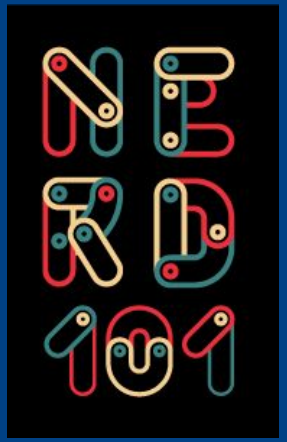
Introduccion al pentest de Android Apps

Nerdearla 101v2 Workshop

01 Temario

- ¿Quién soy?
- Objetivos
- Introducción al formato del apk
- Introducción a reversing de apps (layer Java)
- Metodología de pruebas de apps
  - Bugs de insecure communication
  - Bugs de persistencia de datos
  - Bugs de IPC
  - WebViews
- Introducción a tools más importantes
  - jadx
  - adb
  - Frida
- Próximos pasos
- Recursos interesantes
- Preguntas?

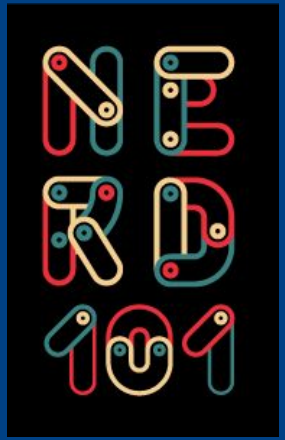
01 Quien soy?



Cesar Rodriguez - hohenhaimmaster@gmail.com - @warlockk87

- Redteam TL - Kavak
- Especialista en Mobile Application Security
- Blogger en: https://cmrodriguez.me
- Streamer en: https://www.twitch.tv/ageofentropy
  - https://www.youtube.com/channel/UCitiWg5p-R6QNLPuJPSE33g
- Jugador de Tabletop RPG
  - Shadowrun, D&D, WH40K, Kindred of the east
- Videogame player
  - Diablo Immortals (Necromancer p7)
  - Wild Rift (Platinum)

02 Objetivos

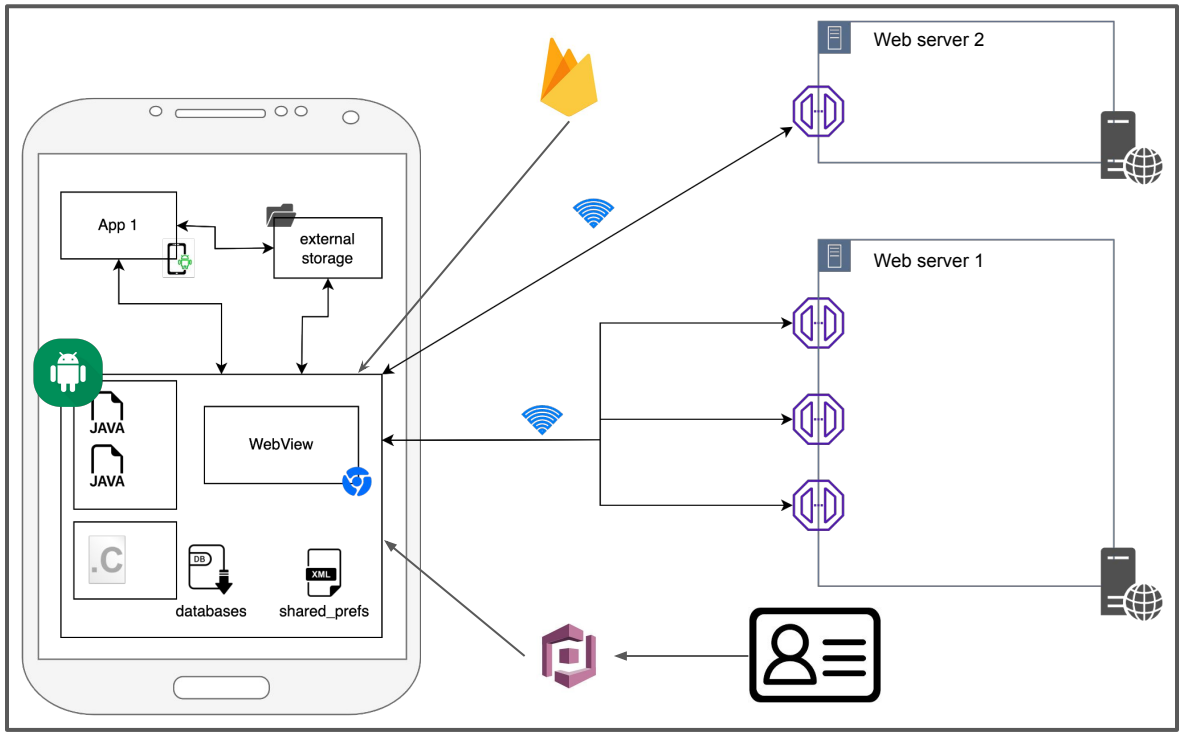


- Orientar a los que quieren empezar en la seguridad de apps Android
- Enfoque práctico
- Acercar al desarrollador mobile a los conceptos de seguridad de las apps de Android.
- Motivar para que mas gente se meta en el mundo de AppSec

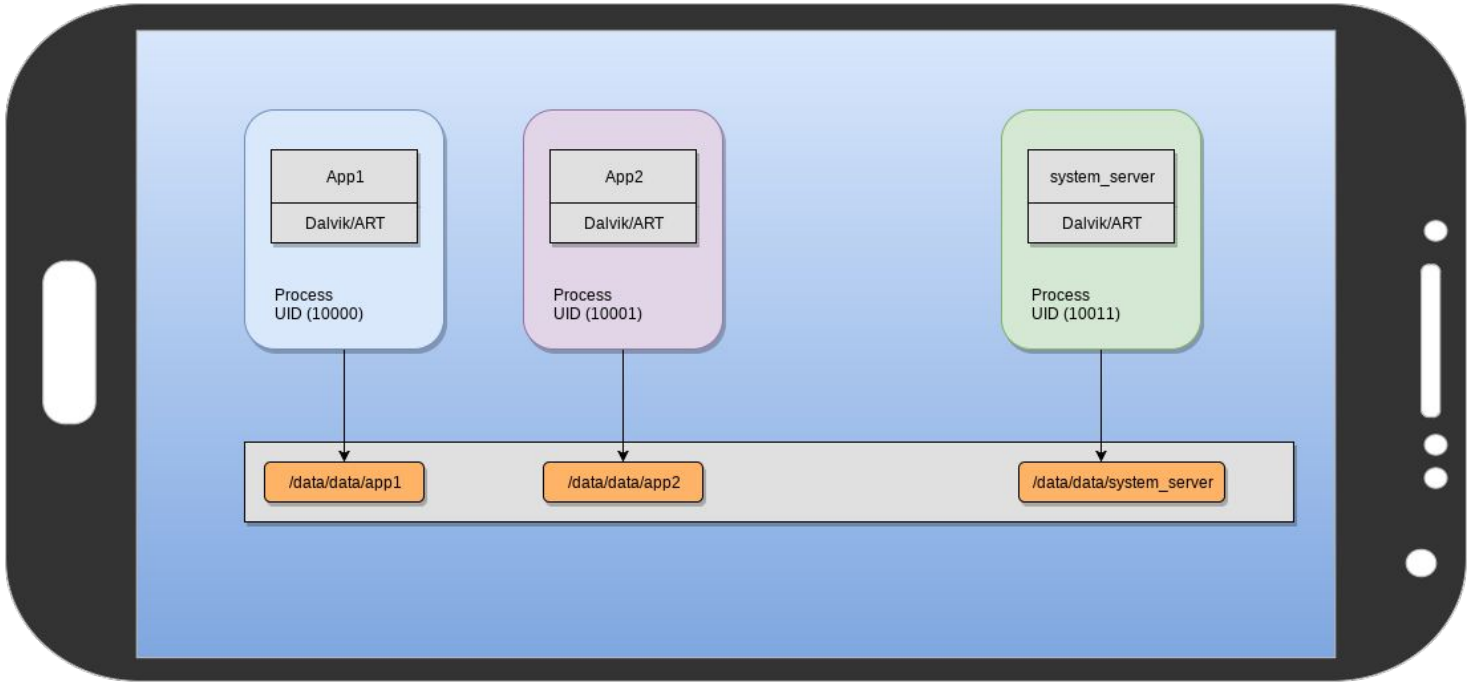
Metodología del Workshop

- Presentación teórica del tema
- Muestra de concepto en forma práctica
  - Hagamos el ejercicio en paralelo :)
- Links para profundizar en el tema
- Breve break de preguntas del tema específico
  - Max 2 preguntas.

