2021 하계 해킹캠프 CTF

Basic Android write up

by arrester (Demon Team)

작성자: 김주원(arrester)

arresterloyal@gmail.com

https://github.com/Demon-KR

https://blog.naver.com/lstarrlodyl

목 차

2021 하계 해킹캠프 CTF

Ⅰ 문제 개요 및 목표

Ⅰ-1. 문제 개요

안드로이드 앱 기초를 익힐 수 있는 문제!

문제 제목: Basic Android

문제 내용: 해킹캠프에서 안드로이드 기초를 알아가 보자! 안드로이드 앱 취약점 진단할 때 가장 먼저 보는 부분이자 개발자들이 많이 실수하는 부분을 찾고 안드로이드 구조를 익히고 flag를 찾아내자!

문제 출제자: arrester

Ⅰ-2. 문제 목표

앱에 flag 3개가 존재한다. 3개의 flag를 찾아서 순서에 맞추면 된다.

2021 하계 해킹캠프 CTF

Ⅱ 문제 풀이

Ⅱ-1. first flag

그림 1 BasicAndroid.apk MainActivity

Nox를 통해 앱을 실행하면 [그림 1]과 같다. 문제에서 힌트를 제시하고 있다. 앱에 flag가 3개 존재한다고 되어있고 FLAG Button이 있는 것을 볼 수 있다.

그림 2 FLAG Button

버튼을 누르면 "good find flag"라는 Toast 메시지가 출력되는 것을 확인할 수 있다.

문제 내용에서 제시한 것을 다시 보면 "안드로이드 앱 취약점 진단할 때 가장 먼저 보는 부분이자 개발자들이 많이 실수하는 부분을 찾고"라고 되어있다. 안드로이드 앱 취약점 진단할 때 가장 먼저 보는 부분은 logcat 정보이다.

logcat은 앱에서 진행되는 로그 정보들이 기록되는 곳인데 개발자들이 개발할 때 어디까지 코드가 정상적으로 진행되고 있는지 확인 용도로 logcat에 기록한다. 향후 앱을 출시할 때 이 log 정보들을 지워야 하는데 지우지 않고 출시되는 경우들이 있다.

그래서 우리는 logcat 정보를 확인하면 된다. 여기서 만약 위 정보를 파악하지 못했다면 우선 앱의 기본적인 리버스 엔지니어링 관점으로 jadx-gui와 같은 도구를 활용하여 앱을 뜯어서 소스 코드를 확인해야 한다.

코드 1 MainActivity Code

```
package demon.arrester.basicandroid;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    String first_flag = "";
    Button hcamp_btn;
    DatabaseReference mDBReference = null;
    /* access modifiers changed from: protected */
    @Override // androidx.activity.ComponentActivity,
androidx.core.app.ComponentActivity, androidx.appcompat.app.AppCompatActivity,
androidx.fragment.app.FragmentActivity
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_main);
        Button button = (Button) findViewById(R.id.hcamp_btn);
        this.hcamp_btn = button;
        button.setOnClickListener(this);
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference();
        this.mDBReference = reference;
        reference.child("HCAMP_23").child("first_flag").addValueEventListener(new
ValueEventListener() {
            /* class demon.arrester.basicandroid.MainActivity.AnonymousClass1 */
            @Override // com.google.firebase.database.ValueEventListener
            public void onDataChange(DataSnapshot dataSnapshot) {
                MainActivity.this.first_flag = dataSnapshot.getValue().toString();
            }
            @Override // com.google.firebase.database.ValueEventListener
            public void onCancelled(DatabaseError databaseError) {
                MainActivity.this.first_flag = "error";
            }
        });
    }
    public void onClick(View view) {
        if (view.getId() == R.id.hcamp_btn) {
            Toast.makeText(this, "good find flag!", 0).show();
            Log.e("first_flag", "first flag: " + this.first_flag);
        }
    }
}
```

코드를 보고 파악할 수 있는 점을 아래와 같이 요약한다.

1. 데이터베이스로 Firebase를 사용한다.
2. first_flag를 Firebase에서 가져온다.
3. 버튼을 클릭했을 때 [그림 2]에서 본 "good find flag!" 문자열이 Toast 메시지로 출력된다.
4. 추가로 Log로도 first_flag 정보가 출력된다.

위 정보들을 활용하여 nox로 logcat 정보를 확인하자

➢ **nox_adb logcat**

위 명령어를 통해 logcat에 접속하고 [코드 1]에서 본 것과 같이 Button을 클릭했을 때 log가 출력되는 것이므로 버튼을 클릭하여 확인하면 된다.

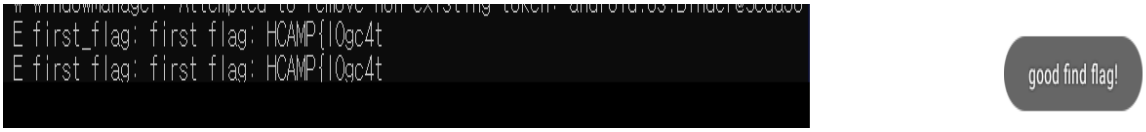

그림 3 first flag check

[그림 3]과 같이 logcat에 flag가 출력된 것을 확인할 수 있다.

