Universidad Católica de Salta

Facultad de Ingeniería

PROYECTO DE GRADO

Título: **HACKING Y SEGURIDAD DE APLICACIONES MÓVILES**

Alumno: **Giancarlo Bianchi Aguilar**

DNI: 38.212.214

Carrera: **Ingeniería en Informática**

Año 2018

HOJA DE EVALUACIÓN

Proyecto de Grado

***Título:*HACKING Y SEGURIDAD DE APLICACIONES MÓVILES**

***Profesor Guía:*Ing.FERNANDO LUCAS RIVERA BERNSDORFF_____**

***Alumno:*GIANCARLO BIANCHI AGUILAR_____**

EVALUACIÓN

CALIFICACIÓN:_____

TRIBUNAL

Jurado: _____ _____

Jurado:_____ _____

Jurado:_____ _____

OBSERVACIONES:

_____

_____

_____

_____

Lugar y Fecha:

Dedicatoria

Sos una mujer que realmente me llena de orgullo, tu manera de pensar, la solidaridad, el gran corazón que llevas dentro y el sentido del humor invaluable que tenés. Gracias por haberme educado y apoyado, por los mensajes de aliento y tu excelente manera de instruirme para afrontar las verdades de esta vida.

Este proyecto te lo dedico, mamá. Te amo, gracias por tanto.

Agradecimientos

A mis padres, Mirta Gloria Aguilar y Carlos Alberto Bianchi, que con amor me apoyaron en todo el proceso de mi vida.

A mis hermanos, Pamela, Luciana, Gianpaulo y Noelia, gracias no solo por haberme ayudado cuando lo necesité, sino también por enseñarme a compartir y por los lindos momentos que pasamos juntos. Ohana.

A mis profesores, por sus diferentes formas de enseñar, quienes me incentivaron en muchos sentidos y su gran apoyo.

Al profesor Ing. Fernando Lucas Rivera Bernsdorff, por ser un excelente profesor que con sus clases y charlas despierta ese niño interno que todos llevamos dentro y saca lo mejor de cada uno, academicamente y como personas.

A la Dirección de Becas de la UCASAL y Banco Macro, por haberme brindado una beca, la cual me abrió puertas para formarme profesionalmente.

A mis tios Elda Lavilla y Walter Aguilar, por su ayuda y compromiso como parientes, les agradezco de corazón.

A la familia Diaz, quienes me brindaron su gran apoyo, ustedes son como mi segunda familia.

A Cinthia Aguirre, Cesar Cisnero, Alfredo Billordo y Estefanía Rodríguez, gracias por su compañerismo, amistad, y por todos los momentos dentro y fuera de la universidad que pasamos juntos.

A mis compañeros de clase, por el compañerismo y buenos momentos que hemos compartido.

A mis amigos, gracias por estar siempre.

"Son nuestras elecciones... las que muestran lo que verdaderamente somos, mucho más que nuestras habilidades".

- *J.K. Rowling*

Índice de contenido

Índice de ilustraciónes, cuadros y graficos:

RESUMEN EJECUTIVO

Este proyecto tiene como objetivo brindar una herramienta para desarrolladores a los fines que les permitan implementar aplicaciones con menor índice de vulnerabilidad para el sistema operativo *Android*. Se ofrece a los interesados en el tema, la posibilidad de conocer, detectar y solucionar las vulnerabilidades (brechas de ataques) en aplicaciones de los dispositivos móviles, basado en el análisis estático. Esto se concreta a través del desarrollo de un sistema de software, cuyo objetivo es automatizar este proceso de análisis generando un reporte con sus respectivos resultados.

La problemática de la carencia de seguridad y los riesgos implícitos de las aplicaciones móviles vulnerables a los que está expuesta la sociedad en general en calidad de usuarios, es el objeto de estudio de la propuesta de desarrollo del presente trabajo final de grado titulado: HACKING Y SEGURIDAD DE APLICACIONES MÓVILES.

Keywords: Hacking - Seguridad - Aplicaciones Móviles - Sistema Operativo Android – Software Seguridad para Aplicaciones Moviles.

1. INTRODUCCIÓN

1.A. Descripción del problema

Los dispositivos móviles se han declarado protagonistas en la vida cotidiana del ser humano, tanto en el ámbito personal como laboral y/o profesional.Estos equipos están demostrando ser un recurso indispensable ya que, según las estadísticas a enero del año 2018, de 7.593 millones de habitantes del planeta, 5.135 millones cuentan con un dispositivo móvil.Quiere decir que el 68 % de la población mundial tienen posibilidades de acceso a dicha tecnología.[1]

ILUSTRACIÓN Nº 1: Estadìsticas de usuarios de dispositivos móviles y redes

Fuente: Informe de HOOTSUITE

El área de los dispositivos móviles ha tenido un especial crecimiento en los últimos años, gracias a la aparición y al constante desarrollo de tablets y smartphones. Estos ofrecen diversidad de alternativas de herramientas utilitarias, logrando responder a cada uno de los aspectos de la vida diaria. Han entrado a formar parte de nuestra cultura de vida para quedarse, convirtiéndose en una necesidad el estar permanentemente conectado.[2]

[1] Informe de HOOTSUITE - https://hootsuite.com/es/pages/digital-in-2018
[2] Publicación de SEARS - http://www.sears.com.mx/celulares/la-importancia-de-los-celulares-en-la-vida-cotidiana/
Publicación de Instituto Tecnológico y de Estudios Superiores de Monterrey http://www.scielo.org.mx/scielo.php?pid=S0188- 52X2010000100008&script=sci_arttext