03 Introducción al formato de APK



03 Introducción al formato de APK



03 Primeros pasos

1) Instalar aplicación en celular/emulador:

```
adb install <path_folder_app>/androgoat.apk
```

2) Ingresar a la terminal del dispositivo:

```
adb root
adb shell
```

3) Navegar la aplicación

```
pm list packages
ls -al /data/data
cd /data/data/owasp.sat.agoat/
```

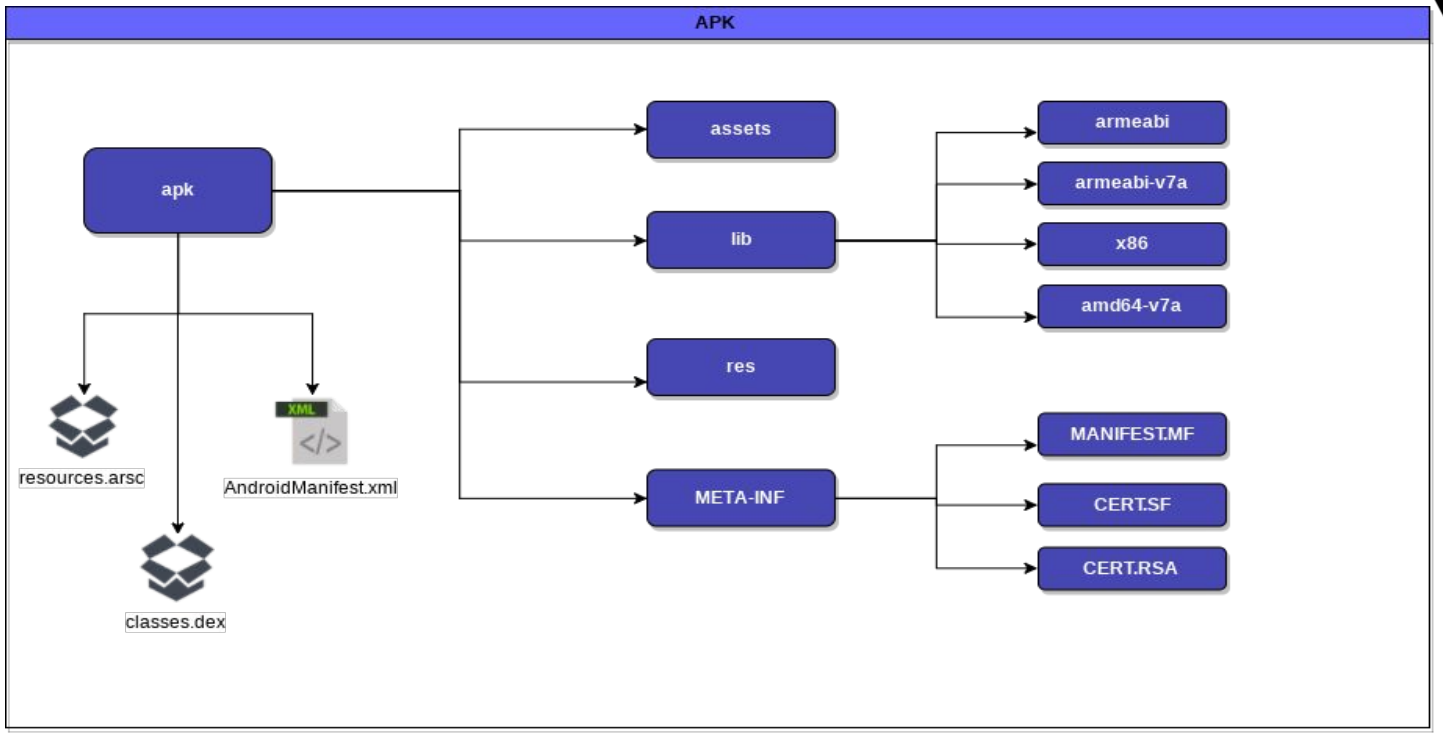
4) Ver contenido de carpetas de aplicación

```
ls -al /data/app/owasp.sat.agoat*/
ls -al /sdcard/
```

```
generic_arm64:/data/data/owasp.sat.agoat # ls
cache  code_cache  shared_prefs
generic_arm64:/data/data/owasp.sat.agoat # cd shared_prefs/
generic_arm64:/data/data/owasp.sat.agoat/shared_prefs # ls
generic_arm64:/data/data/owasp.sat.agoat/shared_prefs # cd ..
generic_arm64:/data/data/owasp.sat.agoat # ls
cache  code_cache  shared_prefs
generic_arm64:/data/data/owasp.sat.agoat # ls -al
total 24
drwx------  5 u0_a61 u0_a61       4096 2022-07-23 20:06 .
drwxrwx--x 81 system system       4096 2022-07-23 20:01 ..
drwxrws--x  2 u0_a61 u0_a61_cache 4096 2022-07-23 20:01 cache
drwxrws--x  2 u0_a61 u0_a61_cache 4096 2022-07-23 20:01 code_cache
drwxrwx--x  2 u0_a61 u0_a61       4096 2022-07-23 20:06 shared_prefs
generic_arm64:/data/data/owasp.sat.agoat # ls -al /data/app/owasp.sat.agoat*/
total 2720
drwxr-xr-x 4 system system     4096 2022-07-23 20:01 .
drwxrwx--x 7 system system     4096 2022-07-23 20:01 ..
-rw-r--r-- 1 system system  2765026 2022-07-23 20:01 base.apk
drwxr-xr-x 2 system system     4096 2022-07-23 20:01 lib
drwxrwx--x 3 system install    4096 2022-07-23 20:01 oat
generic_arm64:/data/data/owasp.sat.agoat # ls -al /sdcard
lrwxrwxrwx 1 root root 21 1969-12-31 21:00 /sdcard -> /storage/self/primary
generic_arm64:/data/data/owasp.sat.agoat # ls -al /sdcard/
total 48
drwxrwx--x 12 root sdcard_rw 4096 2022-07-23 20:00 .
drwx--x--x  4 root sdcard_rw 4096 2022-07-23 20:00 ..
drwxrwx--x  2 root sdcard_rw 4096 2022-07-23 20:00 Alarms
drwxrwx--x  3 root sdcard_rw 4096 2022-07-23 20:00 Android
drwxrwx--x  2 root sdcard_rw 4096 2022-07-23 20:00 DCIM
drwxrwx--x  2 root sdcard_rw 4096 2022-07-23 20:00 Download
drwxrwx--x  2 root sdcard_rw 4096 2022-07-23 20:00 Movies
drwxrwx--x  2 root sdcard_rw 4096 2022-07-23 20:00 Music
drwxrwx--x  2 root sdcard_rw 4096 2022-07-23 20:00 Notifications
drwxrwx--x  2 root sdcard_rw 4096 2022-07-23 20:00 Pictures
drwxrwx--x  2 root sdcard_rw 4096 2022-07-23 20:00 Podcasts
drwxrwx--x  2 root sdcard_rw 4096 2022-07-23 20:00 Ringtones
generic_arm64:/data/data/owasp.sat.agoat #
```

NERDEARLA

04 Introducción al formato de APK



04 Introducción al formato de APK

Android Manifest contiene todos los componentes de la aplicación. Es fundamental para entender configuraciones de seguridad del OS en la aplicacion, que componentes se pueden llamar desde otras aplicaciones y los permisos que pide la aplicacion para funcionar.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="owasp
    <uses-sdk android:minSdkVersion="18" android:targetSdkVersion="26"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debuggab
        <activity android:name="owasp.sat.agoat.SplashActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:label="@string/app_name" android:name="owasp.sat.agoat.MainActivity"/>
```

```
<activity android:label="@string/oscmd" android:name="owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity"/>
<service android:name="owasp.sat.agoat.DownloadInvoiceService" android:enabled="true" android:exported="true"/>
<activity android:label="@string/BinaryPatching" android:name="owasp.sat.agoat.BinaryPatchingActivity"/>
```

```
</activity>
<receiver android:name="owasp.sat.agoat.ShowDataReceiver" android:enabled="true" android:exported="true"/>
<activity android:label="@string/hardcode" android:name="owasp.sat.agoat.HardCodeActivity"/>
<activity android:label="@string/sql" android:name="owasp.sat.agoat.InsecureStorageSQLiteActivity"/>
```

NERDEARLA

05 Reverse Engineering

1) Unzip de aplicacion

