



Mobile Apps and How To Pentest Them.



OWASP

The Open Web Application Security Project



- \$whoami



Ing. Gustavo M. Sorondo (Puky)

CTO @ Cinta Infinita | Information Security

<http://www.cintainfinita.com>





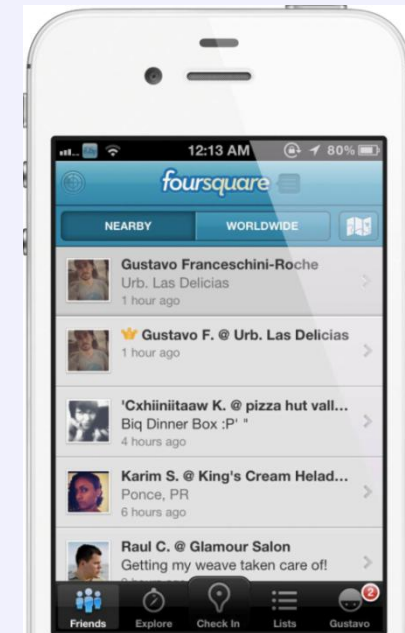
Tipos de Aplicaciones Mobile (Según su programación)



Web Based
App



Native
App



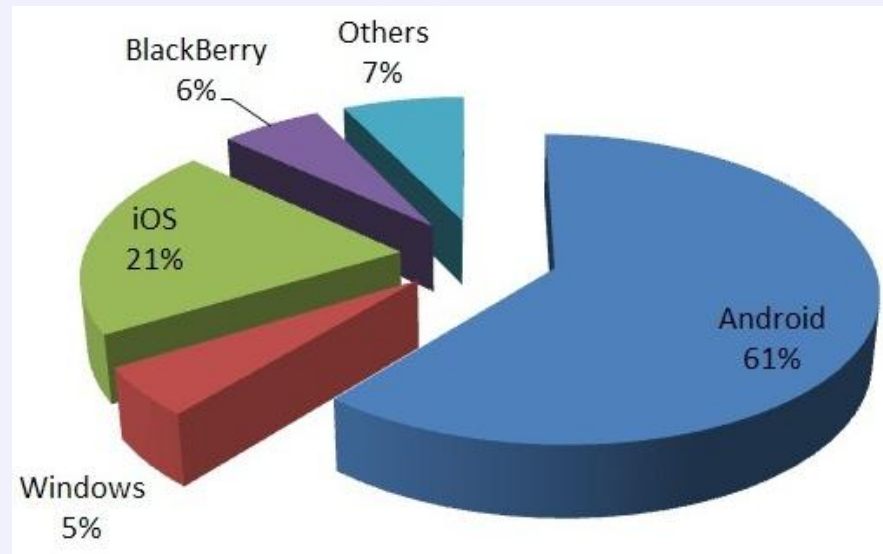
Hybrid
App

¿Comunicaciones? Básicamente HTTP(S)



Plataformas Actuales

(Market Share por Sistema Operativo)