La consecuencia del uso de los dispositivos móviles y la necesidad de estar *"en línea"*, obliga a los usuarios a la utilización de las diferentes funciones que éstos nos ofrecen accediendo a una gran variedad de tipos de aplicaciones móviles. La descarga de dichas herramientas, implican el otorgamiento de información personal, como correo electrónico, contraseñas, fecha de nacimiento, sexo y cualquier otro tipo de datos que lo requieran como condición para su uso. La mayoría de los usuarios finales de las aplicaciones, no se cuestionan responsablemente por la seguridad e integridad de la información que se provee, simplemente existe la tendencia a "confiar" que los datos introducidos en dichas aplicaciones móviles son 100% seguras, en pos de recibir el beneficio de su uso.[3] Como consecuencia, las empresas de desarrollo de software y/o desarrolladores, deben brindar a los usuarios la seguridad de los datos que éstos exponen en sus aplicaciones móviles. No obstante**, son pocas las organizaciones que, conscientes de este panorama, implementan las medidas de seguridad necesarias.**

Por otra parte, hay que tener en cuenta que, si una aplicación móvil no es segura en cuanto a su arquitectura y la forma de interactuar con otro tipo de aplicaciones dentro del mismo dispositivo, podrían estar poniendo en riesgo no solamente los datos del usuario actual de la aplicación, sino de **todos sus usuarios**, comprometiendo la seguridad de todo su sistema.A modo de ejemplo, si un usuario puede acceder a componentes no autorizados, podría causar daños intencionales, como provocar gastos para la misma organización, dañar su reputación, etc.

Debido a que los desarrolladores no están centrados en éste aspecto de la seguridad, los *"crackers"* aprovechan este hueco de seguridad, provocando males intencionados.[4]

Son variados y serios los problemas que se deben analizar si una persona con malas intenciones se propone a vulnerar estas aplicaciones móviles: datos comprometidos de los usuarios (que pueden arriesgar desde su estado financiero hasta su

[3]Publicaciòn especializada de DiarioTi.com - https://diarioti.com/el-50-de-los-grandes-desarrolladores-de-apps-moviles-no-invierte-en-seguridad/86664
[4] Casos de "Crackers" de aplicaciones móviles en el 2018
- ✓ http://www.abc.es/tecnologia/redes/abci-instagram-hackean-instagram-y-roban-cuentas-celebrities-culpa-fallo-seguridad-201708311230_noticia.html
- ✓ https://www.clarin.com/economia/hackean-aplicacion-marca-ropa-deportiva-difunden-datos-150-millones-usuarios_0_rJt9laiqG.html

vida personal o el de su familia); acciones con perjuicios económicos para la empresa desarrolladora; mala reputación de la misma.

Esta problemática debe ser tratada por las empresas desarrolladoras de aplicaciones móviles, tomando conciencia de la Responsabilidad Social[5] que le compete, e implementando los métodos de seguridad necesarios ante cada caso, garantizando los siguientes aspectos[6]:

- ✓ **Confidencialidad**: Se refiere a la privacidad de la información almacenada y procesada en un sistema informático. Las herramientas de seguridad informática deben proteger el sistema de intrusos y accesos por parte de personas y/o programas no autorizados.
- ✓ **Integridad**: Se refiere a la validez y consistencia de los elementos de información almacenados y procesador en un sistema informático. Basándose en este principio, las herramientas de seguridad deben asegurar que los procesos de actualización estén bien sincronizados y no se dupliquen, de forma que todos los elementos del sistema manipulen adecuadamente los mismos datos.
- ✓ **Disponibilidad**: Se refiere a la continuidad de acceso a los elementos de información almacenados y procesados en un sistema informático.

1.B. Carácter del problema

En el área de la seguridad de las aplicaciones móviles, se han intentado mejorar los indicadores a lo largo de los años, poniendo atención a vulnerabilidades comunes que se presentan en diferentes tipos de App, Últimamente se ha incrementado masivamente el número de desarrollo de estas herramientas, sin enfocarse en la seguridad de las mismas, lo que ha generado consecuencias negativas en los propietarios de datos y accesos al sistema.

[5] Responsabilidad Social Empresaria desde la Informática
- ✓ https://www.eleconomista.com.mx/empresas/Seguridad-informatica-como-responsabilidad-20150406-0135.html
- ✓ https://es.slideshare.net/avanet/la-responsabilidad-social-de-la-ingeniera-de-software

[6] KennethAmaditz; Informe especializado en Trustdimension -http://www.trustdimension.com/la-importancia-de-la-seguridad-informatica/

Se han desarrollado investigaciones que se han hecho a lo largo del tiempo, con políticas públicas destinadas a promover este cambio de paradigma al desarrollar las aplicaciones móviles. No obstante, los índices de vulnerabilidad en las aplicaciones actuales siguen en constante crecimiento.Se puede decir entonces que las empresas están enfocadas directamente al desarrollo de App vulnerables. Es por eso que resulta relevante visualizar profundamente esta práctica, analizarla y sentar las bases para que abra el camino a otras modalidades y territorios de atención.

Es preciso **relevar información** respecto a este tipo de problemática, ya que en la actualidad, si bien hay investigaciones realizadas respecto a esta práctica, no se están tomando las medidas necesarias debido al costo que implica, ya sea invertir en una empresa que se dedica a ello o la capacitación de los mismos desarrolladores.

1.C. Motivación para el abordaje del tema

La motivación personal que impulsa mi abordaje sobre éste tema, son los valores éticos y de responsabilidad profesionales con que considero se deben manejar los desarrolladores en el ámbito de la informática, especialmente teniendo en cuenta que, con las App, se expone a la sociedad en general como consecuencia del uso masivo de los dispositivos móviles. La problemática de la carencia de seguridad y los riesgos implícitos de las aplicaciones móviles vulnerables, es el objetivo del presente trabajo final de grado.

La solución que se plantea en el presente proyecto, tiene como objetivo también, la capacitación tomando como eje central el tema de concientización del desarrollador sobre la seguridad informática y sus responsabilidades respecto de la protección de sus usuarios. De éste modo, introduciéndolo al mundo de la seguridad, se intentará que al momento de desarrollar aplicaciones destinadas a cualquier plataforma, el profesional sea consciente de aplicar las mismas métricas aprendidas, *investigando como se aplican hacia la plataforma que está desarrollando.*

Por otra parte, además de la capacitación, se brindará mecanismos de seguridad automatizados, acelerando el proceso de confianza durante el desarrollo de las aplicaciones. El presente proyecto permitirá:

- Aplicar normas de seguridad de datos.
- Identificación de ataques al sistema (tipos de ataque, nivel de riesgo, entre otras características), juntamente con el almacenamiento de la información referida.
- Técnicas de protección de Aplicaciones móviles.

