Auditoría de Seguridad
a Aplicaciones iOS y Android

Autor: Héctor Pauta Martillo

Tutor: Joan Caparrós

Profesor: Víctor Font

Máster Interuniversitario de Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)

TFM: Auditoría de Seguridad a aplicaciones iOS y Android

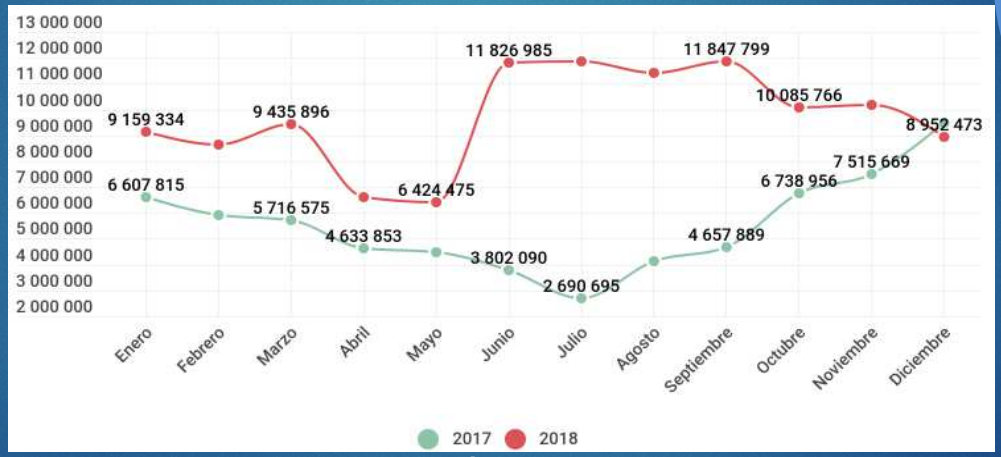
Fecha: 22 de mayo del 2020

Índice

- Introducción
- Aplicaciones móviles y sus amenazas
- Auditoría de seguridad de apps móviles
- Preparación del entorno para auditoría
- Análisis estático y dinámico de una app
- Informe de resultados
- Conclusiones

Introducción: Contexto

Actualmente, existen alrededor de 5.200 millones de usuarios de dispositivos móviles. Ericsson anunció años atrás que habrá alrededor de 6.000 millones de usuarios en 2020, con lo que se superarán a los usuarios de teléfonos fijos, por lo que las amenazas irán en aumento.



Ataques a dispositivos móviles por Malware 2017 y 2018

Introducción: Objetivos

- Proponer una metodología a seguir para la realización de auditorías de seguridad en apps móvil.
- Estudiar los distintos riesgos que pueden afectar a una aplicación en iOS y Android.
- Conocer las distintas herramientas, para la ejecución de auditorías de seguridad.
- Realizar un ejercicio práctico, que nos permita identificar las vulnerabilidades y amenazas.
- Presentación de conclusiones.

Introducción: Planificación

Para la consecución de los objetivos se planteó la entrega de los siguientes hitos:

- PEC-1: Introducción
- PEC-2: Investigación
- PEC-3: Ejercicio práctico
- PEC-4: Memoria del TFM
- Presentación Video

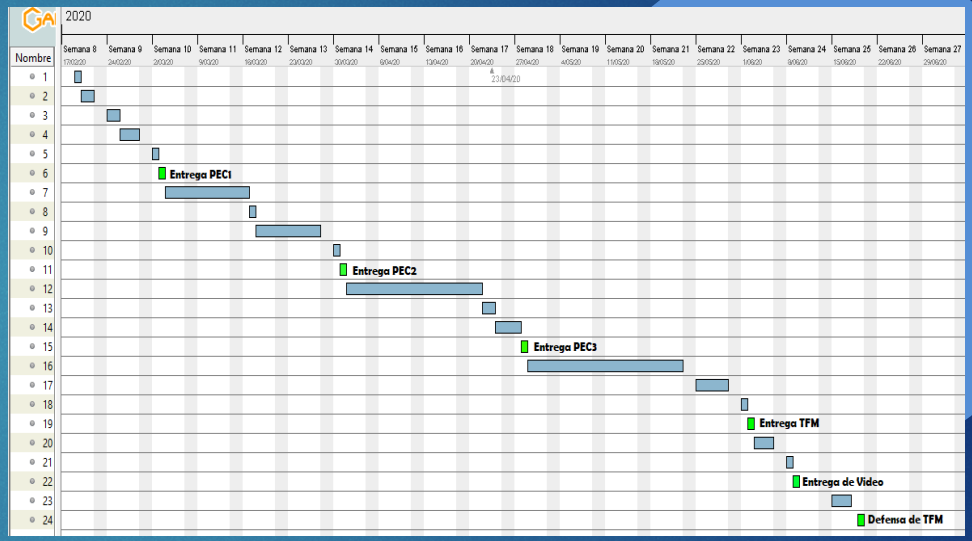


Diagrama de Gantt de TFM

Aplicaciones Móviles y sus amenazas

Son programas diseñados para ser ejecutados en teléfonos, tablets y otros dispositivos móviles, que permiten al usuario realizar actividades profesionales, acceder a servicios y mantenerse informado.

Pueden ser:

- Nativas
- Web apps
- Híbridas

Tipos de Aplicaciones Móviles

Aplicaciones Móviles y sus amenazas

Entre las **principales amenazas**:

- Acceso a los datos que se almacenan en local
- Gestión incorrecta de errores
- Gestión deficiente de sesiones
- Vulnerabilidades en el código fuente

Auditoría de Seguridad

Una Auditoría de Seguridad es un proceso riguroso, basado en una metodología, por el cual se identifican las vulnerabilidades en una aplicación móvil.