```
unzip androgoat.apk -d androgoat_unzipped
```

2) Ver archivo AndroidManifest

```
cd androgoat_unzipped
cat AndroidManifest.xml
```

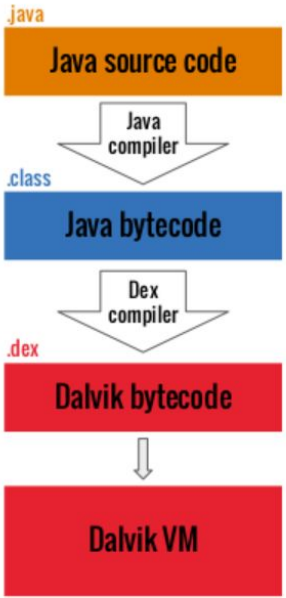
3) Usar apktool para decompilar la app

```
apktool d androgoat.apk -o androgoat_apktool
cd androgoat_apktool
cat AndroidManifest.xml
```

4) Abrir el aplicativo con jadx-gui y ver el contenido

```
jadx-gui androgoat.apk
```

05 Reverse Engineering




```
.method public static a()Z
    .locals 7

    const/4 v0, 0x0

    const-string v1, "PATH"

    invoke-static {v1}, Ljava/lang/System;->getenv(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v1

    const-string v2, ":"

    invoke-virtual {v1, v2}, Ljava/lang/String;->split(Ljava/lang/String;)[Ljava/lang/String;

    move-result-object v2

    array-length v3, v2

    move v1, v0

    :goto_0
    if-ge v1, v3, :cond_0

    aget-object v4, v2, v1

    new-instance v5, Ljava/io/File;

    const-string v6, "su"

    invoke-direct {v5, v4, v6}, Ljava/io/File;-><init>(Ljava/lang/String;Ljava/lang/String;)V

    invoke-virtual {v5}, Ljava/io/File;->exists()Z

    move-result v4

    if-eqz v4, :cond_1

    const/4 v0, 0x1

    :cond_0
    return v0

    :cond_1
    add-int/lit8 v1, v1, 0x1

    goto :goto_0
.end method
```

```
public static boolean a() {
    for (String file : System.getenv("PATH").split(":")) {
        if (new File(file, "su").exists()) {
            return true;
        }
    }
    return false;
}

public static boolean b() {
    String str = Build.TAGS;
    return str != null && str.contains("test-keys");
}

public static boolean c() {
    for (String file : new String[]{"/system/app/Superuse
        if (new File(file).exists()) {
            return true;
        }
    }
    return false;
}
```

06 Metodología de pruebas

1. Conseguir aplicacion.
2. Instalar aplicación.
    a. Ver minSdk de app y de entorno donde va a correr.
    b. Ver arquitectura de app.
3. Análisis estático básico de app
    a. Leer AndroidManifest para sacar componentes claves.
    b. Detectar bugs simples (allowBackup, NSC, cleartextEnabled…)
    c. Analizar componentes exportados (verificar deeplinks, parámetros de intents…)
    d. Ver librerías de terceros instaladas y lugares típicos de credenciales hardcodeadas.
4. Análisis dinámico básico de app
    a. Correr el aplicativo
    b. Bypass de controles anti-tampering (VM, root, cert pinning…)
    c. Aprender funcionamiento de app (pidcat <app> encendido)
5. Ejecución de pruebas sobre vulnerabilidades comunes:
    a. Bugs de persistencia de datos
    b. Bugs de IPC
    c. Bugs de insecure communication
    d. WebViews
    e. otros

07 Bypass control de root

Control de root: Control que verifica si el dispositivo tiene un firmware en el que algún componente tiene acceso al usuario root. En general el rooteo es generado por un usuario que quiere acceder a todas las funcionalidades del OS, o por un security researcher para acceder a la información interna de las aplicaciones y a funcionalidades que facilitan el entendimiento de la app.

Se puede detectar de muchas formas:

1. Existencia de paquetes o archivos particulares como
   a. /system/app/Superuser.apk
   b. eu.chainfire.supersu
2. Existencia de “su”
3. Buscar en directorios (/sbin/su, /system/su, etc)
4. Ejecutar mediante Runtime.getRuntime().exec()
5. Revisar los procesos que corren en /proc
6. Ver permisos de diferentes directorios

NERDEARLA

07 Bypass control de root

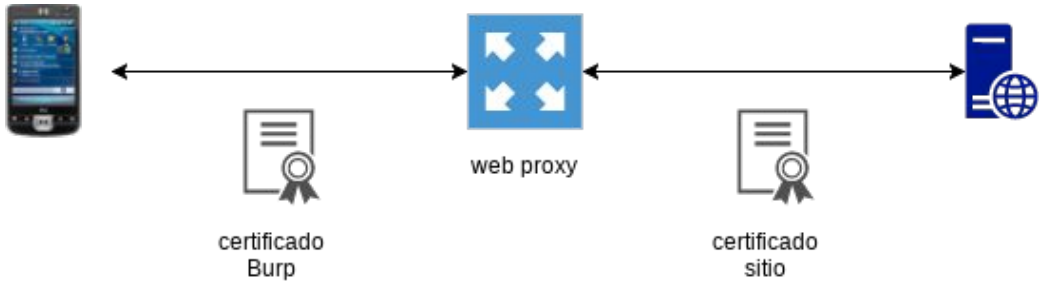
AndroidManifest.xml | RootDetectionActivity

```
        }
        View findViewById = findViewById(i);
        this._$_findViewCache.put(Integer.valueOf(i), findViewById);
        return findViewById;
    }

    /* JADX INFO: Access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.SupportAc
14  public void onCreate(Bundle savedInstanceState) {
15      super.onCreate(savedInstanceState);
16      setContentView(R.layout.activity_root_detection);
29      ((Button) findViewById(R.id.rootCheck)).setOnClickListener(new View.OnClickListener() { // from class: owasp.sat.agoat.RootD
            @Override // android.view.View.OnClickListener
18          public final void onClick(View it) {
19              if (RootDetectionActivity.this.isRooted()) {
21                  Toast.makeText(RootDetectionActivity.this.getApplicationContext(), "Device is rooted", 1).show();
                } else {
25                  Toast.makeText(RootDetectionActivity.this.getApplicationContext(), "Device is not rooted", 1).show();
                }
            }
        });
    }

33  public final boolean isRooted() {
36      boolean result = false;
37      for (String files : new String[]{"/system/app/Superuser/Superuser.apk", "/system/app/Superuser.apk", "/sbin/su", "/system/bi
39          result = new File(files).exists();
40          if (result) {
                break;
            }
        }
45      return result;
    }
```

```
1  Java.perform(function () {
2      var RootDetectionActivity = Java.use("owasp.sat.agoat.RootDetectionActivity");
3      RootDetectionActivity.isRooted.implementation = function () {
4          return false;
5      }
6  });
```

NERDEARLA

08 Interceptando trafico



Request

Raw | Params | Headers | Hex

```
POST
/safebrowsing/downloads?client=navclient-auto-ffox&appver=52.8&pver=2.2&key=no-google-api-key
HTTP/1.1
Host: safebrowsing.google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Length: 503
Content-Type: text/plain
Connection: close
Cookie:
NID=129=Eb2HW8No97VJ3NZsPjy0nOFLUtLxJToutltGZHci6LkxrYGC3rwERKh_DQT-ucamLz1URdWwtf75pdovpafewU
K-8r1up4su9cGLhFv4OsAPnzC4YqeXk11sa5i4sge6
Pragma: no-cache
Cache-Control: no-cache

goog-badbinurl-shavar;a:149095-151157:s:154185-154664,154666-154669,154671-154678,154680-15468
2,154684-154687,154689-154695,154697-154720,154722-154776,154778-155706,155708-156049,156051-1
56080,156082-156106,156108-156242
goog-phish-shavar;a:498708-509651:s:710094-760528
goog-malware-shavar;a:276258-288842:s:272375-277907,277909-281580,281582-281800,281802-283603,
283605-283927,283929-284195,284197-286469
goog-unwanted-shavar;a:105118-116810:s:100486-106780,106782-108915,108917-114813,114815-115041
```

Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Content-Type: application/vnd.google.safebrowsing-update
X-Content-Type-Options: nosniff
Date: Thu, 31 May 2018 11:36:32 GMT
Server: HTTP server (unknown)
Content-Length: 2173
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alt-Svc: quic=":443"; ma=2592000; v="43,42,41,39,35"
Connection: close

n:1757
i:goog-badbinurl-shavar
ad:149095-149330
sd:154185-154410
u:safebrowsing-cache.google.com/safebrowsing/rd/ChVnb29nLWJhZGJpbnVybC1zaGF2YXI4AEACSgwIARCQyA
kY6sgJIAFKDAgBENTHCRiOyAkgAUoMCAEQxscJGNLHCSABSgwIARDExwkYxMcJIAFKDAgBELvHCRjCxwkgAUoMCAEQ-cY
JGLnHCSABSgwIARC7xgkY98YJIAFKDAgBEOLFCRi5xgkgAUoMCAEQ08QJG0DFCSABSgwIARDLwwkYy8MJIAFKRggAEPac
CRiQoQkgASo4EW2OAZABkwGeAaMBpgHSAdgB2gHeAd8B5wGIAooCnAKxArIC2QLhAuQC9QL3AqUDpgOtA64DrwM
i:goog-malware-shavar
ad:276258-277238
sd:272375-273440
u:safebrowsing-cache.google.com/safebrowsing/rd/ChNnb29nLW1hbHdhcmUtc2hhdmFyOABAAkoMCAEQhr4RGI
nAESAB
```