Además, al ser una solución que implica bajos costos para las empresas, permitirá a los desarrolladores una capacitación acelerada y eficiente con respectos a otros métodos de aprendizaje,

1.D. Pasos del Proyecto

Los pasos a realizar para el presente desarrollo serán:

1. Selección del sistema operativo móvil.
2. Identificar la arquitectura de una aplicación para dicho sistema operativo.
3. Identificar archivo/s principal/es que contiene información relevante de la aplicación.
4. Analizar quétipos de vulnerabilidades se podrían detectar en dichos archivos.
5. Desarrollo del software:
   a) Diseño de la arquitectura de la solución.
   b) Selección de un lenguaje de programación para hacer análisis de expresiones regulares.
   c) Selección de framework/biblioteca para mostrar la salida de la solución.
   d) Codificación
   e) Testing con aplicaciones desarrolladas para prueba.
   f) Elaboración documental.

1.E. Viabilidad Del Proyecto - Criterios de éxito.

El presente proyecto, debe ser analizado a través de diferentes puntos de vista. Identificando los diferentes aspectos de análisis de factibilidad.

- ✓ **Económico**: la inversión del presente proyecto debe superar los costos de contratar una empresa de seguridad informática. De esta manera, reducirán sus costos de manera significativa.
- ✓ **Técnico**: conocimientos y recursos necesarios para el desarrollo del proyecto.
- ✓ **Operativo**: la empresa contratista del presente proyecto debe tener los recursos disponibles y personal capacitado para operar satisfactoriamente.
- ✓ **Legal**: no debe infringir ninguna ley y norma legal. Haciendo uso de principios éticos.

En cuanto a aspectos técnicos observamos:

- ✓ Desarrollo de una herramienta que permita hacer análisis de código estático de la aplicación.
- ✓ Identificar vulnerabilidades.
- ✓ Proporcionar la posibilidad de aplicar mecanismos de seguridad automatizados a las vulnerabilidades encontradas.
- ✓ Informar sobre las vulnerabilidades encontradas.
- ✓ Capacitar al desarrollador sobre los mecanismos de seguridad que se han aplicado.

2. ESTADO DE LA CUESTIÓN

2.A. Antecedentes relacionados con el problema

- [7]**Título**: "*Evaluación de la seguridad de las aplicaciones móviles bancarias*"

  Tipo de Investigación: Tesis para optar al grado de magister en ciencias, mención computación.

  Autores: Cristián Andrés Rojas Poblete

  Institución: Universidad Católica de Chile.

  Año de Publicación: 2016

  Aporte para la Investigación: Esta tesis tiene como objetivo clasificar las aplicaciones bancarias móviles ofrecidas por los bancos nacionales dentro de la taxonomía, y dar lineamientos respecto de buenas prácticas de seguridad al momento de diseñar, implementar y lanzar al mercado aplicaciones móviles que manejan información sensible para sus usuarios.

- [8]**Título**: "*Seguridad en dispositivos móviles: un enfoque práctico*"

  Tipo de Investigación: Investigación cientifica

  Autores: Lic. Nicolás Macia, Lic. EinarLanfranco, Lic Paula Venosa.

  Institución: Universidad Nacional de la Plata. Facultad de informática. Laboratorio de investigación de nuevas tecnologías informáticas.

  Año de Publicación: 2014

  Aporte para la Investigación: El aporte de esta investigación es amplio y directo, ya que analiza los distintos problemas de seguridad a los que una persona se expone utilizando los dispositivos móviles, incluso cuando se les da un uso adecuado, creando pruebas de conceptos que permiten ejemplificar los problemas posibles y

[7] Biblioteca virtual Universidad Católica de Chile.
- ✓ http://repositorio.uchile.cl/bitstream/handle/2250/144529/Evaluaci%C3%B3n-de-la-seguridad-de-aplicaciones-m%C3%B3viles-bancarias.pdf?sequence=1

[8] Biblioteca virtual Universidad Nacional de la Plata. Facultad de informática. Laboratorio de investigación de nuevas tecnologías informáticas
- ✓ http://sedici.unlp.edu.ar/bitstream/handle/10915/43678/Documento_completo.pdf?sequence=1

metodologías de análisis que permitan determinar la existencia de amenazas de este tipo de dispositivos.

- [9]**Título**: "*Estudio y análisis de seguridad en dispositivos móviles. BYOD y su impacto en las organizaciones.*"

  Tipo de Investigación: tesina

  Autores: Pacheco Veliz, Sebastian Exequiel

  Institución: Universidad Nacional de la Plata.

  Año de Publicación: 2016

  Aporte para la Investigación: Esta tesis trata sobre el impacto del uso de los dispositivos móviles personales en las organizaciones lo cual se conoce como: BringYourOwnDevice (BYOD). Este fenómeno constituye un campo de interés para que quienes estudianproblemáticas relacionadas a la seguridad de la información en dispositivos móviles, puedan expandir sus análisis a ambientes organizacionales.

2.B. Fundamentos Teóricos y Prácticos que cimientan las soluciones escogidas.

Las aplicaciones móviles solo pueden ser tan seguras como la base sobre la cual están construidas; los dispositivos móviles y los sistemas operativos en los que se ejecutan. Es imperativo entender los riesgos inherentes asociados con los dispositivos móviles, las medidas de seguridad nativas en los sistemas operativos móviles y las mejores prácticas para mitigar los riesgos de seguridad de las aplicaciones móviles.[10]

[9] Biblioteca virtual Universidad Nacional de la Plata.
- ✓ http://sedici.unlp.edu.ar/bitstream/handle/10915/58591/Documento_completo.%20y%20Piazza%20Orlando,%20C.%20Estudio%20y%20an%C3%A1lisis%20de%20seguridad%20en%20dispositivos%20m%C3%B3viles.pdf-PDFA.pdf?sequence=4

[10] Search Data Center; Guía Esencial: Mejores prácticas para el gobierno de TI en las empresas
- ✓ https://searchdatacenter.techtarget.com/es/cronica/Mejores-practicas-en-seguridad-de-aplicaciones-moviles-para-proteger-datos-corporativos

Los ecosistemas móviles están muy por detrás de la infraestructura. Es por eso, que es necesario analizar la base sobre la que se construyen las aplicaciones móviles.