Lenguaje de Programación Utilizado



Java



Objective-C



Java



.NET C#
Silverlight



OWASP

Mobile Security Project

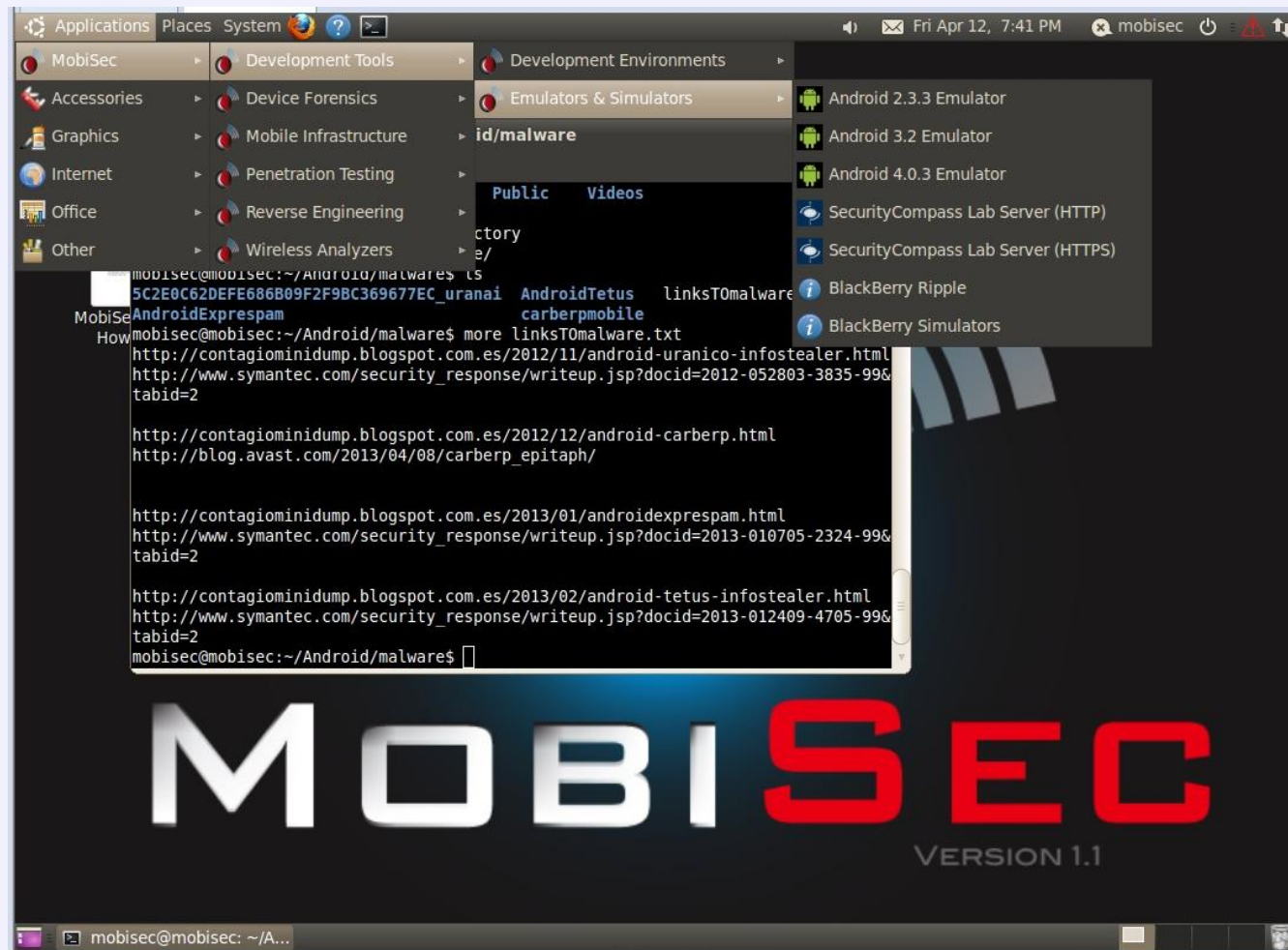




- Presentado en AppSec 2011
- Mobile Threat Model.
- Mobile Testing Tools.
 - GoatDroid / iGoat. / DVIA
 - MobiSec.
 - Androick / NowSecure App Testing / Seraphimdroid
- Mobile Top 10 Risks. (Version final 2014)
- Mobile Top 10 Controls & Design Principles.
- Testing Guide.
- Development Guide. (muy verde todavía...)



MobiSec v1.3 (Nov. 2014)



MOBISec

VERSION 1.1

<http://sourceforge.net/projects/mobilsec/>



OWASP Mobile Top 10 Risks

M1 – Weak Server
Side Controls

M2 – Insecure
Data Storage

M3 - Insufficient
Transport Layer
Protection

M4 - Unintended
Data Leakage

M5 - Poor
Authorization and
Authentication

M6 - Broken
Cryptography

M7 - Client Side
Injection

M8 - Security
Decisions Via
Untrusted Inputs

M9 - Improper
Session Handling

M10 - Lack of
Binary Protections

Call to Action for 2015

M1: Weak Server Side Controls



- Nada nuevo por aquí...
- DO NOT TRUST THE CLIENT!





- Almacenamiento de información sin protección.
- Local / Nube.
- Cache / Malas Prácticas.



M3: Insufficient Transport Layer Protection



- Aplicaciones que se comunican sin cifrado.
- Uso de cifradores débiles..
- No responder correctamente ante errores de certificados.





- Leaking de información por usar librerías de terceros que no se controlan completamente.
 - Logs
 - Caches
 - Archivos temporales
 - Etc.



- Un poco mas de lo mismo, pero con algunas variantes...
- Uso de datos del equipo para autenticación:
 - IMEI / IMSI
 - UUID
- Problema? Persisten con el equipo.



OWASP

The Open Web Application Security Project

- Ya es conocido pero vale la pena refrescar:
 - Encoding != Encryption
 - Obfuscation != Encryption
 - Serialization != Encryption
- Utilización de algoritmos fuertes pero mal implementados.
 - Almacenamiento de claves junto con los datos.
- Programar su propia criptografía = generalmente malo.



- Aplicaciones Web / Hybrid.
- ¿XSS? ¿SQL Injection?
- Nuevos recursos. SMS, llamadas, servicios pagos.

“Mandá OWASP al 2020”



- Estas decisiones permiten saltar restricciones o modelos de seguridad.
 - iOS – Abuso de URL Schemes
 - Android – Abuso de Intents
- Vectores de ataque
 - Aplicaciones maliciosas.
 - Client-side Injections.

M8: Security Decisions Via Untrusted Inputs



- Ej: Ravi Borgaonkar - ekoparty 2012
- Explotaba una vulnerabilidad en el intent "tel:" que permitía marcar un número desde otra aplicación y borrar cualquier teléfono Samsung.