De acuerdo al enfoque puede ser:

- Caja negra
- Caja blanca
- Caja gris

Auditoría de Seguridad: Metodología

OWASP Mobile Security Project, se trata de un proyecto centralizado destinado a proporcionar a desarrolladores y equipos de seguridad los recursos que necesitan para construir y mantener aplicaciones móviles seguras.

El objetivo es clasificar los riesgos de seguridad móvil y proporcionar controles para reducir su impacto o posibilidad de explotación.

Auditoría de Seguridad: Metodología

OWASP Mobile Security Project

- Mobile Top 10 Risks
- Mobile Top 10 Controls
- App Security Testing Cheat S
- Security Mobile Development
- Security Testing Guide
- Developer Cheat Sheet

Auditoría de Seguridad: Metodología

TOP 10
Owasp Mobile
Top10 Risks 2016

- M1 - Uso incorrecto de la plataforma
- M2 - Almacenamiento de datos inseguro
- M3 - Comunicación Insegura
- M4 - Autenticación Insegura
- M5 - Criptografía Insuficiente
- M6 - Autorización Insegura
- M7 - Calidad del código del cliente
- M8 - Manipulación del código
- M9 - Ingeniería Inversa
- M10 - Funcionalidad extraña

Auditoría de Seguridad: Metodología

TOP 10
Owasp Mobile
Top10 Controls 2016

- C1 - Protección para datos sensibles
- C2 - Protección autenticación por claves
- C3 - Protección para datos en tránsito
- C4 - Autenticación segura
- C5 - Plataforma servidor segura
- C6 - Integración segura con terceros
- C7 - Consentimiento para la recogida info
- C8 - Protección para los servicios de pago
- C9 - Distribución segura de App y Updates
- C10 - Impedir el uso de intérpretes

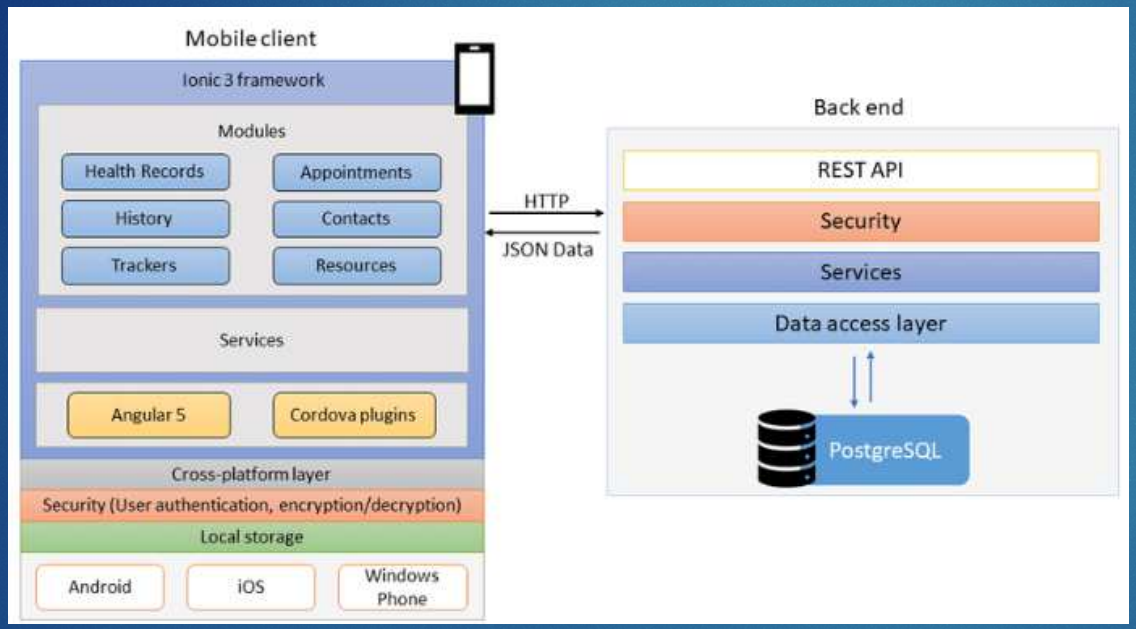
Auditoría de Seguridad: Metodología

Centrándonos en el proyecto OWASP Mobile Security podemos dividir la auditoría en tres fases:

- Reconocimiento
- Análisis estático
- Análisis dinámico

Auditoría de Seguridad: Reconocimiento

Se invierten recursos y tiempo en un exhaustivo conocimiento de la app a auditar.