Tipos de Aplicaciones móviles

Existen diferentes tipos de aplicaciones móviles en el mercado, generalmente se suelen distinguir entre:

- ✓ WebApp.
- ✓ Nativas.
- ✓ Hibridas.

Una aplicación basada en Web (WebApp), es exactamente lo mismo que decir una aplicación desarrollada con tecnologías web como JavaScript o HTML5 para proveer interacción, navegación y capacidades de personalización. Se ejecutan dentro del navegador web del dispositivo móvil y son renderizadas por una petición de página web desde el servidor *backend*. No es raro ver la misma aplicación utilizada como una aplicación representada por el navegador habitual y como una aplicación, ya que proporciona beneficios de no duplicar los esfuerzos.[11]

A diferencia de las aplicaciones basadas en web, las **aplicaciones móviles nativas** proporcionan un alto rendimiento y un alto grado de confiabilidad. Brindan un tiempo de respuesta rápido ya que toda la aplicación no es recuperada del servidor y puede aprovechar la solidez del soporte nativo proporcionado por el sistema operativo. Además, los usuarios pueden usar algunas aplicaciones sin la conexión a internet. Sin embargo, las aplicaciones desarrolladas usando tecnologías nativas no son una plataforma independiente y están vinculadas a una plataforma móvil en particular, por lo que las empresas buscan disminuir el esfuerzo al momento de desarrollar una aplicación para diferentes sistemas operativos.[12]

Las aplicaciones móviles hibridas intentan combinar lo mejor de ambas partes, aplicaciones nativas y aplicaciones basadas en web. Estas se ejecutan en un

[11]Kotipalli& Imran, 2016.Hacking Android.Página 108.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.
[12]Kotipalli& Imran, 2016.Hacking Android.Página 108.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.

dispositivo como una aplicación móvil nativa pero desarrollada con tecnologías web (*HTML5, CSS y JavaScript*). Las aplicaciones hibridas se ejecutan dentro de un contenedor nativo y aprovechan el motor del navegador (pero no el navegador) para representar el *HTML* y procesar el *JavaScript* localmente. Una capa de abstracción de web a nativo habilita el acceso a los recursos de un dispositivo que no pueden ser accedidas por una aplicación basada en web, como el acelerómetro, cámara y almacenamiento local. Normalmente, estos tipos de aplicaciones son desarrolladas usando un *framework* como *PhoneGap* o *ReactNative*, sin embargo no es poco común que las organizaciones creen sus propios contenedores. [13]

El termino seguridad como la ausencia de peligro, daño o riesgo.[14]

La seguridad consiste en hacer que el riesgo se reduzca a niveles aceptables, debido a que el riesgo es inherente a cualquier actividad y nunca puede ser eliminado.

Cuando nos referimos a **seguridad informática** hace referencia al proceso de prevenir y detectar el uso no autorizado de un sistema informático. Implica el proceso de proteger contra intrusos el uso de nuestros recursos informáticos con intenciones maliciosas o con intención de obtener ganancias, o incluso la posibilidad de acceder a ellos por accidente.[15]

Es por eso, que se ha desarrollado un proyecto de código abierto denominado ***OWASP Mobile Security*** perteneciente a **OWASP.** Este proyecto es el más conocido en el ambito, dada a la gran colaboración de las empresas que contribuyen de manera directa al proyecto, aportando datos y conocimientos.

[13]Kotipalli& Imran, 2016.Hacking Android.Página 108.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.
[14] Publicación de Forodeseguridad - http://www.forodeseguridad.com/artic/discipl/4163.htm
[15] Informe especializado en Universidad Internacional de Valencia - https://www.universidadviu.es/la-seguridad-informatica-puede-ayudarme/

Projects/OWASP Mobile Security Project[16]

Open Web Application Security Project (*OWASP*)[17] es una organización sin ánimo de lucro a nivel mundial dedicada a mejorar la seguridad de las aplicaciones y del software en general. Su misión es hacer que la seguridad dentro de las aplicaciones sea más visible para que, así, las organizaciones y los particulares puedan tomar decisiones sobre conceptos de seguridad basándose en información verídica y contrastada.

Es importante destacar que cualquiera puede participar en *OWASP* y que todos sus proyectos, materiales y documentación están disponibles de forma gratuita. Del mismo modo, y algo importante, es que todos los productos y servicios recomendados no son productos comerciales, sino que son productos también libres y de código abierto.

Como organización, tienen una serie de valores:

- ✓ Transparencia: todo lo relativo a *OWASP* es transparente, desde sus finanzas (que pueden ser consultadas directamente desde su página web) hasta el código de sus proyectos (los repositorios de código también están accesibles).
- ✓ Innovación: promueven y favorecen la experimentación para aquellas soluciones a los nuevos desafíos de seguridad que aparecen.
- ✓ Global: cualquier persona de cualquier parte del mundo está invitada a participar en la comunidad de *OWASP* (se puede aplicar desde un formulario).
- ✓ Integridad: se declaran como una comunidad honesta y en la que se puede confiar. También se declaran neutrales sobre el aspecto comercial de los productos de seguridad disponibles en el mercado, no posicionándose sobre alguno de ellos.

Por lo tanto, *OWASP* pretende ser el centro de referencia de toda la información de seguridad enfocada a las aplicaciones web. Ya veremos que no sólo información sino también estudios, soluciones y proyectos con los que nos ayudarán a los desarrolladores a hacer que nuestras aplicaciones sean más seguras.

