeText(this, "Please Enter a valid pin!", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 }  
}

Приложение позволяет устанавливать/изменять пин-код для доступа к заметкам и получать доступ ним, введя пин-код.

1. Очевидно, что можно сначала просто изменить пин-код, а затем получить по нему данные.

Это можно рассмотреть как **CWE-306: Missing Authentication for Critical Function** (отсутствие аутентификации для функций, требующих доказуемой идентификации пользователя).

Решение: введение механизма аутентификации по действующему пин-коду для его изменения.

Добавим в layout файл вместо одного поля ввода два – для текущего и нового пин-кода:

<EditText  
android:layout\_width="match\_parent"  
android:layout\_height="wrap\_content"  
android:id="@+id/aci3CurrentPin"  
android:hint="Текущий ПИН"  
android:textAlignment="center"  
android:inputType="numberPassword"  
android:maxLength="4" />  
  
<EditText  
android:layout\_width="match\_parent"  
android:layout\_height="wrap\_content"  
android:id="@+id/aci3NewPin"  
android:hint="Новый ПИН"  
android:textAlignment="center"  
android:inputType="numberPassword"  
android:maxLength="4" />

Изменим метод addPin, заменив старые переменные для поля ввода на переменные новых полей ввода и добавив проверку на совпадение пин-кода, заменяя старую логику:

public void addPin(View view) {  
 SharedPreferences spref = PreferenceManager.*getDefaultSharedPreferences*(this);  
 SharedPreferences.Editor spedit = spref.edit();  
 EditText currentPinTxt = findViewById(R.id.aci3CurrentPin);  
 EditText newPinTxt = findViewById(R.id.aci3NewPin);  
 String currentPin = currentPinTxt.getText().toString();  
 String newPin = newPinTxt.getText().toString();  
 String storedPin = spref.getString(getString(R.string.pkey), "");  
  
 if (currentPin.isEmpty() || newPin.isEmpty()) {  
 Toast.makeText(this, " Please Enter a valid pin!", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 if (currentPin.equals(storedPin)) {  
 spedit.putString(getString(R.string.pkey), newPin);  
 spedit.commit();  
 Toast.makeText(this, "PIN updated", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.makeText(this, "Current PIN is incorrect", Toast.*LENGTH\_SHORT*).show();  
 }  
}

Также, соответственно, добавим в метод accessNotes вместо старых элементов новые:

public void accessNotes(View view) {  
 EditText currentPinTxt = findViewById(R.id.aci3CurrentPin);  
 EditText newPinTxt = findViewById(R.id.aci3NewPin);

Button abutton = (Button) findViewById(R.id.aci3naccessbutton);

...

if (userpin.equals(pin)) {

...  
 SimpleCursorAdapter adapter = new SimpleCursorAdapter(this, R.layout.notes\_entry ,cr, columns, fields, 0);  
 lview.setAdapter(adapter);  
 currentPinTxt.setVisibility(View.*NVISIBLE*);  
 newPinTxt.setVisibility(View.*INVISIBLE*);  
 abutton.setVisibility(View.*INVISIBLE*);

//cr.close();  
 }

...  
}

1. **CWE-732: Incorrect Permission Assignment for Critical Resource** (неправильное назначение разрешений для критического ресурса так, что этот ресурс может быть прочитан или изменен непреднамеренными участниками).

Доступ к заметкам происходит напрямую в коде к файлу этих заметок из ContentProvider через NotesProvider.CONTENT\_URI:

Cursor cr = getContentResolver().query(NotesProvider.*CONTENT\_URI*, new String[] {"\_id", "title", "note"}, null, null, null);

Доступ к данным может быть получен из других приложений по URI.