08 Interceptando trafico

1. Configurando proxy
   a. Http OK
   b. Https NO OK

2. Agregando certificado API <= 23
   a. Http OK
   b. Https OK

3. Agregando certificado API > 23
   a. Http OK
   b. Https NO OK

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest >
    <application android:networkSecurityConfig="@xml/network_security_config" >
        ...
    </application>
</manifest>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config>
        <trust-anchors>
            <certificates src="system"/>
            <certificates src="user"/>
        </trust-anchors>
    </base-config>
</network-security-config>
```

NERDEARLA

08 Interceptando trafico

1) Configurar proxy en dispositivo (Burp Community)
2) Set proxy en telefono
3) Ejecutar Request HTTP desde Network Interception
4) Ejecutar Request HTTPS desde Network Interception
5) Instalar certificado como usuario en telefono

```
adb push burp.cer /sdcard
```

6) Ejecutar Request HTTPS
7) Ejecutar Request CERTIFICATE PINNING
8) Verificar como se implementa Cert Pinning en aplicacion

08 Bypass Certificate pinning

1) Identificar si hay certificate pinning

2) Ver como se implementa el certificate pinning

3) Opciones de bypass
    a) Patch aplicacion
    b) XPosed y bypass de cert pinning
    c) Frida

Con Frida

1) Lanzar frida-server: **/<path>/frida-server &**
2) Tomar el pID de la app de AndroGoat: **frida-ps -U**
3) frida -U -p **<process-id>**
4) Correr script:

```
Java.perform(function () {
        var CertPinnBuilder = Java.use("okhttp3.CertificatePinner$Builder");
        CertPinnBuilder.add.implementation = function(patter, pins) {
                return this;
        }
})
```

09 Bugs de Persistencia

En este tipo de bugs se analiza que se está almacenando por la aplicación, quién tiene acceso y si hay alguna forma de exfiltrar la información. Se encuentran bugs ejecutando pruebas y analizando el file system.

Hay distintos tipos comunes de archivos y lugares donde almacenarlos:

	Localización (por defecto)	Expuesto por defecto	allowBackup=true	WORLD_READABLE	Rooted	Content Provider	Services	BroadCast Receiver
SharedPreferences	/data/data/<apk>/shared_prefs	no	si	si	si	parc	parc	parc
DB sqlite	/data/data/<apk>/databases	no	si	si	si	parc	parc	parc
DB Realm	/data/data/<apk>/files	no	parc	si	si	parc	parc	parc
Archivo en storage interno	/data/data/<apk>/files	no	parc	si	si	parc	parc	parc
Archivo en storage externo	?	si	parc	si	si	parc	parc	parc
KeyStore	?	no	no	no	si	parc	parc	parc

09 Bugs de Persistencia

Shared Preferencess 1

1) Cargar creds en app por primera vez

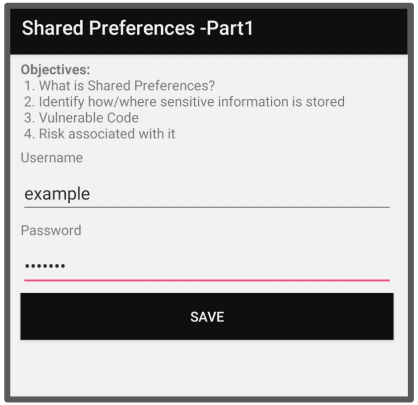
2) Buscar en la app el archivo con data sensible

```
cd /data/data/owasp.sat.agoat/
ls
cat shared_prefs/users.xml
```

```
emulator64_arm64:/data/data/owasp.sat.agoat # cat shared_prefs/users.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">example</string>
    <string name="username">example</string>
</map>
emulator64_arm64:/data/data/owasp.sat.agoat #
```

```
    @Override // android.view.View.OnClickListener
    public final void onClick(View it) {
        SharedPreferences.Editor editor = InsecureStorageSharedPrefs.this.getSharedPreferences("users", 0).edit();
        EditText editText = this.$username;
        Intrinsics.checkExpressionValueIsNotNull(editText, "username");
        editor.putString("username", editText.getText().toString());
        EditText editText2 = this.$password;
        Intrinsics.checkExpressionValueIsNotNull(editText2, "password");
        editor.putString("password", editText2.getText().toString());
        if (editor.commit()) {
            Toast.makeText(InsecureStorageSharedPrefs.this.getApplicationContext(), "Data saved", 1).show();
        } else {
            Toast.makeText(InsecureStorageSharedPrefs.this.getApplicationContext(), "Data not saved", 1).show();
        }
    }
});
```

3) Ver clase owasp.sat.agoat.InsecureStorageSharedPrefs



09 Bugs de Persistencia

SQLite

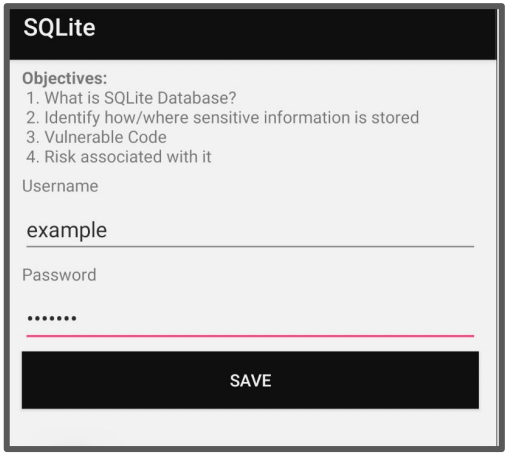
1) Cargar creds en app por primera vez

2) Buscar en la app el archivo con data sensible

```
cd /data/data/owasp.sat.agoat/databases
sqlite3 aGoat
select * from users;
```

```
emulator64_arm64:/data/data/owasp.sat.agoat/databases # ls
aGoat aGoat-journal
emulator64_arm64:/data/data/owasp.sat.agoat/databases # sqlite3 aGoat
SQLite version 3.22.0 2018-12-19 01:30:22
Enter ".help" for usage hints.
sqlite> .tables
android_metadata  users
sqlite> select * from users;
1|example|example
sqlite>
```

```
@Override // android.view.View.OnClickListener
public final void onClick(View it) {
    StringBuilder sb = new StringBuilder();
    sb.append("INSERT INTO users (username, password) VALUES('");
    EditText editText = this.$username;
    Intrinsics.checkExpressionValueIsNotNull(editText, "username");
    sb.append(editText.getText().toString());
    sb.append("','");
    EditText editText2 = this.$password;
    Intrinsics.checkExpressionValueIsNotNull(editText2, "password");
    sb.append(editText2.getText().toString());
    sb.append("')");
    String qry = sb.toString();
```



3) Ver clase owasp.sat.agoat.InsecureStorageSQLiteActivity

09 Bugs de Persistencia

External Storage

1) Cargar creds en app por primera vez

2) Buscar en la app el archivo con data sensible

cd /sdcard
cat users*tmp

```
emulator64_arm64:/sdcard # ls
Alarms Android DCIM Download Movies Music Notifications Pictures
emulator64_arm64:/sdcard # cat users*tmp
 This data is stored in SdCard on Mon Jul 25 03:27:49 GMT-03:00 1
 Username - example Password -example
emulator64_arm64:/sdcard #
```