[16] Publicación de OWASP - https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks
[17] RafaelVindel Amor; Publicación en Adictos al trabajo - https://www.adictosaltrabajo.com/tutoriales/introduccion-a-owasp/

Es necesario aclarar, la existencia de otras metodologías existentes en el mercado, tales como *OSSTMM, ISSA, PTES*. Desarrolladas con el fin de establecer un estándar para realizar pruebas de penetración. Para el presente proyecto, escogi OWASP ya que proporciona detalladamente información destinada a desarrolladores y equipos de seguridad acerca de los recursos necesarios que se necesitan para crear y mantener aplicaciones móviles seguras, el cual se adapta para proporcionar una solución a la problemática planteada.

OWASP Mobile Security[18] es un proyecto que tiene la intención de brindar recursos necesarios para que las aplicaciones móviles sean más seguras. A través de este proyecto, el objetivo es clasificar los riesgos de seguridad móviles y proporcionar controles en el desarrollo para reducir su impacto y probabilidad de explotación.

El enfoque principal es en la capa de aplicación haciendo hincapié en las aplicaciones móviles desplegadas en los dispositivos de usuario final y en la infraestructura de servidor que comunican esas aplicaciones móviles, así como en la integración entre las aplicaciones, los servicios de autenticación remota y las características específicas de la plataforma en la nube.

Existe, en este proyecto, un Top 10 de vulnerabilidades en aplicaciones móviles que se identifican todos los años. Con el objetivo de comenzar a hacer hincapié en solucionar estas vulnerabilidades principalmente, desde el punto de vista de desarrollador.

Muchas organizaciones de seguridad informática, al momento de hacer un análisis de vulnerabilidades, toman como referencia este Top 10.

Los datos para generar el top 10 son proporcionados por diferentes organismos, este análisis se realiza todos los años.[19]

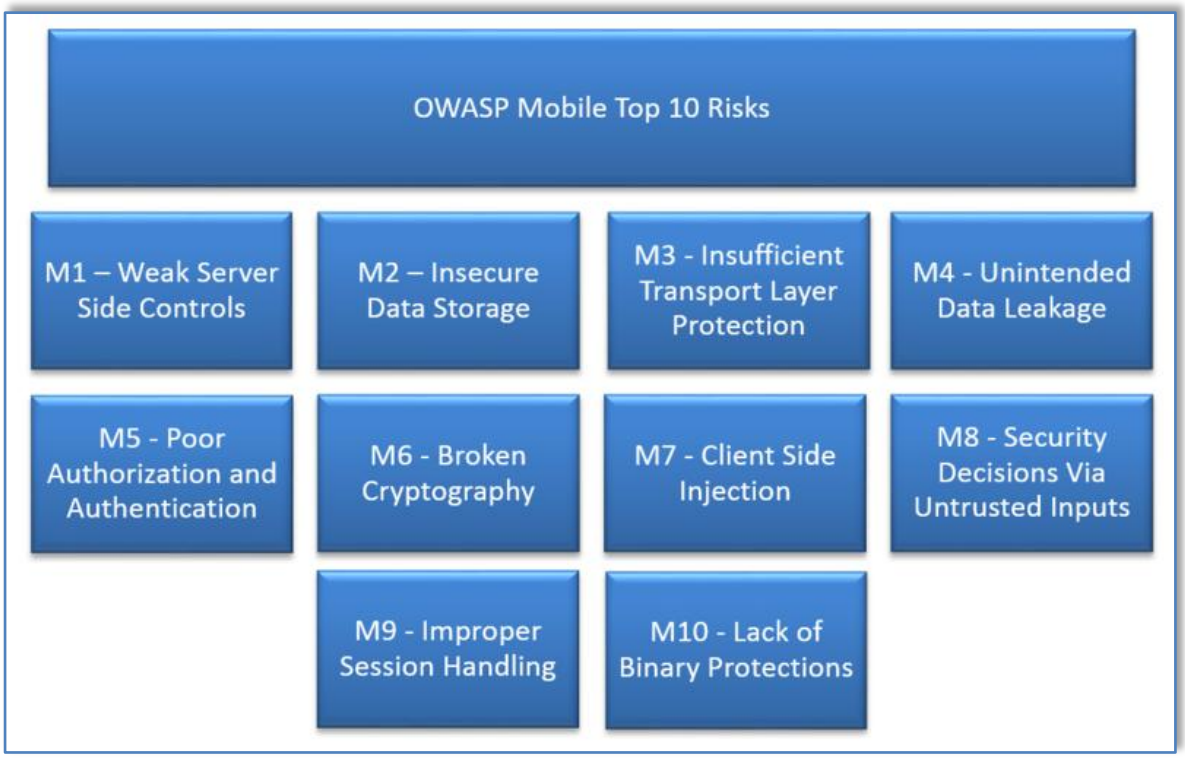
A continuación, se hace mención de las 10 vulnerabilidades en aplicaciones móviles, más comunes. Esto, corresponde al Top Ten de Owasp Mobile

[18]Publicación en Segu.Info - https://blog.segu-info.com.ar/2015/03/owasp-mobile-security-espanol.html
[19] Mobile Top Ten Contributions - https://www.owasp.org/index.php/Mobile_Top_Ten_Contributions

Security correspondiente al año 2014. He tomado la referencia de este año (2014), ya que más adelante al hacer uso de la bibliografía del libro "*Hacking Android - SrinivasaRaoKotipalli, Mohammed A. Imran*" hace mención al Top 10 del año mencionado.

El presente cuadro representa las 10 vulnerabilidades más comunes en una aplicación móvil encontradas hasta el año 2014.