<iframe src="tel:*2767*3855#" width="320" height="240"></iframe>

M9: Improper Session Handling



- Esto ya lo conocemos...
 - Cookies, Oauth, Single-Sign-On, etc...
 - Sesiones mas duraderas que en Web.



Gimmi your cookies!



- Reversing del binario
- Análisis del código fuente
 - Lógica de la App.
 - Búsqueda de vulnerabilidades más simple.
 - Leak de información en el código.
- Ya veremos más sobre esto...



- Por OWASP y la “European Network and Information Security Agency” (ENISA).
- Controles y buenas prácticas para evitar cada riesgo del Top 10.

OWASP Mobile Security Project

[Project Overview](#)[Top Ten Mobile Risks](#)[Mobile Tools](#)[Mobile Security Testing](#)[Mobile Cheat Sheet Series](#)[Secure Mobile Development](#)[Top Ten Mobile Controls](#)[OWASP Mobile Threat Model Project](#)

OWASP/ENISA Collaboration

OWASP and the European Network and Information Security Agency (ENISA) collaborated to build a joint set of controls. ENISA has published the results of the collaborative effort as the "Smartphone Secure Development Guideline":
<http://www.enisa.europa.eu/activities/application-security/smartphone-security-1/smartphone-secure-development-guidelines>

Contributors

This document has been jointly produced with ENISA as well as the following individuals:

Pentesting Mobile Apps

(acá viene lo divertido...)





- Armado del laboratorio.
- Análisis y comprensión de la lógica de la app.
- Análisis de seguridad de los servidores.
- Análisis de los archivos creados por la app.
- Análisis de los datos en memoria.
- Ingeniería Inversa de código fuente / búsqueda de información sensible y vulns.
- Búsqueda de vulns a través del análisis y manipulación de la mensajería.

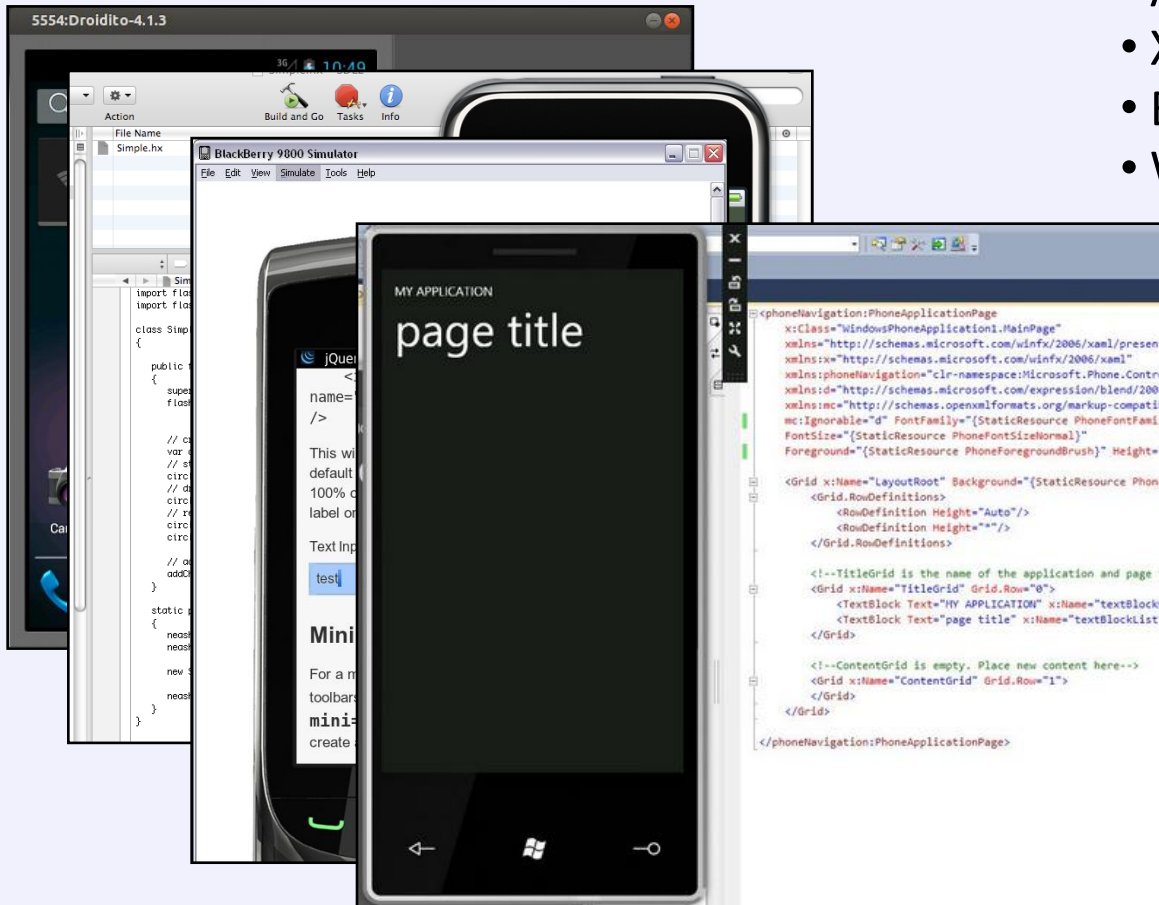
Armado del Laboratorio





Dispositivos Reales vs. Emuladores

- Android SDK Emulators
- XCode iOS Simulator
- Blackberry Simulators
- Windows Phone Emulator





OWASP

The Open Web Application Security Project

Análisis de Archivos





- ¿Qué se busca?
- ¿Para que?
- Root/ Jailbreak necesario en dispositivos reales.