Estructura de una Aplicación Móvil

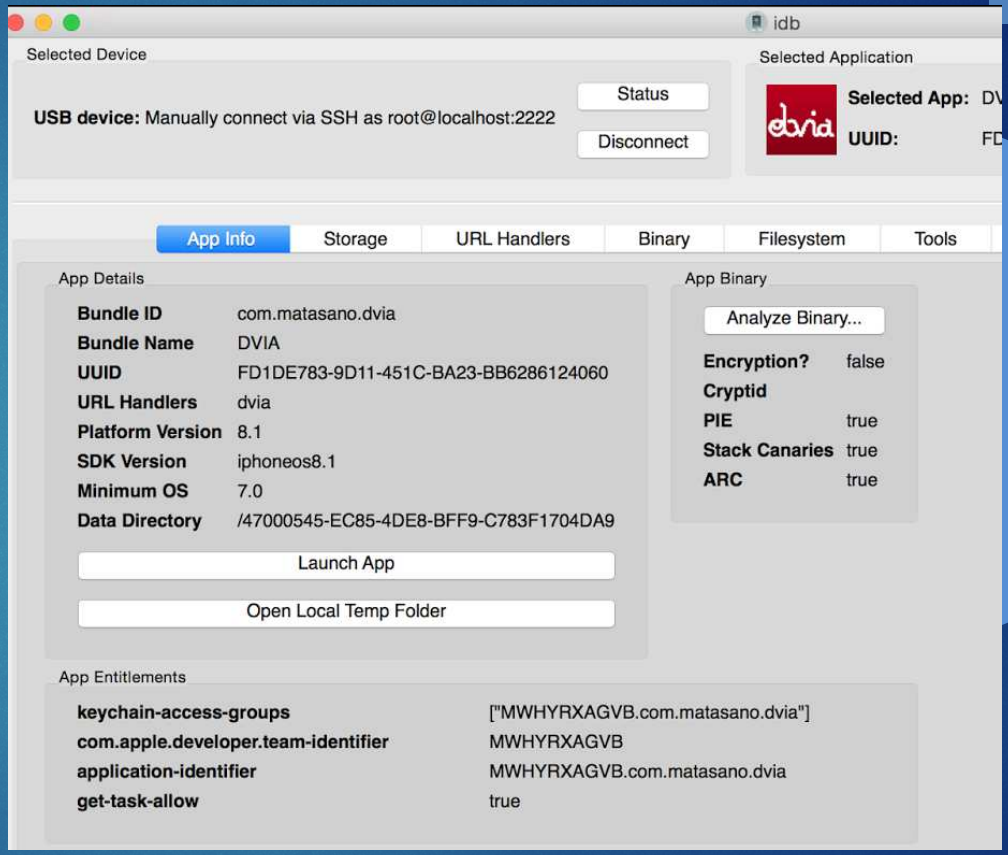
Auditoría de Seguridad: Reconocimiento

Android:

- Apk Analyzer
- Drozer

iOS:

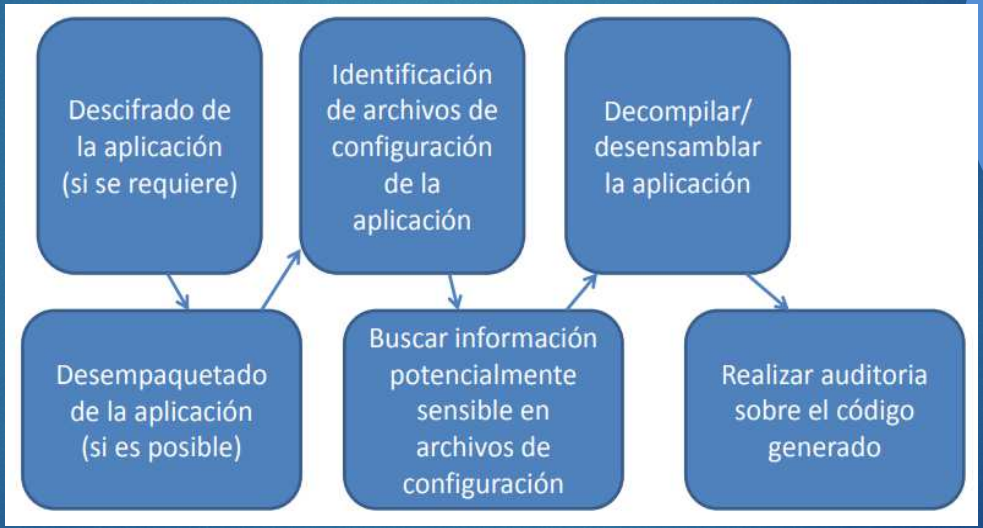
- iNalyzer
- IDB



Aplicativo IDB

Auditoría de Seguridad: Análisis Estático

También denominado SAST. Consiste en la revisión del código fuente analizando los distintos componentes de ésta, ya sea de forma manual o automática.



Tareas para el Análisis Estático

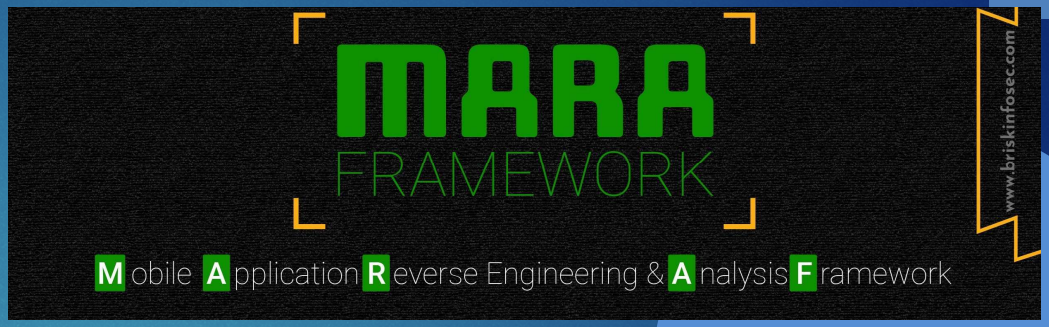
Auditoría de Seguridad: Análisis Estático

Android:

- Mara
- Mobsf
- Radare2

iOS:

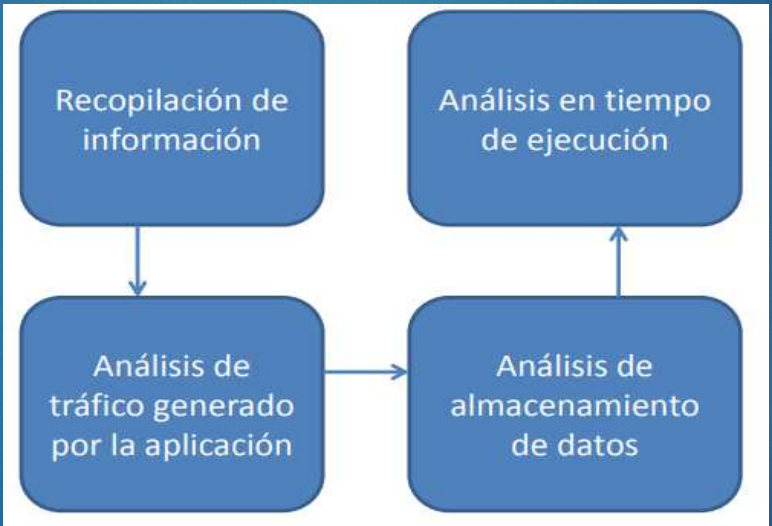
- Hopper/iRET
- Mobsf
- Radare2



Herramientas para el Análisis Estático

Auditoría de Seguridad: Análisis Dinámico

También conocido como DAST consiste en el análisis de seguridad durante la propia ejecución de la App, con ello es posible descubrir vulnerabilidades en tiempo real.



Tareas para el Análisis Dinámico

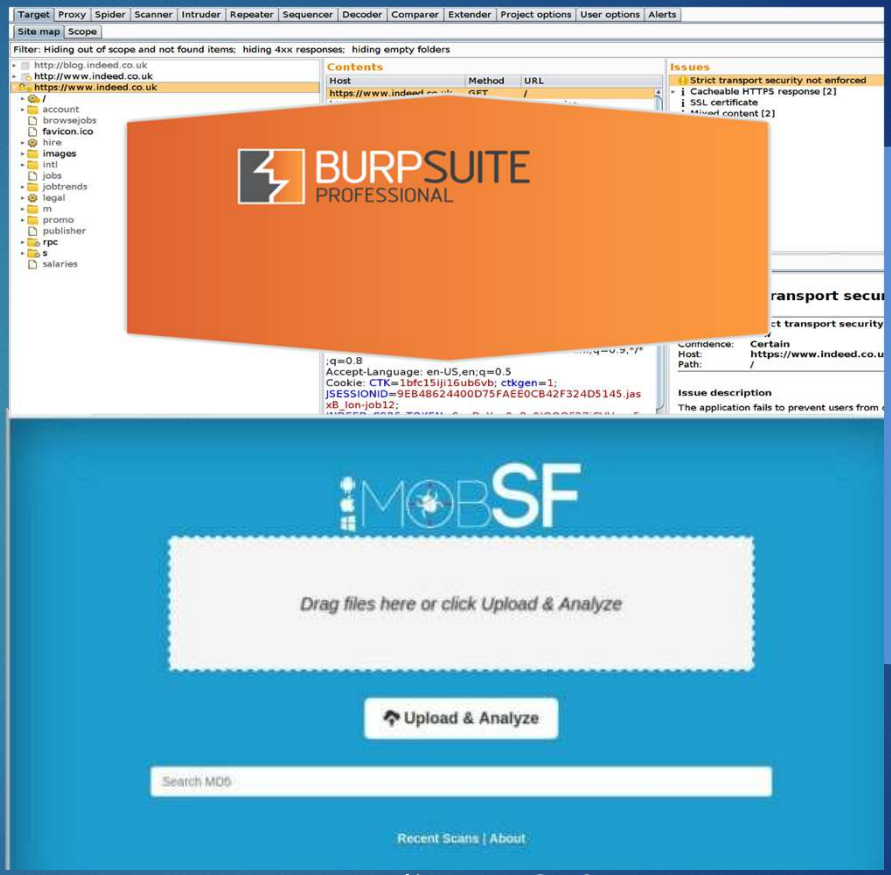
Auditoría de Seguridad: Análisis Dinámico

Android:

- Drozer
- Burpsuite
- Xposed

iOS:

- Cycript
- Burpsuite
- Frida

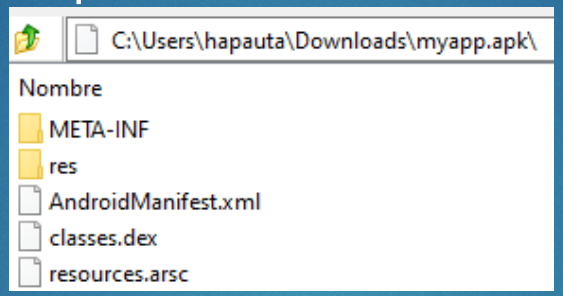


Burpsuite y MOBSF

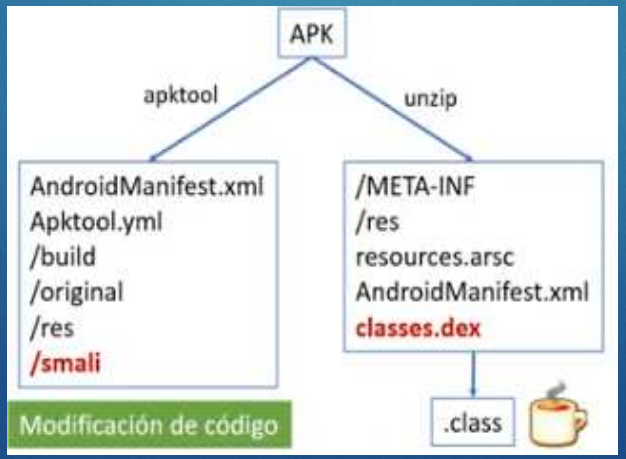
Demo Android

Demo Android: Análisis Estático

El .apk es posible descomprimirlo y consultar los diferentes recursos que se muestran.



O decompilarlo utilizando Apktool o Apk Easy Tool.



Demo Android: Análisis Estático

Al decompilar el binario con Apk Easy Tool obtenemos el código en Smali. Aquí podríamos realizar un cambio a nivel de librerías y recompilar el .apk.