CUADRO Nº 1: 10 Vulnerabilidades más comunes de App



Fuente: Mobile Top Ten Contributions - https://www.owasp.org/index.php/Mobile_Top_Ten_Contributions

- **M1: Weak Server SideControls – Controles pobres en el servidor.**

Controles pobres en el servidor habla acerca de de los ataques al *backend* de la aplicación. Muchas aplicaciones usan conexión a internet para comunicarse con el backend usando *REST* o *SOAPAPIs*. La seguridad principal asociada con el tradicional *webservers* y *web application* es que se puede usar el mismo *backend*, para diferentes tipos

de aplicaciones *frontend* (Cliente móvil en nuestro caso). Normalmente, el vector de ataque está en encontrar los puntos de entrada de la *Api* y abuzar de varias vulnerabilidades, explotando malas configuraciones de servidores. [20]

Entre las vulnerabilidades más comunes que podemos encontrar tenemos:

- ✓ Inyección *SQL*, *XSS*.
- ✓ Defectos de autenticación.
- ✓ Defectos de gestión de sesiones.
- ✓ Vulnerabilidades de control de acceso.
- ✓ Utilizar componentes con vulnerabilidades conocidas.

Inyección *SQL* [21] es una técnica donde un atacante crea o altera comandos *SQL* existentes para exponer datos ocultos, sobrescribir los valiosos, o peor aún, ejecutar comandos peligrosos a nivel de sistema en el equipo que hospeda la base de datos. Esto se logra a través de la práctica de tomar la entrada del usuario y combinarla con parámetros estáticos para elaborar una consulta *SQL*.

Consiste en modificar valores que la aplicación usa para pasar variables entre dos *layout*. Un clásico ejemplo de esto es hacer que a través de un buscador se ejecute un mensaje de alerta en *JavaScript*. Con *XSS* reflejado, el atacante podría robar las cookies para luego robar la identidad, pero para esto, debe lograr que su víctima ejecute un determinado comando dentro de su dirección.[22]

La autenticación adecuada y la gestión de sesiones son críticas en la seguridad de las aplicaciones. Deficiencias en estas áreas, con mucha frecuencia implican fallas en la protección de credenciales y testigos (*tokens*) de sesión a través de su ciclo de vida. Estos defectos pueden conducir a un robo de usuarios ó cuentas de administración, sabotaje de los controles de autorización y registro, causando violaciones a la privacidad.[23]

[20] *Kotipalli& Imran, 2016.Hacking Android.Página 115.*353 Páginas.*Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*
[21] Publicación en página especializada Php - http://php.net/manual/es/security.database.sql-injection.php
[22] Ignacio Pérez; Informe especializado en Welivesecurity - https://www.welivesecurity.com/la-es/2015/04/29/vulnerabilidad-xss-cross-site-scripting-sitios-web/
[23] Publicación de OWASP - https://www.owasp.org/index.php/Top_10_2007-Autenticaci%C3%B3n_y_Gesti%C3%B3n_de_Sesiones_Disfuncional

Las deficiencias en el mecanismo principal de autenticación son comunes, pero con mayor frecuencia se presentan a través de las funciones auxiliares de autenticación como el cierre de sesión, gestión de contraseñas, expiración de sesión, recuérdame, pregunta secreta y actualización de cuenta.

Vulnerabilidades de control de acceso referencia a lograr encontrar archivos, documentos, rutas, en donde previamente tendría que existir una autenticación de usuario y un usuario logra encontrarlas sin pasar por ningún control de autenticación.

Utilizar componentes con vulnerabilidades conocidas el cual el atacante identifica o conoce mediante escaneos o análisis manuales, los componentes débiles de una aplicación web como los *frameworks* y ejecuta el ataque.[24]

Se refiere al uso de componentes tales como librerías, *frameworks* y otros módulos de software, que en muchas ocasiones funcionan con todos los privilegios. Si se ataca un componente vulnerable esto podría facilitar la intrusión en el servidor o una pérdida de datos.

- **M2: Insecure Data Storage – Almacenamiento de datos inseguros.**

Los desarrolladores asumen que los datos almacenados en el sistema de archivo de un dispositivo no son accesibles por los atacantes. Con esta suposición ellosfrecuentemente almacenan datos sensibles como nombre de usuarios, autenticación basada en *Token*, contraseñas, *Pins*, información personal y direcciones en el sistema de archivo del dispositivo usando conceptos como archivos de preferencias o bases de datos SQLite. Hay múltiples maneras de acceder a estos datos que están almacenados de manera local. La técnica más común sería iniciar el dispositivo móvil con accesos privilegiados *("rootear")* yllegar a los datos, usando ataques basados en copias de seguridad (*backcup*). [25]

Tipos de estructuras y datos que almacenan las aplicaciones móviles:

[24] Liliana CújarBahamón; Publicación en liliseguridadinformatica - https://liliseguridadinformatica.webnode.es/guia-de-buenas-practicas/diseno/a9/
[25] *Kotipalli& Imran, 2016.Hacking Android.Página 115.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*

- ✓ Bases de datos *SQLit.*
- ✓ Archivos de *log*.
- ✓ Archivos Manifiesto.
- ✓ Cookies almacenados: datos cacheados por la aplicación.
- ✓ Archivos de preferencias.

Base de datos *SQLite*[26] es un sistema de gestión de bases de datos relacional compatible con *ACID*, contenida en una relativamente pequeña (~275 kiB) biblioteca escrita en *C*. *SQLite* es un proyecto de dominio público creado por D. Richard Hipp.

Debido a su pequeño tamaño, *SQLite* es muy adecuado para los sistemas integrados, y también está incluido en diferentes dispositivos móviles con sistema operativo como *Android&Ios*.

Los Archivos de registro[27](o archivos de *log*) son archivos que contienen mensajes sobre el sistema, incluyendo el *kernel*, los servicios y las aplicaciones que se ejecutan en dicho sistema. Existen diferentes tipos de archivos de log dependiendo de la información. Por ejemplo, existe un archivo de log del sistema, un archivo de *log* para los mensajes de seguridad y un archivo de log para las tareas cron.

En el caso de los dispositivos móviles, este archivo de *log* podría estar almacenado de manera inseguro y un intruso tendría la posibilidad deacceder a los mismos con fines mal intencionados.

El archivo manifiesto es el archivo en formato *XML*, en nuestro caso, de una aplicación móvil. Este archivo se podría definir como la "columna vertebral" de una aplicación. Este archivo es uno de los más importantes del sistema ya que presenta información relevante sobre la aplicación:

- ✓ Mínima versión de Api en que se ejecuta.