`/data/data/<nombre_pkg>/`

Shared Preferences

Base de datos SQLite



`/private/var/mobile/
Applications/<id_app>`

PropertyList (.plist)

Base de datos SQLite



- Ejemplo:

A screenshot of a mobile application's login screen. At the top, there's a status bar showing "9:42 AM". Below it is a header with the word "Login". The form has two input fields: "Username" and "Password". Below the password field is a checkbox labeled "Remember Me" which is checked, highlighted by a red rectangular box. At the bottom of the form are two blue buttons: "Login" and "Register". The OWASP logo is at the very bottom of the screen.

```
public void saveCredentials (String userName, String password){  
    SharedPreferences credentials = this.getSharedPreferences("credentials", MODE_WORLD_READABLE);  
    SharedPreferences.Editor editor = credentials.edit();  
    editor.putString("username", userName);  
    editor.putString("password", password);  
    editor.putBoolean("remember", true);  
    editor.commit();  
}
```



- Veamos algunas tools...
- adb: Android Debugging Bridge
 - android-sdk/platform-tools/adb
 - Dispositivo Físico = USB Debugging ON
 - <http://developer.android.com/tools/help/adb.html>
 - Comandos: devices, shell, pull, push, install, ...
- SQLiteMan
 - apt-get install sqliteman



OWASP

The Open Web Application Security Project

Reversing de Código Fuente





- Binario compilado -> Código legible
- Bastante simple en Android
 - Podemos llegar al código Java completo.
 - APK = ZIP
- En iOS solo podríamos llegar a un código Assembler
 - Windows Phone y Blackberry mas simple.



- Ejemplo: Mobile Banking

A screenshot of an IDE. On the left is a project tree with folders like 'org.json' and 'toobjectivec'. The 'toobjectivec' folder is expanded, showing files like 'ConvertPropertiesFile', 'ToObjectiveC', 'AttachNote', 'DeleteAllMovementResources', 'MovementResourceList', and 'VerificarRecurso'. The 'VerificarRecurso' file is selected. On the right, the code for 'VerificarRecurso' is shown. It includes a method 'testLoginArray()' which returns a list of 'VerificarRecurso.TestLoginUser' objects. A red circle highlights the 'TestLoginUser' objects in the list, specifically the ones with IDs '193227', '133216', '400622', and '410827'.

```
private List<VerificarRecurso.TestLoginUser> testLoginArray()
{
    ArrayList localArrayList = new ArrayList();
    localArrayList.add(new VerificarRecurso.TestLoginUser("193227", "123456", null));
    localArrayList.add(new VerificarRecurso.TestLoginUser("133216", "123456", null));
    localArrayList.add(new VerificarRecurso.TestLoginUser("400622", "123456", null));
    localArrayList.add(new VerificarRecurso.TestLoginUser("410827", "123456", null));
    return localArrayList;
}

protected void onAccountMovements()
```

A screenshot of an IDE showing the source code of 'ToObjectiveC.class'. The code is in Java and includes package declarations, imports, and class definitions. The 'ToObjectiveC' class has a private field 'path' and a private method 'convert()' that throws a 'Throwable'.

```
ToObjectiveC.class
package toobjectivec;

import java.io.File;

public class ToObjectiveC
{
    private java.lang.String path = "/Users/franco/Dropbox/devel/Banca Movil [redacted]/Banca Movil [redacted]/src/comm/beans";

    private void convert()
        throws Throwable
    {
        Class[] arrayOfClass = new Class[0];
    }
}
```



- Ejemplo: Padrón Electoral 2013

http://wsp.mininterior.gov.ar/ws_escuela.php?param=v%23v%23gWHVDcQRVRtNmMWRjY6FjT

```
public static String codificar(String s)
{
    String s1 = (new StringBuffer(
        (new StringBuilder(Base64.encodeBytes((new StringBuffer(
            new StringBuilder(Base64.encodeBytes(s.getBytes()))).toString()
            .replace("a", "#t").replace("e", "#x").replace("i", "#f").
            replace("o", "#l").replace("u", "#7").replace("=", "#g"))).
            reverse().toString().getBytes()))).toString().replace("a", "#j").
            replace("e", "#p").replace("i", "#w").replace("o", "#8").replace("u", "#0")
            .replace("=", "#v"))).reverse().toString();

    String s2;
    try
    {
        s2 = URLEncoder.encode(s1, "utf-8");
    }
    catch(UnsupportedEncodingException unsupportedencodingexception)
    {
        return s1;
    }
    return s2;
}
```



¿Cómo hacemos? Vamos a las tools...