Ⅱ-2. second flag

[코드 1]에서 jadx-gui를 통해 앱 내부를 확인했다면 앱에서는 접근할 수 없던 액티비티가 있는 것을 볼 수 있다. 바로 SecretActivity다.

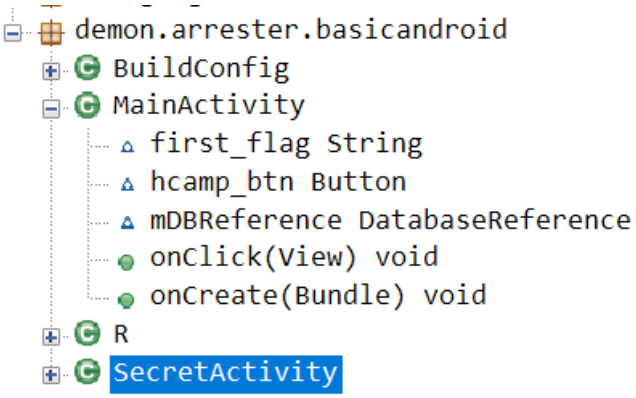

그림 4 SecretActivity find

[그림 4]와 같이 SecretActivity를 확인할 수 있고 first flag를 찾은 것과 동일하게 코드를 확인하면 된다.

코드 2 SecretActivity Code

```
package demon.arrester.basicandroid;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.io.InputStream;
public class SecretActivity extends AppCompatActivity implements View.OnClickListener {
    Button final_btn;
    String flag = "";
    DatabaseReference mDBReference = null;
    TextView secret_msg;
    String third_flag = "";
    /* access modifiers changed from: protected */
    @Override // androidx.activity.ComponentActivity,
androidx.core.app.ComponentActivity, androidx.appcompat.app.AppCompatActivity,
androidx.fragment.app.FragmentActivity
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_secret);
        this.secret_msg = (TextView) findViewById(R.id.secret_msg);
        Button button = (Button) findViewById(R.id.final_btn);
        this.final_btn = button;
        button.setOnClickListener(this);
        try {
            InputStream openRawResource = getResources().openRawResource(R.raw.flag);
            byte[] bArr = new byte[openRawResource.available()];
            openRawResource.read(bArr);
            this.flag = new String(bArr);
        } catch (Exception unused) {
            this.flag = "fail";
        }
        this.secret_msg.setText("find second flag");
        DatabaseReference reference = FirebaseDatabase.getInstance().getReference();
        this.mDBReference = reference;
        reference.child("HCAMP_23").child("third_flag").addValueEventListener(new
ValueEventListener() {
            /* class demon.arrester.basicandroid.SecretActivity.AnonymousClass1 */
            @Override // com.google.firebase.database.ValueEventListener
            public void onDataChange(DataSnapshot dataSnapshot) {
                SecretActivity.this.third_flag = dataSnapshot.getValue().toString();
            }
            @Override // com.google.firebase.database.ValueEventListener
            public void onCancelled(DatabaseError databaseError) {
                SecretActivity.this.third_flag = "error";
            }
        });
    }
    public void onClick(View view) {
        if (view.getId() == R.id.final_btn) {
            Toast.makeText(this, "third flag: " + this.third_flag, 0).show();
        }
    }
}
```

코드를 보고 파악할 수 있는 점을 아래와 같이 요약한다.

1. second flag는 없고 third flag만 Firebase를 통해 얻어온다.
2. second flag로 보이는 부분으로 R.raw.flag에서 값을 가져온다.
3. second flag를 가져오기만 하고 출력하고 있지는 않다. 즉, R.raw.flag 위치를 찾아가야 한다.
4. third flag의 경우 해당 액티비티에서 버튼을 클릭하면 flag가 출력된다.

얻게 된 정보를 분석하면 R.raw.flag 경로를 찾아야 한다. R은 안드로이드에서 Resource를 뜻하고 raw는 해당 디렉토리 이름이다. jadx-gui를 통해 해당 경로로 직접 접근할 수 있다.

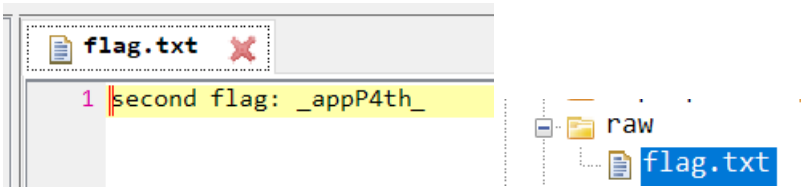

그림 5 second flag check

이렇게 두 번째 flag 정보까지 획득할 수 있다.