Smali similar al lenguaje ensamblador, pero en este caso se ven los nombres de las clases y métodos de Java.

```
133     .line 131
134     .local v0, "hasSpecialChar":Z
135     iget-object v5, p0, Lcom/mwr/example/sieve/PINActivity;->currentPIN:Ljava/lang/String;
136
137     invoke-virtual {v3, v5}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z
138
139     move-result v5
140
141     if-eqz v5, :cond_3
142
```

Lenguaje Smali

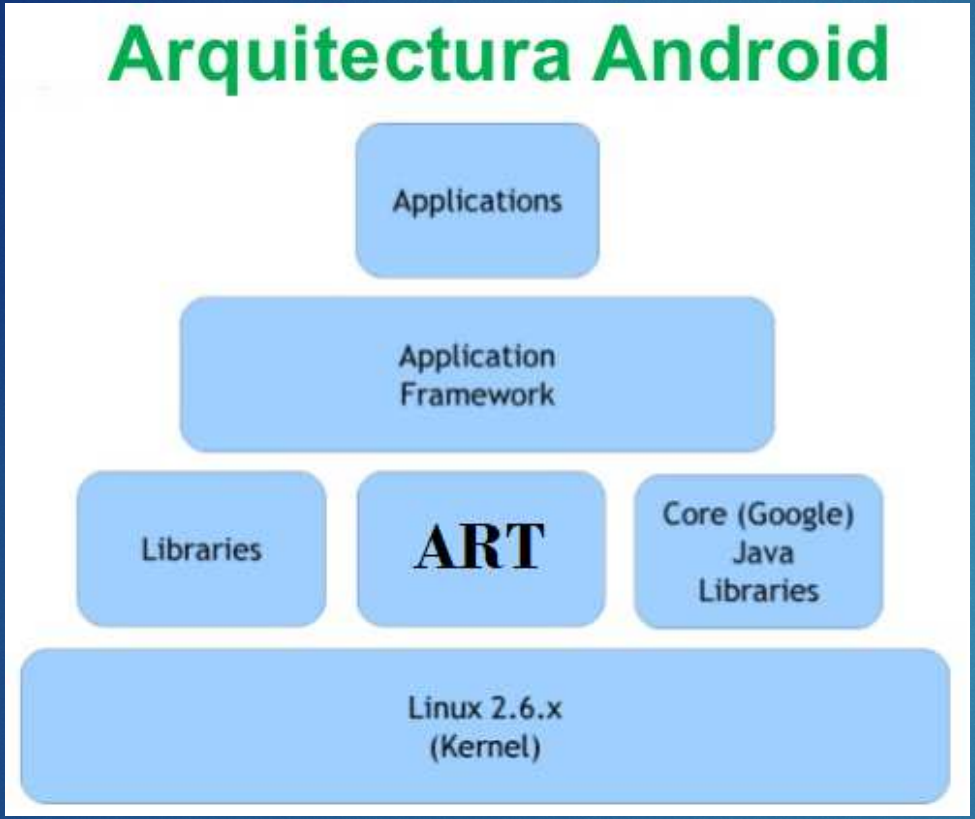
Demo Android: Análisis Estático

Las aplicaciones deberán declarar la necesidad de acceder a recursos concretos (permisos) en manifest.xml

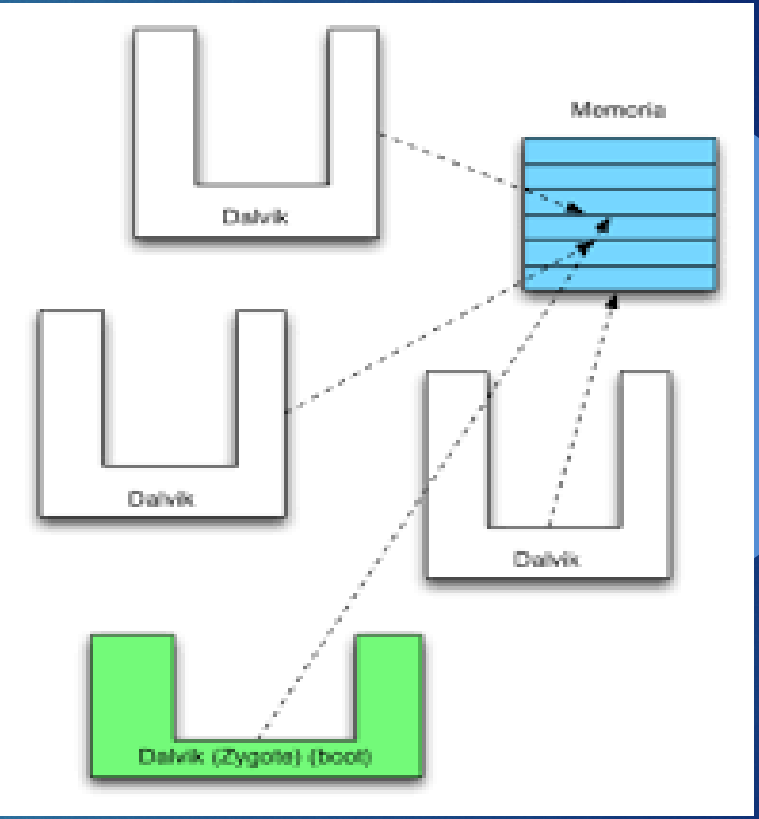
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.novaapps.findevents" android:versionCode="1"
    android:versionName="1.0"
    android:installLocation="preferExternal">
    <uses-sdk android:minSdkVersion="4" />
    <supports-screens
        android:largeScreens="true"
        android:normalScreens="true"
        android:smallScreens="true"
        android:resizeable="true"
        android:anyDensity="true" />