- APKDownloader
 - Extensión de Chrome.
 - <http://apps.evozi.com/apk-downloader/>
- APKTool
 - Permite decompilar y compilar código en formato SMALI (un assembler de DALVIK).
 - Útil para realizar cambios básicos (ej. Modificar valores)
 - <https://code.google.com/p/android-apktool/>



Mas tools...

- dex2jar
 - Permite pasar de formato dex (Dalvik Executable) a JAR.
 - <http://code.google.com/p/dex2jar/>
- JDGUI
 - Permite decompilar un JAR a código JAVA.

¿Las probamos?



¿Protección? ProGuard

A screenshot of a Java IDE window titled 'com.whatsapp-41547_dex2jar.jar'. The left sidebar shows a project tree with a package 'a' containing a class 'a' (with methods 'a: long', 'a(Reader)', and 'readLine(): String') and a series of classes 'ab' through 'jb'. The main editor displays the decompiled source code for 'eb.class'. The code includes an import for 'com.whatsapp.App', a 'public class eb' that implements 'db' and 'v', and a 'static' block with various array and variable declarations and initializations.

```
import com.whatsapp.App;

public class eb
    implements db, v
{
    private static final x[] a;
    private static final String[] z;
    private x[] b = a;
    private char[] c;
    private int d;
    private int e;
    private Object[] f;
    private int[] g;
    private int[] h;
    private int[] i;
    private int j;

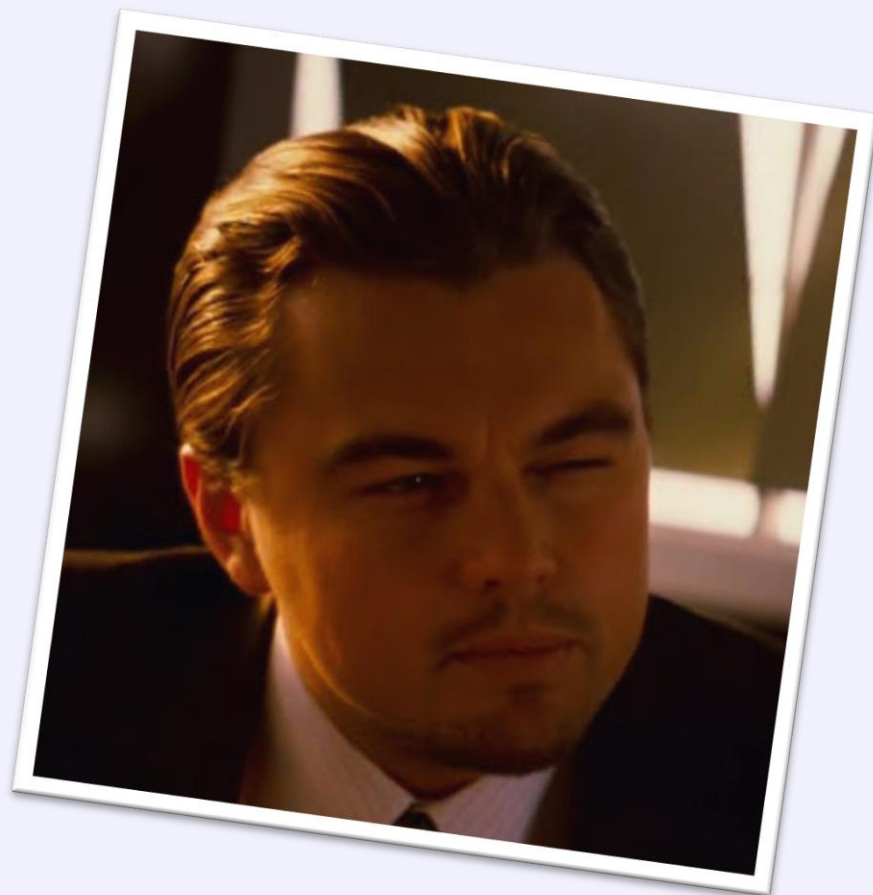
    static
    {
        String[] arrayOfString = new String[13];
        char[] arrayOfChar1 = "t${}\f1t!\t\t\t".toCharArray();
        int k = arrayOfChar1.length;
        int m = 0;
        char[] arrayOfChar2;
        int i3;
        char[] arrayOfChar3;
        int i7;
        char[] arrayOfChar4;
        int i11;
        char[] arrayOfChar5;
    }
}
```



OWASP

The Open Web Application Security Project

Análisis de Lógica de la Aplicación





- Activity: Clase que representa una acción que la App hace (generalmente una pantalla).
- Intent: Mensaje que una App puede enviar para ejecutar “activities” (propias o de otra App).
- Las activities disponibles para ejecutarse por Intents estan en el `AndroidManifest.xml`