```
@Override // android.view.View.OnClickListener
public final void onClick(View it) {
    try {
        StringBuilder sb = new StringBuilder();
        sb.append(" This data is stored in SdCard on ");
        sb.append(new Date());
        sb.append(": \n Username - ");
        EditText editText = this.$username;
        Intrinsics.checkExpressionValueIsNotNull(editText, "username");
        sb.append(editText.getText().toString());
        sb.append(" Password -");
        EditText editText2 = this.$password;
        Intrinsics.checkExpressionValueIsNotNull(editText2, "password");
        sb.append(editText2.getText().toString());
        sb.append("\n");
        String data = sb.toString();
        File externalStorageDirectory = Environment.getExternalStorageDirectory();
        Intrinsics.checkExpressionValueIsNotNull(externalStorageDirectory, "Environment.getExternalStorageDirectory()");
        File userinfo = File.createTempFile("users", "tmp", new File(externalStorageDirectory.getAbsolutePath()));
        System.out.println((Object) ("userinfo " + userinfo));
        userinfo.setReadable(true);
        userinfo.setWritable(true);
        FileWriter fw = new FileWriter(userinfo);
        fw.write(data);
        fw.close();
        Toast.makeText(InsecureStorageSDCardActivity.this.getApplicationContext(), "Data saved", 1).show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

3) Ver clase owasp.sat.agoat.InsecureStorageSDCardActivity

09 Bugs de Persistencia

External Storage

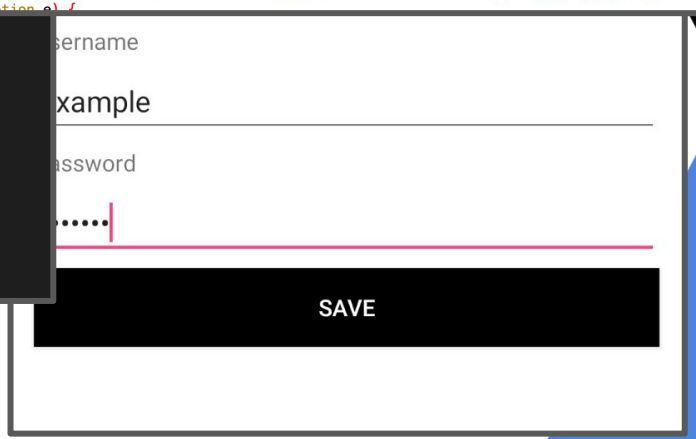
1) Cargar creds en app por primera vez

2) Buscar en la app el archivo con data sensible

cd /data/data/owasp.sat.agoat/
cat users*tmp

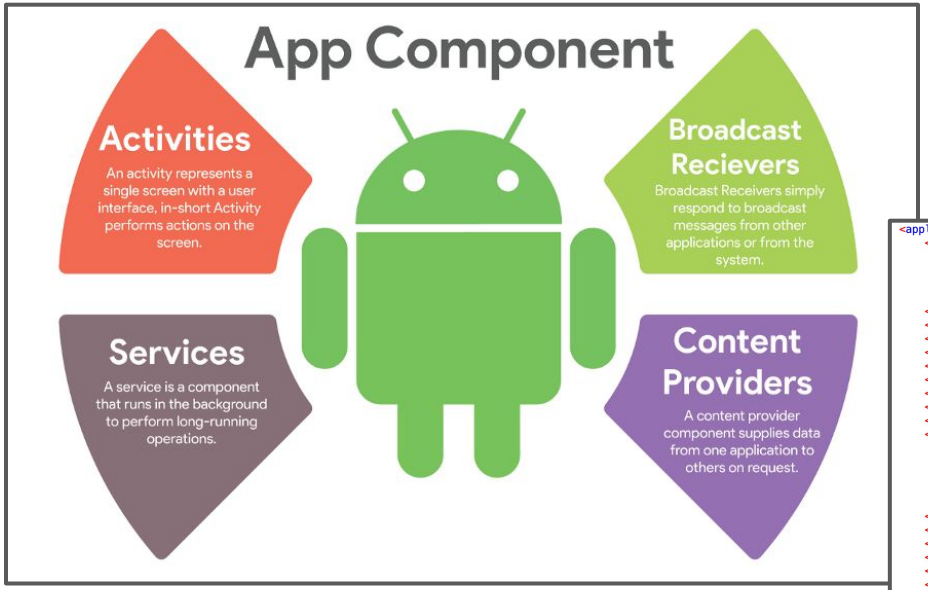
```
[1|emulator64_arm64:/sdcard # cd /data/data/owasp.sat.agoat/
[emulator64_arm64:/data/data/owasp.sat.agoat # ls
cache code_cache databases shared_prefs users2941182716961087640tmp
[emulator64_arm64:/data/data/owasp.sat.agoat # ls -al
total 40
drwx------   6 u0_a117 u0_a117       4096 2022-07-26 07:21 .
drwxrwx--x 149 system  system        8192 2022-07-25 03:04 ..
drwxrws--x   2 u0_a117 u0_a117_cache 4096 2022-07-25 03:04 cache
drwxrws--x   2 u0_a117 u0_a117_cache 4096 2022-07-25 03:04 code_cache
drwxrwx--x   2 u0_a117 u0_a117       4096 2022-07-25 03:05 databases
drwxrwx--x   2 u0_a117 u0_a117       4096 2022-07-25 03:10 shared_prefs
-rw-------   1 u0_a117 u0_a117         40 2022-07-26 07:21 users2941182716961087640tmp
emulator64_arm64:/data/data/owasp.sat.agoat #
```

```
@Override // android.view.View.OnClickListener
public final void onClick(View it) {
    try {
        File userinfo = File.createTempFile("users", "tmp", new File(InsecureStorageTempActivity.this.getApp
        userinfo.setReadable(true);
        userinfo.setWritable(true);
        FileWriter fw = new FileWriter(userinfo);
        StringBuilder sb = new StringBuilder();
        sb.append("username is ");
        EditText editText = this.$username;
        Intrinsics.checkExpressionValueIsNotNull(editText, "username");
        sb.append(editText.getText().toString());
        sb.append("\n");
        fw.write(sb.toString());
        StringBuilder sb2 = new StringBuilder();
        sb2.append("password is ");
        EditText editText2 = this.$password;
        Intrinsics.checkExpressionValueIsNotNull(editText2, "password");
        sb2.append(editText2.getText().toString());
        sb2.append("\n");
        fw.write(sb2.toString());
        fw.close();
        Toast.makeText(InsecureStorageTempActivity.this.getApplicationContext(), "Data saved", 1).show();
    } catch (Exception e) {
```



3) Ver clase owasp.sat.agoat.InsecureStorageTempActivity

10 Inter Process Communication



App aGoat: 27 componentes

```
<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debugg
    <activity android:name="owasp.sat.agoat.SplashActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
    <activity android:label="@string/app_name" android:name="owasp.sat.agoat.MainActivity"/>
    <activity android:label="@string/root" android:name="owasp.sat.agoat.RootDetectionActivity"/>
    <activity android:label="@string/logging" android:name="owasp.sat.agoat.InsecureLoggingActivity"/>
    <activity android:label="@string/xss" android:name="owasp.sat.agoat.XSSActivity"/>
    <activity android:label="@string/sqli" android:name="owasp.sat.agoat.SQLinjectionActivity"/>
    <activity android:label="@string/sp1" android:name="owasp.sat.agoat.InsecureStorageSharedPrefs"/>
    <activity android:label="@string/tempFile" android:name="owasp.sat.agoat.InsecureStorageTempActivity"/>
    <activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControlIssue1Activity"/>
    <activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
        <intent-filter>
            <action android:name="android.intent.action.VIEW"/>
            <category android:name="android.intent.category.DEFAULT"/>
            <data android:scheme="androgoat" android:host="vulnapp"/>
        </intent-filter>
    </activity>
    <receiver android:name="owasp.sat.agoat.ShowDataReceiver" android:enabled="true" android:exported="true"/>
    <activity android:label="@string/hardcode" android:name="owasp.sat.agoat.HardCodeActivity"/>
    <activity android:label="@string/sql" android:name="owasp.sat.agoat.InsecureStorageSQLiteActivity"/>
    <activity android:label="@string/sp2" android:name="owasp.sat.agoat.InsecureStorageSharedPrefs1Activity"/>
    <activity android:label="@string/network" android:name="owasp.sat.agoat.TrafficActivity"/>
    <activity android:name="owasp.sat.agoat.ContentProviderActivity"/>
    <activity android:label="@string/emulator" android:name="owasp.sat.agoat.EmulatorDetectionActivity"/>
    <activity android:label="@string/sdCard" android:name="owasp.sat.agoat.InsecureStorageSDCardActivity"/>
    <activity android:label="@string/wbviewAccess" android:name="owasp.sat.agoat.InputValidationsWebViewURLActivity"/>
    <activity android:label="@string/oscmd" android:name="owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity"/>
    <service android:name="owasp.sat.agoat.DownloadInvoiceService" android:enabled="true" android:exported="true"/>
    <activity android:label="@string/BinaryPatching" android:name="owasp.sat.agoat.BinaryPatchingActivity"/>
    <activity android:label="@string/clipboard" android:name="owasp.sat.agoat.ClipboardActivity"/>
```