[26] Publicación en página especializada Sqlite - http://www.sqlite.org/about.html
[27]Informeespecializadoen Massachusetts Institute of Technology - http://web.mit.edu/rhel-doc/3/rhel-sag-es-3/ch-logfiles.html

- ✓ Componentes de una aplicación.
- ✓ Estructura de una aplicación.
- ✓ Acciones que se pueden ejecutar en cada componente.
- ✓ Nombre del paquete.
- ✓ Permisos que requiere para instalar.
- ✓ Bibliotecas con las que tiene que estar vinculada.

Se observaque en este archivo podría encontrar información relevante para hacer el análisis de una aplicación.

***Cookies* almacenados** también llamada galleta informática es una pequeña información enviada por un sitio web y almacenado en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del usuario.

Sus principales funciones son:

- ✓ Llevar el control de usuarios: cuando un usuario introduce su nombre de usuario y contraseña, se almacena una galleta para que no tenga que estar introduciéndolas para cada página del servidor. Sin embargo, una galleta no identifica a una persona, sino a una combinación de computadora de la clase de computación-navegador-usuario.
- ✓ Conseguir información sobre los hábitos de navegación del usuario, e intentos de spyware (programas espía), por parte de agencias de publicidad y otros. Esto puede causar problemas de privacidad y es una de las razones por la que las cookies tienen sus detractores.

Los archivos de preferencia son archivos que contienen información que almacena en forma de clave-valor, es decir, cada una de ellas estará compuesta por un identificador único (p.e. “email”) y un valor asociado a dicho identificador (p.e. “prueba@email.com”). Además, y a diferencia de SQLite, los datos no se guardan en un fichero binario de base de datos, sino en ficheros XML

El modo de acceso indicará qué aplicaciones tendrán acceso a la colección de preferencias y qué operaciones tendrán permitido realizar sobre ellas. Así, tendremos tres posibilidades principales:

- ✓ *MODE_PRIVATE*. Sólo nuestra aplicación tiene acceso a estas preferencias.
- ✓ *MODE_WORLD_READABLE*. Todas las aplicaciones pueden leer estas preferencias, pero sólo la nuestra puede modificarlas.
- ✓ *MODE_WORLD_WRITABLE*. Todas las aplicaciones pueden leer y modificar estas preferencias.

- ➢ **M3: Insufficient Transport Layer Protection - Insuficiente protección de la capa de transporte**

Similar a las aplicaciones webs, las App móviles pueden transmitir información sensible a través de redes inseguras, que puede conducir a ataques graves. Esto es muy común en un café-bar y aeropuertos que disponen de wifi abierto, en donde atacantes maliciosos pueden realizar un ataque *MITM* (*ManintheMidle*) para interceptar datos que envían los usuarios a través de la red.

Cuando se realiza un proceso de *pentesting*, hay varios escenarios en donde la aplicación puede pasarinformación a través de la red, contraseñas de credenciales o *tokens* de sesiones. Por lo tanto, siempre es una opción para analizar el tráfico de una aplicación a los fines deverificarsi está pasando información sensible a través de la red.

Hay otro escenario importante donde la mayoría de las aplicaciones son vulnerables. Si una aplicación está utilizando autenticación con el protocolo de comunicación *HTTPS* y enviando autenticación de cookies a través de *HTTP*, la autenticación es vulnerable desde el punto de vista de un atacante, ya que puede fácilmente obtener la autenticación basada en cookies que es pasada a través del protocolo *HTTP*, y estas cookies son poderosas como nombres de usuarios y contraseñas para iniciar sesión en

una aplicación. La ausencia de certificados de verificación y débiles procesos de intercambio de comunicación privada son problemas comunes en la capa de transporte. [28]

Es importante analizar, desde el punto de vista de un atacante, el alcance de información que podría llegar a obtener:

- ✓ Información privada utilizando protocolos sin ninguna encriptación.
- ✓ Información encriptada pero utilizando claves "*hardcodeadas*".
- ✓ No validar los certificados al usar *SSL*/*TSL*.

SSL (*SecureSocketsLayer*) [29] traducido al español significa Capa de Conexiones Seguras. Es un protocolo que hace uso de certificados digitales para establecer comunicaciones seguras a través de Internet. Recientemente ha sido sustituido por *TLS* (*TransportLayerSecurity*) el cual está basado en *SSL* y son totalmente compatibles.

Permite confiar información personal a sitios web, ya que los datos se ocultan a través de métodos criptográficos mientras navegas en sitios seguros.

Es utilizado ampliamente en bancos, tiendas en línea y cualquier tipo de servicio que requiera el envío de datos personales o contraseñas.

El cifrado es el proceso que transforma tu información de manera que no cualquier usuario pueda entenderla, se realiza con base a un elemento único conocido como llave, así nadie, excepto el poseedor puede leerla. El procedimiento inverso al cifrado es el descifrado.

Llave Pública y Llave Privada

Son un par de "llaves" digitales asociadas a una persona o entidad y generadas mediante métodos criptográficos. La llave pública es usada para cifrar la información, haciendo una analogía, es como la llave utilizada para cerrar una puerta y mantener fuera a cualquier persona mientras que la llave privada se usa para descifrar, es

[28] *Kotipalli& Imran, 2016.Hacking Android.Página 115.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*
[29] Dante Odín Ramírez López, Carmina Cecilia Espinosa Madrigal; Informe especializado en Universidad Autónoma de México
- ✓ http://revista.seguridad.unam.mx/numero-10/el-cifrado-web-ssltls

decir, la llave que abre la puerta y sólo la posee la persona autorizada, por lo tanto ésta debe mantenerse en secreto.

Firma Digital[30]

Una firma digital es un mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente identificar a la entidad originadora de dicho mensaje (autenticación de origen y no repudio), y confirmar que el mensaje no ha sido alterado desde que fue firmado por el originador (integridad).

Autoridad Certificadora(*AC*)

Una Autoridad Certificadora (*AC*, en inglés *CA*) es una entidad confiable que se encarga de garantizar que el poseedor de un certificado digital sea quien dice ser, bridando confianza a ambas partes de una comunicación segura *SSL*/*TLS*.