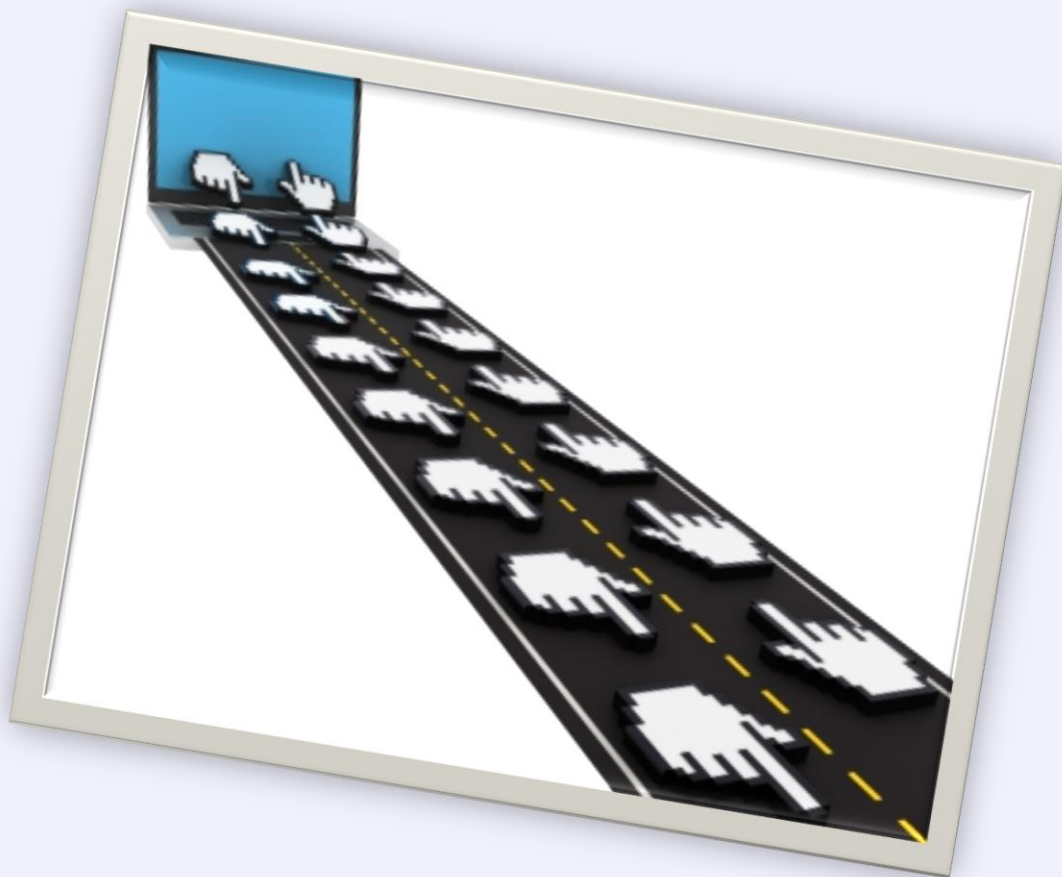
- Tools...
- am – activity manager
 - Incluido con ADB.
 - <http://developer.android.com/tools/help/adb.html#am>

Ejemplo:

```
$ adb shell
```

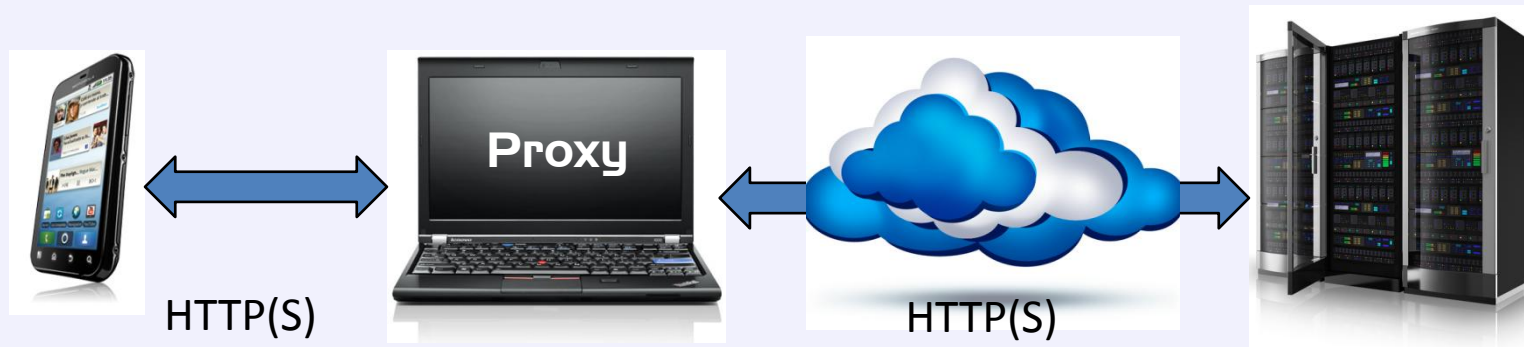
```
# am start -a android.intent.action.VIEW -d http://www.owasp.org/
```