NERDEARLA

10 Componentes exportados

Componente exportado: En Android los componentes exportados se pueden llamar desde otras aplicaciones. No necesariamente es una vulnerabilidad, pero abre la superficie de ataque para que una app maliciosa instalada en el celular llame al componente. La vulnerabilidad se da cuando el componente es exportado cuando no debería, y confía en los parámetros que recibe, ejecutando acciones sensibles.

```
<service android:name="owasp.sat.agoat.DownloadInvoiceService" android:enabled="true" android:exported="true"/>
<activity android:label="@string/BinaryPatching" android:name="owasp.sat.agoat.BinaryPatchingActivity"/>
<activity android:label="@string/clipboard" android:name="owasp.sat.agoat.ClipboardActivity"/>
```

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="androgoat" android:host="vulnapp"/>
    </intent-filter>
</activity>
```

```
<receiver android:name="owasp.sat.agoat.ShowDataReceiver" android:enabled="true" android:exported="true"/>
<activity android:label="@string/hardcode" android:name="owasp.sat.agoat.HardCodeActivity"/>
<activity android:label="@string/sql" android:name="owasp.sat.agoat.InsecureStorageSQLiteActivity"/>
```

```
<activity android:name="owasp.sat.agoat.SplashActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```

NERDEARLA

10 Recibiendo Parametros

```
public void onClick(View view) {
    Intent i = new Intent(this, ActivityTwo.class);
    i.putExtra("Value1", "This value one for ActivityTwo ");
    i.putExtra("Value2", "This value two ActivityTwo");
    // set the request code to any code you like,
    // you can identify the callback via this code
    startActivityForResult(i, REQUEST_CODE);
}
```

```
Bundle extras = getIntent().getExtras();
if (extras == null) {
    return;
}
// get data via the key
String value1 = extras.getString(Intent.EXTRA_TEXT);
if (value1 != null) {
    // do something with the data
}
```

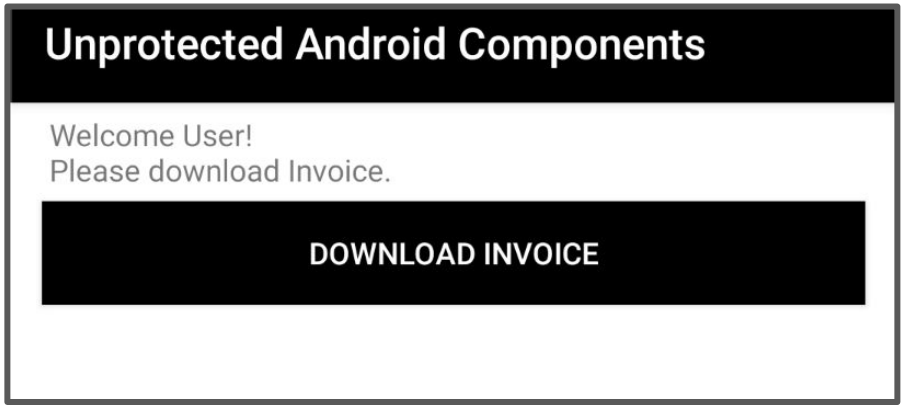
```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```

10 Usando componentes exportados

1) Llamar activity exportada

am start-activity owasp.sat.agoat/owasp.sat.agoat.AccessControl1ViewActivity

2) Trackear en el codigo que hace la aplicacion



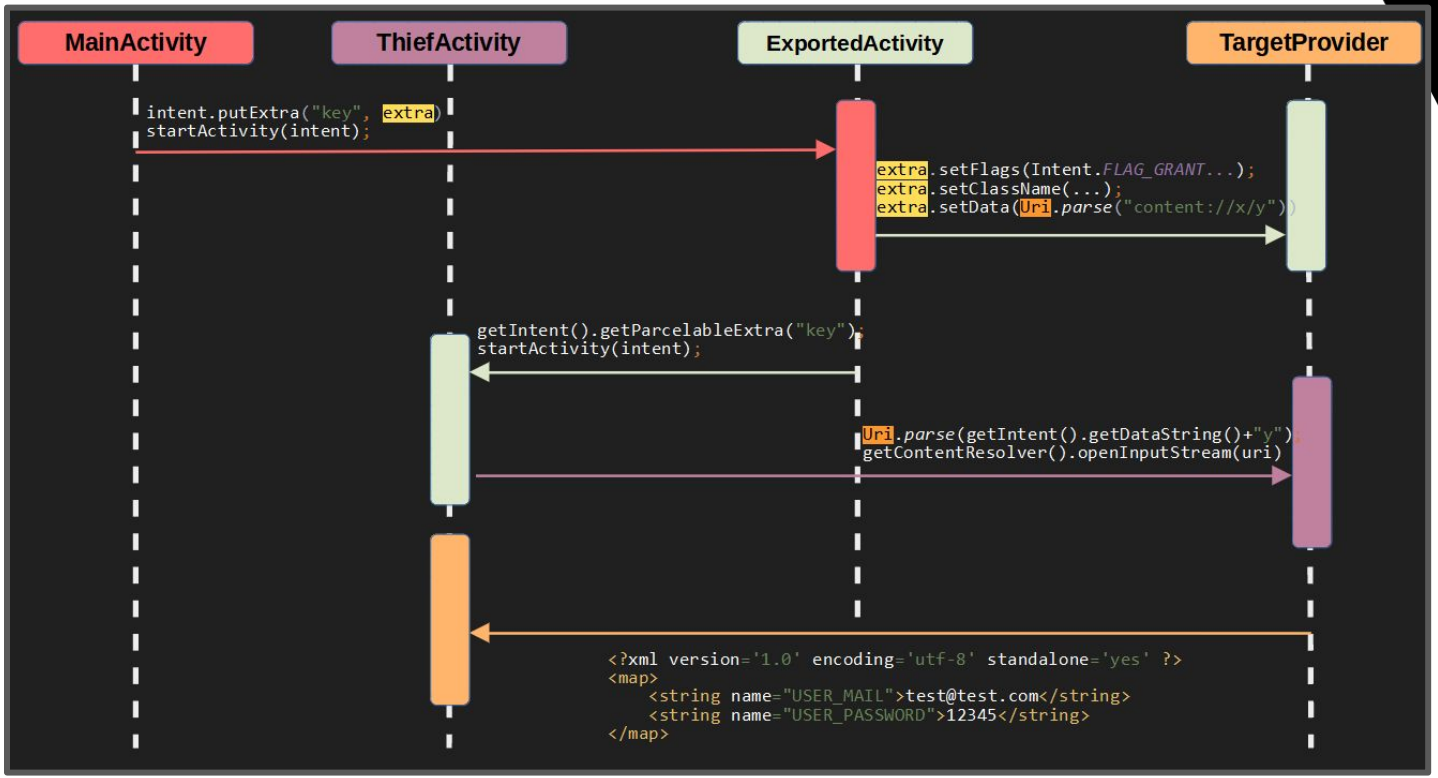
10 Explicit vs Implicit intent

```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```

```
// Create the text message with a string.
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Try to invoke the intent.
try {
    startActivity(sendIntent);
} catch (ActivityNotFoundException e) {
    // Define what your app should do if no activity can handle the intent.
}
```