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".FindEventsGADroidActivity"
            android:label="@string/app_name" android:configChanges="orientation|keyboardHidden">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name="com.phonegap.DroidGap" android:label="@string/app_name"
            android:configChanges="orientation|keyboardHidden">
            <intent-filter>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

AndroidManifest.xml

Demo Android: Análisis Dinámico



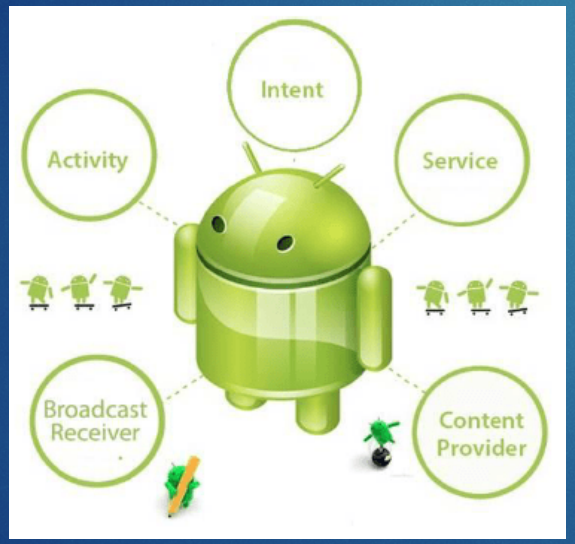
Pila de Software

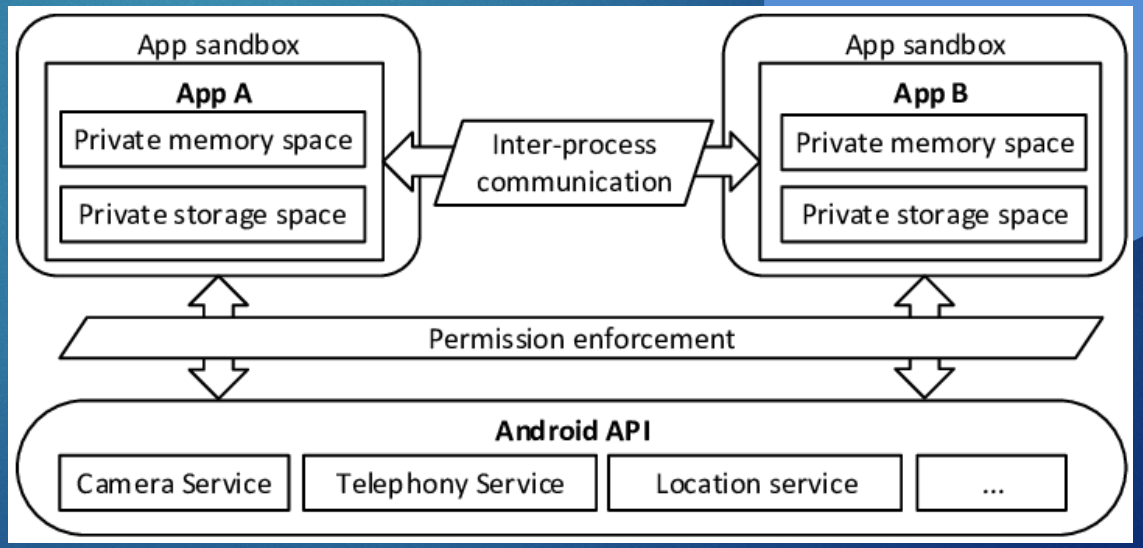


VM por cada Aplicación

Demo Android: Análisis Dinámico

IPC (Inteprocess Communication) cada aplicación de Android está protegida, por lo que cada proceso es independiente el uno del otro. aquí aparece IPC para la inter comunicación entre aplicaciones y componentes.





IPC: Comunicaciones en Android

Demo Android

VIDEO

Demo iOS: Análisis Estático

DRM (Gestión de Derechos Digitales)

- Las aplicaciones que nos descargamos o instalamos están cifradas. Mientras que las aplicaciones preinstaladas no están cifradas.
- Con las aplicaciones cifradas no existe una decompilación real, lo que se hace es un dumpeo en memoria de las cabeceras y métodos.

Demo iOS: Análisis Estático

Property List Files: Son archivos que almacenan objetos serializados. Éstos archivos usan la extensión .plist, por lo que son conocidos como archivos plist. Es posible abrirlos con Plist Editor Pro.

```
Black1pad:/Applications/DamnVulnerableIOSApp.app root#
Black1pad:/Applications/DamnVulnerableIOSApp.app root# cd Documents/
Black1pad:/Applications/DamnVulnerableIOSApp.app/Documents root# ls -l
total 60
-rw-r--r-- 1 root admin 24576 Nov 29 15:52 CoreData.sqlite
-rw-r--r-- 1 root admin 32768 Nov 29 15:52 CoreData.sqlite-shm
-rw-r--r-- 1 root admin     0 Nov 29 15:52 CoreData.sqlite-wal
-rw-r--r-- 1 root admin   286 Nov 29 15:53 userInfo.plist
Black1pad:/Applications/DamnVulnerableIOSApp.app/Documents root#
Black1pad:/Applications/DamnVulnerableIOSApp.app/Documents root# file userInfo.plist
userInfo.plist: XML
Black1pad:/Applications/DamnVulnerableIOSApp.app/Documents root#
Black1pad:/Applications/DamnVulnerableIOSApp.app/Documents root# plutil userInfo.plist
{
    password = hackandbeers;
    username = miguel;
}
```

Fichero Userinfo.plist (XML)