Análisis de Mensajería





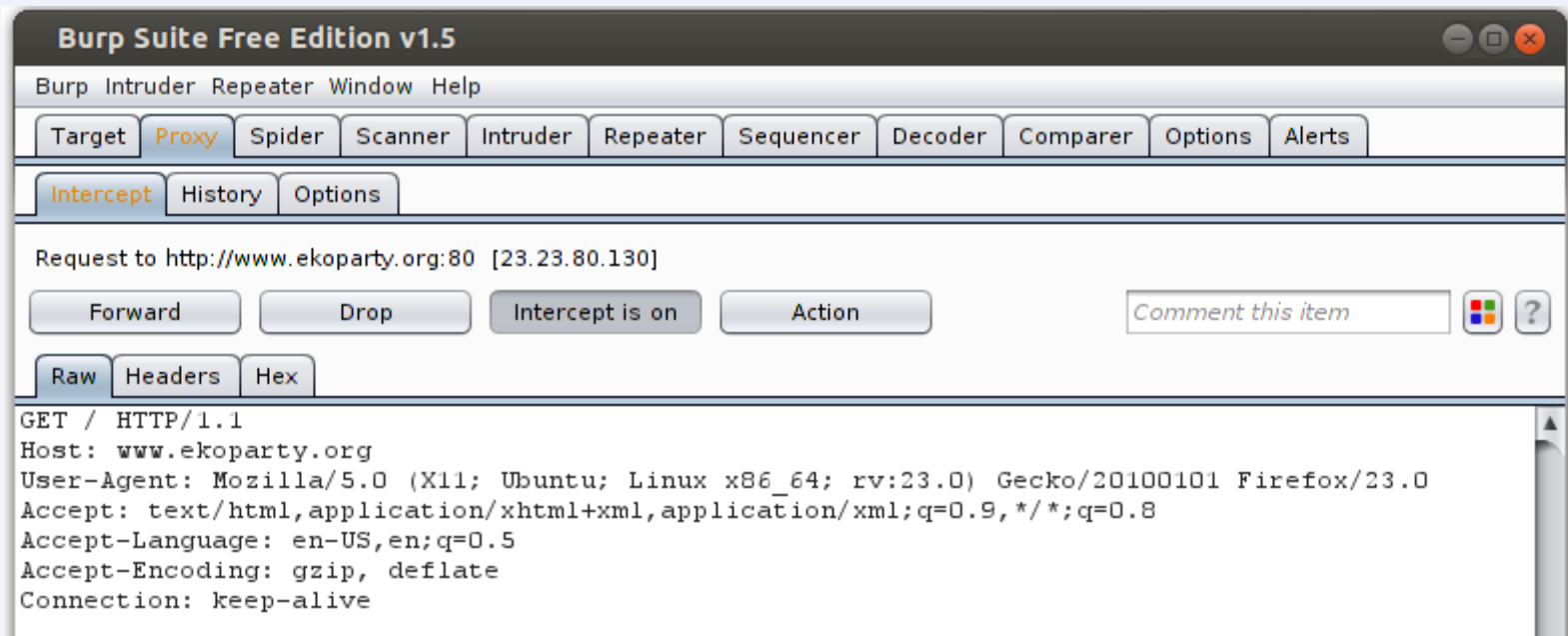
- Man-in-the-middle:



- Configuración de proxy:
 - Soportada por todos los emuladores.
 - Soportada en algunos terminales.
 - DNS Spoof / Packet Forwarding / Hosts.

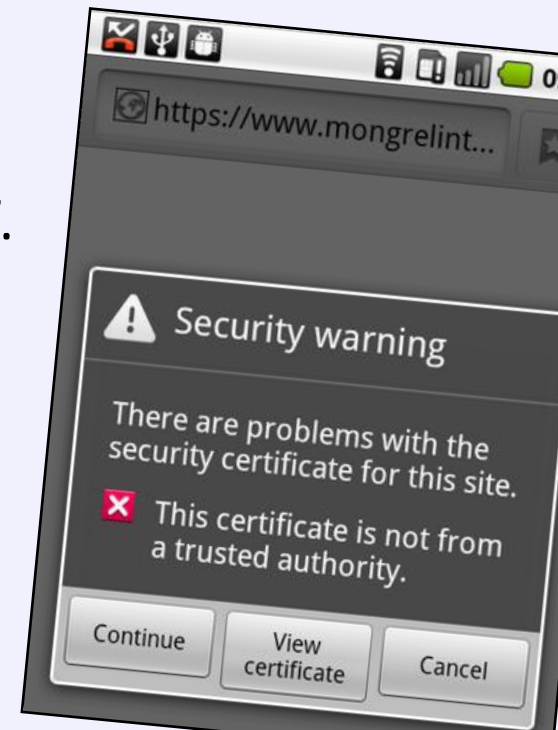


- HTTP(S) Proxy: Burp Suite
 - Free
 - <http://portswigger.net/burp/>
 - `$/emulator @Droidito-4.1.3 -http-proxy 127.0.0.1:8080 -debug-proxy`





- ¿Qué pasa con SSL?
 - El tratamiento de certificados depende de la aplicación.
- Certificate Pinning
 - Moxie Marlinspike:
 - “Your app shouldn't suffer SSL's problems”.
 - CA System vs. Keystores Propias.
 - Pinning del certificado o de la CA.
 - ¿Esto es Bueno o Malo?