Sacado de: https://developer.android.com/guide/components/intents-filters

10 Implicit intent



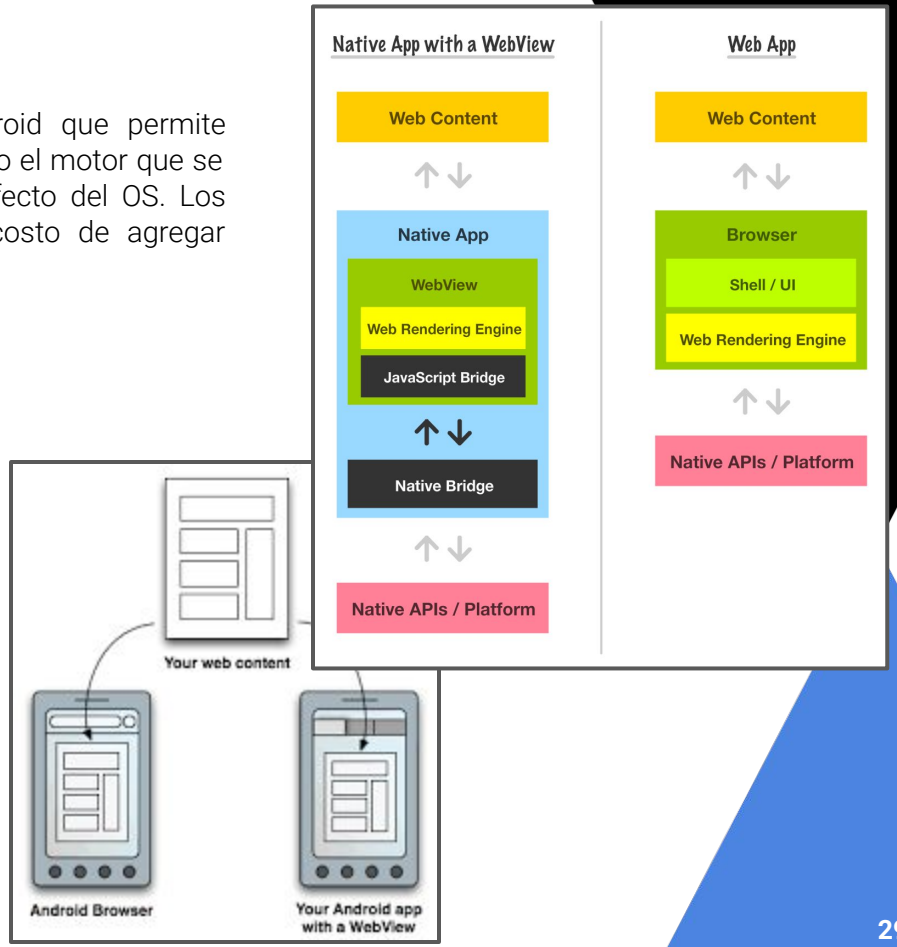
Sacado de: https://www.hebunilhanli.com/wonderland/mobile-security/android-component-security/

11 WebViews

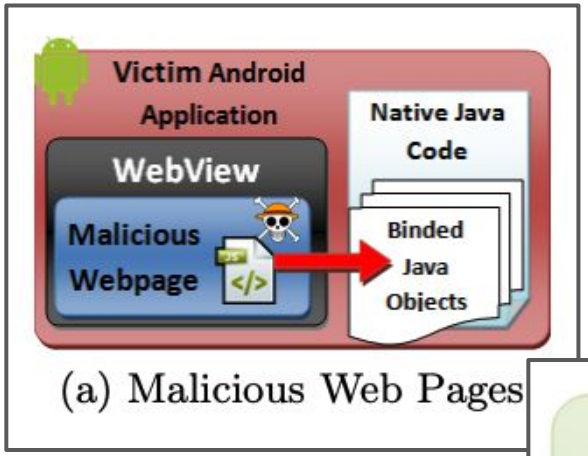
Un WebView es un componente especial del SDK de Android que permite renderizar HTML y javascript como si fuera un browser. De hecho el motor que se ejecuta es normalmente el mismo que el del browser por defecto del OS. Los WebViews agregan funcionalidades y extensibilidad con el costo de agregar complejidad de seguridad y aumentar la superficie de ataque

Aumento de superficie de ataque:

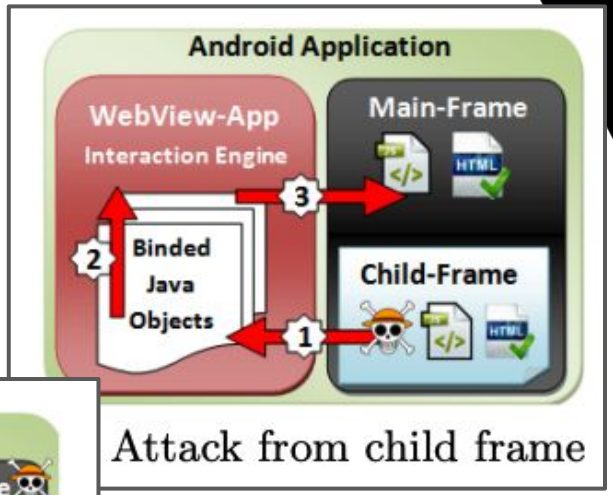
- XSS
- Open Redirect
- Acceso a archivos internos mediante JS
- Llamar otros componentes mediante intent schema
- Leak de contenido mediante misconfigurations