AndroidManifest с возможным к NotesProvider доступом из других приложений:

<provider  
 android:name=".NotesProvider"  
 android:authorities="jakhar.aseem.diva.provider.notesprovider"  
 android:enabled="true"  
 android:exported="true" >  
</provider>

NotesProvider с доступом по URI:

static final String *AUTHORITY* = "jakhar.aseem.diva.provider.notesprovider";  
static final Uri *CONTENT\_URI* = Uri.*parse*("content://" + *AUTHORITY* + "/" + *TABLE*);  
  
static final UriMatcher *urimatcher*;  
static {  
 *urimatcher* = new UriMatcher(UriMatcher.*NO\_MATCH*);  
 *urimatcher*.addURI(*AUTHORITY*, *TABLE*, *PATH\_TABLE*);  
 *urimatcher*.addURI(*AUTHORITY*, *TABLE* + "/#", *PATH\_ID*);

}

Решение: установка доступа к Content Provider только из данного приложения или использование проверки разрешений в нем.

Изменим в AndroidManifest.xml доступ из других приложений к Content Provider:

<provider  
...  
 android:exported="false" >  
</provider>

Без изменений доступа извне можно также добавить проверки разрешений для методов ContentProvider, таких как запрос (query), вставка (insert), обновление (update), удаление (delete).

Опишем в AndtoidManifest разрешения:

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="jakhar.aseem.diva" >

...

<permission  
 android:name="jakhar.aseem.diva.provider.READ\_NOTE\_PERMISSION"  
 android:protectionLevel="normal" />  
  
<permission  
 android:name="jakhar.aseem.diva.provider.READ\_NOTES\_PERMISSION"  
 android:protectionLevel="normal" />  
  
<permission  
 android:name="jakhar.aseem.diva.provider.WRITE\_NOTE\_PERMISSION"  
 android:protectionLevel="normal" />

...

<provider  
...

<meta-data  
 android:name="jakhar.aseem.diva.provider.READ\_NOTE"  
 android:permission="jakhar.aseem.diva.provider.READ\_NOTE\_PERMISSION" />  
  
<meta-data  
 android:name="jakhar.aseem.diva.provider.READ\_NOTES"  
 android:permission="jakhar.aseem.diva.provider.READ\_NOTES\_PERMISSION" />  
  
<meta-data  
 android:name="jakhar.aseem.diva.provider.WRITE\_NOTE"  
 android:permission="jakhar.aseem.diva.provider.WRITE\_NOTE\_PERMISSION" />  
  
...  
  
</provider>

...

</manifest>

Добавим проверки в NotesProvider:

@Override  
public Cursor query(Uri uri, String[] projection, String selection, String[]

...

int match = urimatcher.match(uri);  
 switch (match) {  
 case PATH\_TABLE:  
 // Разрешение для чтения всех записей  
 if (getContext().checkCallingOrSelfPermission("jakhar.aseem.diva.provider.READ\_NOTES") == PackageManager.PERMISSION\_DENIED) {  
 throw new SecurityException("Permission denied for reading all notes");  
 }  
 break;  
 case PATH\_ID:  
 // Разрешение для чтения конкретной записи по ID  
 if (getContext().checkCallingOrSelfPermission("jakhar.aseem.diva.provider.READ\_NOTE") == PackageManager.PERMISSION\_DENIED) {  
 throw new SecurityException("Permission denied for reading specific note");  
 }  
 break;  
 default:  
 throw new IllegalArgumentException("Unknown URI: " + uri);  
 }

...

}

@Override  
public Uri insert(Uri uri, ContentValues values) {  
...

if (getContext().checkCallingOrSelfPermission("jakhar.aseem.diva.provider.WRITE\_NOTE") == PackageManager.PERMISSION\_DENIED) {  
 throw new SecurityException("Permission denied for inserting a note");  
 }

...  
}

@Override  
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {  
...

if (getContext().checkCallingOrSelfPermission("jakhar.aseem.diva.provider.WRITE\_NOTE") == PackageManager.PERMISSION\_DENIED) {  
 throw new SecurityException("Permission denied for updating a note");  
 }

...  
}

@Override  
public int delete(Uri uri, String selection, String[] selectionArgs) {  
...

if (getContext().checkCallingOrSelfPermission("jakhar.aseem.diva.provider.WRITE\_NOTE") == PackageManager.PERMISSION\_DENIED) {  
 throw new SecurityException("Permission denied for deleting a note");  
 }

...  
}