¿Podemos saltar estas restricciones?

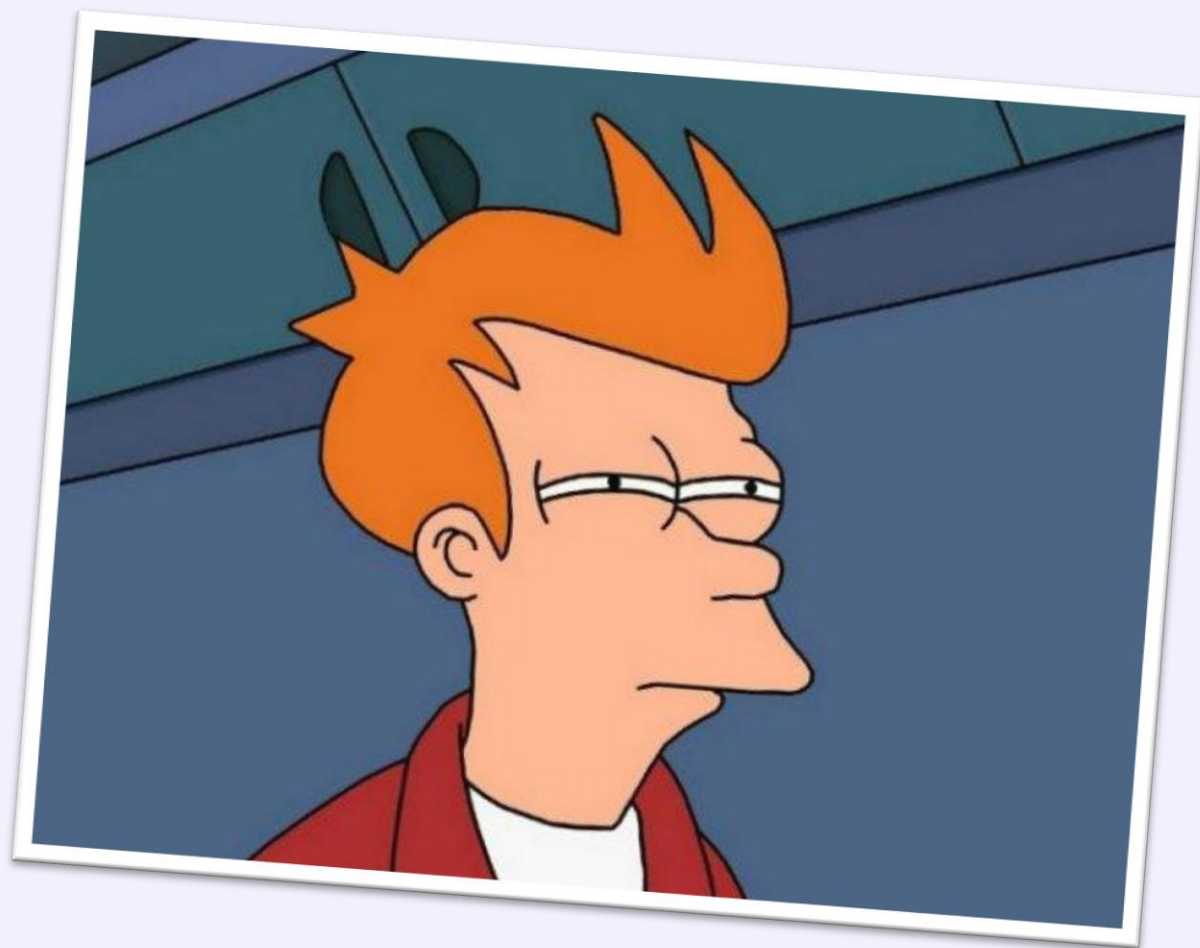
- Modo manual
 - Download/Decompile/Modify/Compile/Sign/Install
 - Tools:
 - keytool y jarsigner (incluidas en JDK)
 - Todas las anteriores...
- Modo Automático (todavía verde...)
 - Function hooking
 - Android SSL Bypass by ISEC Partners
 - Black Hat USA 2012
 - También iOS SSL KillSwitch



OWASP

The Open Web Application Security Project

¿Preguntas?





OWASP

The Open Web Application Security Project

Muchas gracias!!



gs@cintainfinita.com.ar



[@iampuky](https://twitter.com/iampuky)



[/in/gustavosorondo](https://in.linkedin.com/in/gustavosorondo)