Ⅱ-3. third flag

세 번째 flag의 경우 second flag를 확인할 때 같은 소스 코드에서 확인을 했다. 해당 액티비티는 MainActivity에서는 접근할 수 없다. 그러면 어떻게 접근해야 할까? 안드로이드에서 해당 앱에 접근이 허용되어 있는지 파악할 수 있는 곳이 있다.

바로 AndroidManifest.xml이다.

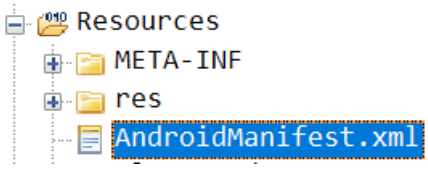

그림 6 AndroidManifest.xml

AndroidManifest도 마찬가지로 Resource에서 찾을 수 있다.

코드 3 AndroidManifest.xml Code

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="1" android:versionName="1.0" android:compileSdkVersion="31"
android:compileSdkVersionCodename="12" package="demon.arrester.basicandroid"
platformBuildVersionCode="31" platformBuildVersionName="12">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="31"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission
android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE"/>
    <application android:theme="@style/Theme.BasicAndroid"
android:label="@string/app_name" android:icon="@mipmap/trip_android"
android:allowBackup="true" android:supportsRtl="true"
android:roundIcon="@mipmap/trip_android"
android:appComponentFactory="androidx.core.app.CoreComponentFactory">
        <activity android:name="demon.arrester.basicandroid.SecretActivity"
android:exported="true"/>
        <activity android:name="demon.arrester.basicandroid.MainActivity"
android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <service android:name="com.google.firebase.components.ComponentDiscoveryService"
android:exported="false" android:directBootAware="true">
            <meta-data
android:name="com.google.firebase.components:com.google.firebase.database.DatabaseRegist
rar" android:value="com.google.firebase.components.ComponentRegistrar"/>
            <meta-data
android:name="com.google.firebase.components:com.google.firebase.analytics.connector.int
ernal.AnalyticsConnectorRegistrar"
android:value="com.google.firebase.components.ComponentRegistrar"/>
            <meta-data
android:name="com.google.firebase.components:com.google.firebase.installations.FirebaseI
nstallationsRegistrar"
android:value="com.google.firebase.components.ComponentRegistrar"/>
        </service>
        <receiver
android:name="com.google.android.gms.measurement.AppMeasurementReceiver"
android:enabled="true" android:exported="false"/>
        <service android:name="com.google.android.gms.measurement.AppMeasurementService"
android:enabled="true" android:exported="false"/>
        <service
android:name="com.google.android.gms.measurement.AppMeasurementJobService"
android:permission="android.permission.BIND_JOB_SERVICE" android:enabled="true"
android:exported="false"/>
        <provider android:name="com.google.firebase.provider.FirebaseInitProvider"
android:exported="false"
android:authorities="demon.arrester.basicandroid.firebaseinitprovider"
android:initOrder="100" android:directBootAware="true"/>
        <activity android:theme="@style/Theme.Translucent.NoTitleBar"
android:name="com.google.android.gms.common.api.GoogleApiActivity"
android:exported="false"/>
        <meta-data android:name="com.google.android.gms.version"
android:value="@integer/google_play_services_version"/>
    </application>
</manifest>
```

코드를 보고 파악할 수 있는 점을 아래와 같이 요약한다.

1. SecretActivity에 exported가 true로 허용되어 있다.

즉, 외부에서 해당 액티비티로 바로 접근할 수 있다는 이야기가 된다.

접근하는 방법은 다양하다.

1. nox_adb 명령
2. 안드로이드 취약점 분석 도구를 활용(ex. drozer)

본 풀이에서는 기본 nox로 접근하는 방법으로 진행한다.

nox_adb shell로 접근 후 아래의 명령어를 입력하면 된다.

- **am start demon.arrester.basicandroid/.SecretActivity**

그림 7 third flag check

그러면 정상적으로 nox에서 SecretActivity에 접근된 것을 확인할 수 있고 [코드 2]에서 본 내용과 동일하게 버튼을 클릭하면 [그림 7]과 같이 세 번째 flag 정보를 획득할 수 있다.

최종 flag: HCAMP{l0gc4t_appP4th_4ctiv1ty}

2021 하계 해킹캠프 CTF

Ⅲ 결론

안드로이드 앱에 대한 기초를 공부할 수 있도록 만든 문제입니다.
각 문제 의도는 아래와 같습니다.

first flag 의도: 개발자들이 개발 중간 코드 실행 결과를 확인하고자 log 정보로 진행 상황 체크를 하는 경우가 많다. 이후 출시할 때 이것을 제거하지 않아서 생기는 문제를 알아가길 바랍니다.

second flag 의도: 안드로이드 앱 내에 경로 관련 구조 이해

third flag 의도: 액티비티를 직접 실행해서 열어볼 수 있다는 점으로 막힌 액티비티 우회 가능 (feat. exported=true)

by. Demon Team arrester