Demo iOS: Análisis Estático

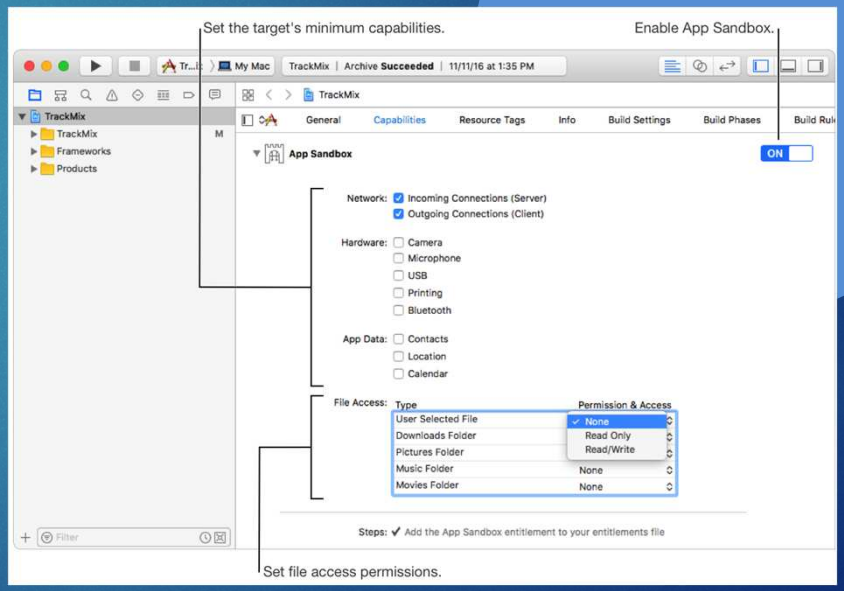
- Análisis de archivos de configuración
- Comprobación del cifrado
- Volcado de clases
- Revisión de métodos y funciones
- Comprobación de activación del ASLR
- Depuración

Demo iOS: Análisis Dinámico

Sandboxing

La implementación del Sandboxing la realiza el desarrollador, en el propio proyecto, desde Xcode.

```
Blacklpad:/ root# ps xau
USER       PID %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
mobile   11904   1.3  1.4   850228  13616   ??  Ss    4:29PM   1:47.55 /usr/libexec/backboardd
mobile   11300   0.1  0.7   711792   7020   ??  Ss   Sun08PM   4:00.69 /System/Library/Private
root     11946   0.0  0.3   723888   2764   ??  Us    5:11PM   0:02.36 /System/Library/Private
mobile   11945   0.0  0.3   709152   3316   ??  Us    5:11PM   0:00.22 /System/Library/Private
mobile   11942   0.0  0.7   710992   7228   ??  Ss    5:10PM   0:00.36 /usr/libexec/atc
mobile   11925   0.0  4.4   741040  44060   ??  Ss    4:38PM   0:02.90 /Applications/Cydia.app/
mobile   11923   0.0  0.2   706992   1536   ??  Ss    4:37PM   0:00.05 /System/Library/Private
mobile   11920   0.0  2.1   736832  20652   ??  Ss    4:34PM   0:04.88 /Applications/Preference
mobile   11916   0.0  0.1   680672    580   ??  Ss    4:30PM   0:00.04 /System/Library/Private
mobile   11915   0.0  0.6   714464   5616   ??  Ss    4:30PM   0:00.49 /Applications/MobileMail
mobile   11914   0.0  0.1   680304    688   ??  Ss    4:30PM   0:00.04 /System/Library/Private
mobile   11913   0.0  0.1   680576    552   ??  SNs   4:30PM   0:00.02 /System/Library/Private
mobile   11912   0.0  0.1   680784   1188   ??  Ss    4:30PM   0:00.08 /Applications/MobileSafa
root     11908   0.0  0.0   680592    352   ??  Ss    4:29PM   0:00.02 /usr/libexec/pfd -d
root     11907   0.0  0.1   680720    696   ??  Ss    4:29PM   0:00.04 /usr/libexec/misd
mobile   11906   0.0  0.3   711120   3400   ??  Ss    4:29PM   0:00.58 /System/Library/Private
mobile   11903   0.0  5.4  1027648  53908   ??  Ss    4:29PM   0:29.36 /System/Library/CoreServ
root     11799   0.0  0.1   539136    980 s000  S     3:50PM   0:00.15 sh
root     11740   0.0  0.1   681600    564   ??  Ss    3:15PM   0:00.03 /System/Library/Private
mobile   11737   0.0  0.2   680016   1520   ??  Ss    3:15PM   0:00.43 /System/Library/Private
mobile   11735   0.0  0.5   709104   5072   ??  Ss    3:15PM   0:00.67 /System/Library/Private
mobile   11734   0.0  0.6   714992   5976   ??  Ss    3:15PM   0:00.64 /System/Library/Framewo
root     11733   0.0  0.1   680624   1316   ??  Ss    3:14PM   0:03.93 /System/Library/Private
_installd 11732   0.0  1.1   711968  10628   ??  Ss    3:14PM   0:20.68 /usr/libexec/installd
mobile   11726   0.0  0.1   539136    756 s000  Ss    3:13PM   0:00.09 -sh
root     11725   0.0  0.1   548880    908   ??  Ss    3:13PM   0:01.86 sshd: mobile@ttys000
```