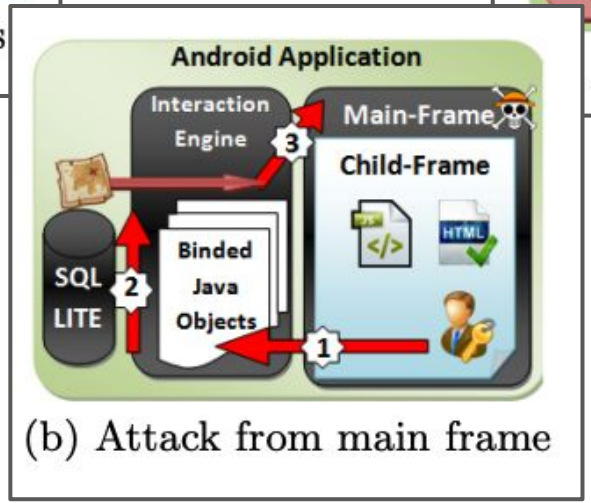
11 WebViews



(a) Malicious Web Pages



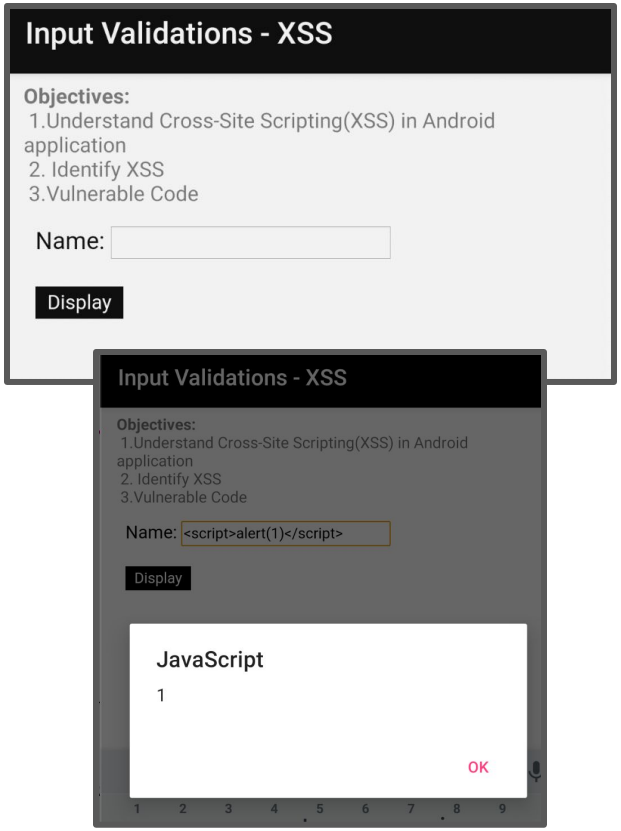
Attack from child frame



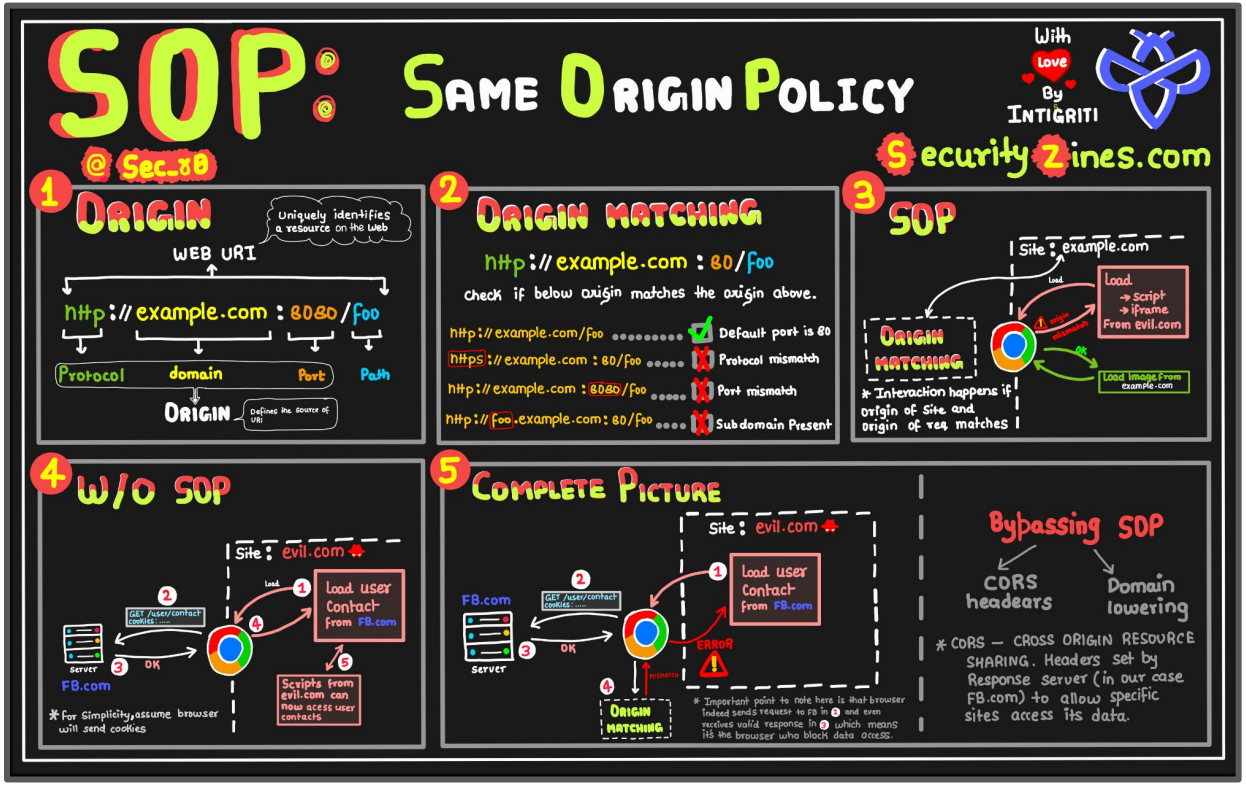
(b) Attack from main frame

11 WebViews

1) Abrir Input Validation - XSS
2) Probar con **<script>alert(1)</script>**
3) Revisar codigo con **chrome://inspect**
4) Ver valor de window.origin
5) Ver el codigo de owasp.sat.agoat.XSSActivity

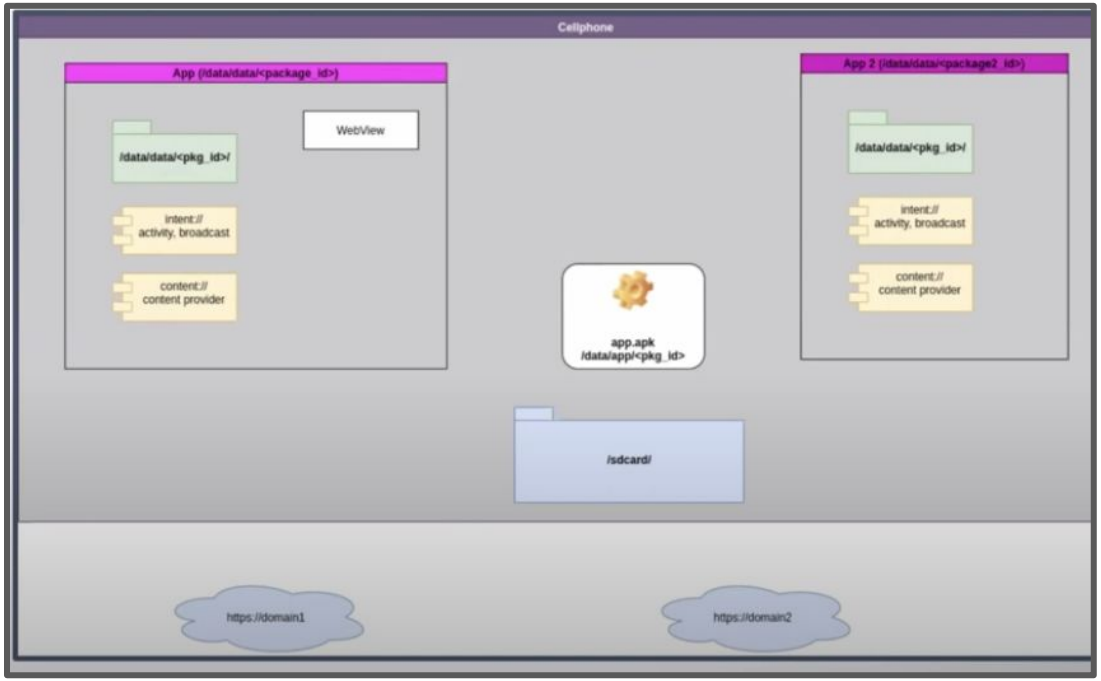


11 Same Origin Policy



Sacado de: https://securityzines.com/flyers/sop.html

11 Origin mobile

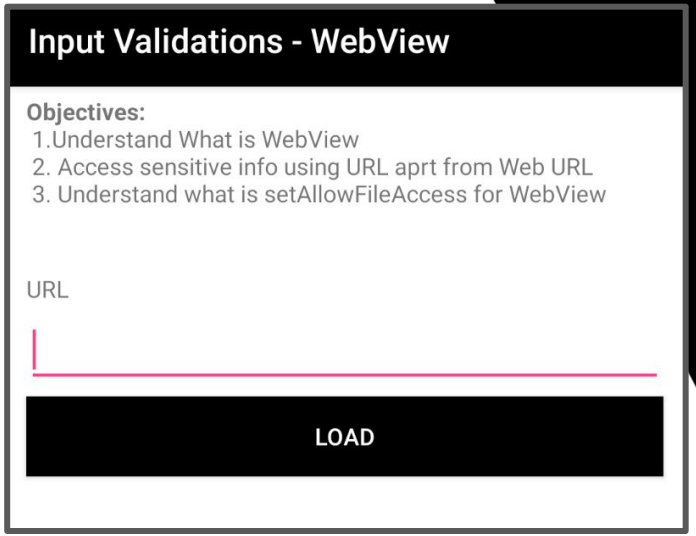


11 WebViews

1) Abrir Input Validation - XSS
2) Crear un archivo en /sdcard con el siguiente contenido:

   ```
   echo "<h1>vamos nerdaerla101v2</h1>" > example.html
   ```

3) Poner como URL **file://sdcard/example.html**
4) View value of window.origin
5) Ver el codigo de owasp.sat.agoat.InputValidationsWebViewURLActivity



12 Siguientes pasos

1) Familiarizarse con el proceso de pentest de aplicaciones mobiles (https://github.com/OWASP/owasp-mstg).
2) Probar conceptos en aplicaciones vulnerables (https://www.linkedin.com/pulse/10-vulnerable-android-applications-beginners-learn-hacking-anugrah-sr/)
3) Desarrollar algunas aplicaciones para entender el codigo
4) Leer libros o articulos sobre seguridad en aplicaciones Android
5) Tomar aplicaciones reales de programas de bounties y practicar los conceptos (no se frustren!! Los principios son la base de la diversion)
6) Leer y probar frameworks (Flutter, ReactNative, Unity, Xamarin, etc)
7) Lanzate a pileta!! Se aprende mucho mas si te dedicas a eso exclusivamente

Preguntame si tenes dudas!!

Twitter - @warlockk87
Linkedin - https://www.linkedin.com/in/cesar-mario-rodriguez/
Mail - hohenhaimmaster@gmail.com

13 Recursos interesantes

- https://owasp.org/www-project-mobile-top-10/
- https://github.com/OWASP/owasp-mstg
- https://github.com/vaib25vicky/awesome-mobile-security
- https://www.youtube.com/channel/UCitiWg5p-R6QNLPuJPSE33g
- https://blog.oversecured.com/
- https://fuzzinglabs.com/learn-android-hacking-android-security/

The key to achieving true expertise in any skill is simply a matter of practicing, albeit in the correct way, for at least 10 000 hours.

Preguntas?


¡CHAU CHICOS!
El Comandante