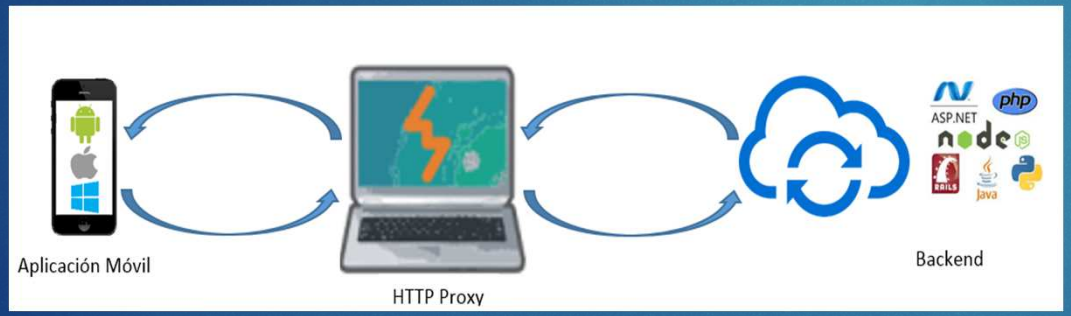
Demo iOS: Análisis Dinámico

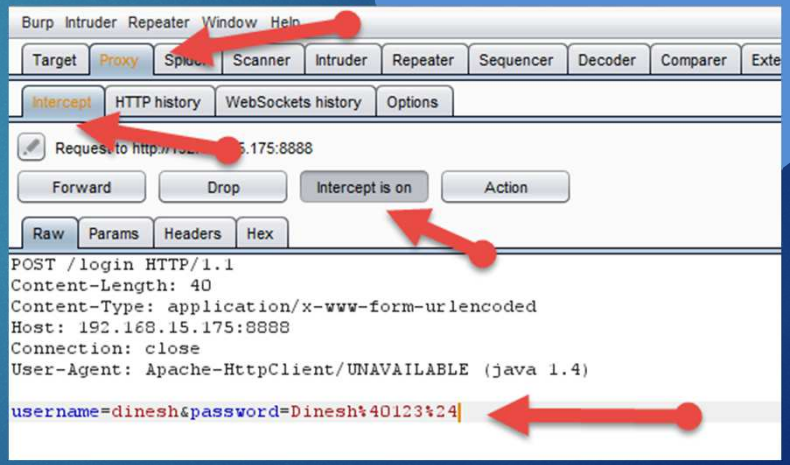
- Exploración de la aplicación mientras se ejecuta
- Invocación de métodos en tiempo de ejecución
- **Análisis de comunicaciones**
- **Hooking**

Demo iOS: Análisis Dinámico

Análisis de Comunicaciones

Es una tarea esencial en toda auditoría de seguridad y consiste en interceptar las peticiones entre el cliente y el servidor. por ejemplo con OWASP ZAP proxy y Burpsuite.





Interceptando Peticiones con Burpsuite

Demo iOS: Análisis Dinámico

Hooking

Un conjunto de técnicas para alterar el comportamiento del software mediante la interceptación de las llamadas a métodos y funciones entre componentes.

Para ello utilizaremos herramientas como: Cycript, Frida, entre otras.

Demo iOS

VIDEO

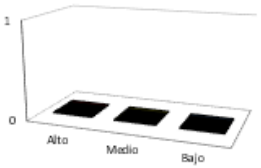
El Informe de Auditoría

3.RESUMEN EJECUTIVO

Nivel de seguridad Global

NIVEL MUY BAJO
0 / 10

Total Vulnerabilidades



RESUMEN	
Nivel	Total
Alto	0
Medio	0
Bajo	0
TOTAL	0

Resumen Ejecutivo

El objetivo del servicio de auditoría de aplicación realizado para NOM-CLIENTE es el de detectar las deficiencias de seguridad que pudieran existir en la aplicación auditada, proponiendo las soluciones más adecuadas a dichas deficiencias. Asimismo, en caso de detectar aspectos de mejora en la seguridad de la aplicación se ofrecen recomendaciones para llevar a cabo dichas mejoras.

Con este objetivo se han realizado distintas actuaciones que han permitido revisar todos aquellos aspectos de seguridad que debieran haberse contemplado durante el desarrollo de la aplicación y que, en caso contrario, derivan en deficiencias y vulnerabilidades de seguridad.

Resumen Ejecutivo

2	Información sensible en base de datos		CVSS 4.9

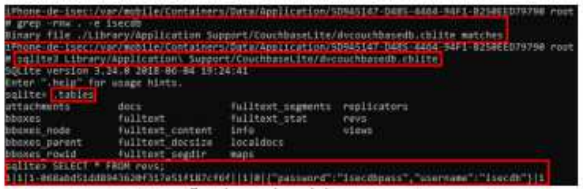
iOS

AV	AC	Au	C	I	A
Local	Baja	Ninguna	Completa	Ninguna	Ninguna

Descripción

Algunas aplicaciones almacenan peticiones en la base de datos de cache para poder acceder a ellas más tarde. Se ha detectado que la aplicación almacena información sensible en la cache (Cache.db y Cache.db-wal):

- Información del usuario
- Tokens
- Contraseña en claro



Credenciales del usuario

Un atacante con acceso al directorio de datos de la aplicación podría obtener información sensible del usuario, incluyendo su contraseña en claro.

Recomendaciones

Se recomienda no almacenar información sensible no cifrada en los ficheros de cache de la aplicación. Se debe especificar explícitamente en el código que no se quiere usar caché, ya que automáticamente se almacenan las peticiones cuando se usa *NSURLCache* y se crea una *NSURLRequest*. Alternativamente, se puede usar SQLcipher de Sqlite para almacenar datos cifrados en base de datos.

Detalle de Vulnerabilidades

Conclusiones

- La Auditoría de Seguridad se ha convertido en una tarea imprescindible para garantizar la confidencialidad, integridad y disponibilidad de la información.
- Su oportuna ejecución evitaría que las aplicaciones móviles se distribuyan con defectos o vulnerabilidades.
- Un entorno 100% seguro es imposible, sin embargo una reducción de la superficie de ataque y de la zona expuesta ayudaría a proteger una app móvil.

FIN

GRACIAS