Certificado Digital *SSL*/*TLS*

Es un documento digital único que garantiza la vinculación entre una persona o entidad con su llave pública.

Contiene información de su propietario como nombre, dirección, correo electrónico, organización a la que pertenece y su llave pública, así como información propia del certificado por mencionar: periodo de validez, número de serie único, nombre de la *AC* que emitió, firma digital de la *AC* cifrada con su llave privada y otros datos más que indican cómo puede usarse ese certificado.

HTTPS: Simplemente es una combinación del protocolo *HTTP* (usado en cada transacción web) con el protocolo *SSL*/*TLS* usada para establecer comunicaciones cifradas en sitios web.

Información privada utilizando protocolos sin ninguna encriptación: hace referencia a los protocolos de comunicación que no son seguros. Por ejemplo el protocolo *HTTP*. Si una

[30]Boris Balacheff et ali., "Trusted computing platforms. TCPA technology in context". Prentice Hall PTR 2003
Dennis K. Branstad,"Report of the Nist Workshop on Digital Signature Certificate anagement". U.S. Department of Commerce 1983

aplicación envía datos al servidor a través de este tipo de protocolo, podría ser interceptada por un atacante y robar la información que se está enviando.

Información encriptada pero utilizando claves "*hardcodeadas*" :el termino *Hardcode*[31]hace referencia inicialmente a malas prácticas en el desarrollo de software que consiste en incrustar datos directamente en el código fuente del programa, en lugar de obtener esos datos de una fuente externa como un fichero de comunicación o parámetros, o un archivo de recursos.Si tenemos claves que han sido generadas de esta manera, la comunicación podría estar comprometida ya que un atacante podría descifrar la clave y por ende la comunicación.

No validar los certificados al usar *SSL/TSL*

Cuando el navegador tiene el certificado del sitio al que desea acceder, realiza algunas verificaciones antes de confiar en el sitio:

- ✓ Integridad del certificado: Verifica que el certificado se encuentre íntegro, esto lo hace descifrando la firma digital incluida en él mediante la llave pública de la *AC* y comparándola con una firma del certificado generada en ese momento, si ambas son iguales entonces el certificado es válido.
- ✓ Vigencia del certificado: Revisa el periodo de validez del certificado, es decir, la fecha de emisión y la fecha de expiración incluidos en él.
- ✓ Verifica emisor del certificado: Hace uso de una lista de Certificados Raíz almacenados en tu computadora y que contienen las llaves públicas de las *ACs* conocidas y de confianza (Imagen 2). Puedes acceder a esta lista desde las opciones avanzadas de tu navegador web (en este caso usamos *GoogleChrome*).

Tomando como base esta lista, el navegador revisa que la AC del certificado sea de confianza, de no serlo, el navegador mostrará una advertencia indicando que el certificado fue emitido por una entidad en la cual no confía.

[31]Publicación de Babylon - http://thesaurus.babylon-software.com/hard-coded

- **M4: Unintened Data Leakage: Fuga de datos no intencionada**

Cuando una aplicación maneja información sensible originariade una entrada por el usuario (*input*) o de cualquier otro recurso, dichos datos podríanalojarseen una ubicación insegura del dispositivo. Esta localización insegura brinda la posibilidad de accesoa otras aplicaciones maliciosas que están instaladas en el mismo dispositivo, consecuentemente, dejando el estado del equipoen un estado grave. El código se vuelve vulnerable a ataques serios. Explotar estas vulnerabilidades se convierte fácilmente para un atacante, pudiendosimplemente escribir una pieza de código muy corta para acceder a la localización donde los datos sensibles están almacenados. Podemos usar herramientas como *adb* para acceder a estas direcciones.

A continuación se exponeuna lista de ejemplos de escenarios en donde existen fugas de datos no intencionadas:

- ✓ Fugas de proveedor de contenidos.
- ✓ Copiar / Pegar el almacenamiento en memoria cache del *buffer*.
- ✓ Archivo de registro (*Logging*).
- ✓ Almacenamiento en caché de URL.
- ✓ Objetos cookies del navegador.
- ✓ Datos analíticos enviados a terceros. [32]

Un proveedor de contenidos no es más que el mecanismo proporcionado para permitir compartir información entre aplicaciones. Una aplicación que desee que todo o parte de la información que almacena esté disponible de una forma controlada para el resto de aplicaciones del sistema deberá proporcionar un *contentprovider* a través del cual se pueda realizar el acceso a dicha información. Este mecanismo es utilizado por muchas de las aplicaciones estándarde un dispositivo, como por ejemplo la lista de contactos, la aplicación de SMS, o el calendario/agenda. Esto quiere decir que podríamos acceder a los datos gestionados por estas aplicaciones desde nuestras propias aplicaciones Android

[32] *Kotipalli& Imran, 2016.Hacking Android.Página 116.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*

haciendo uso de los *contentproviders* correspondientes.[33] Algunos sistemas operativos móviles como **Ios**, no tienen disponible añadir proveedores de contenidos por razones de seguridad para el usuario, por el contrario, el sistema operativo Android si los implementa.

Copiar / Pegar el almacenamiento en memoria cache del buffer[34]**,** es uno de los problemas más comunes en una aplicación. Si un usuario copia información sensible como el número de tarjeta de crédito, un atacante podría leer esta información con una pequeña pieza de código.

Archivo de registro o Loggin[35], es un archivo de texto en el que constan cronológicamente los acontecimientos que han ido afectando a un sistema informático (programa, aplicación, servidor, etc.), así como el conjunto de cambios que estos han generado. Este archivo podría ser accedido por un atacante si la aplicación no tiene los mecanismos de seguridad implementados.

Almacenamiento en caché de URL la caché es utilizada para guardar datos y acelerar ciertos procesos o peticiones, con el objetivo de hacer aplicaciones más eficientes. Podría tener accesoun atacante con una pequeña pieza de código para capturar datos en caché.

Objetos cookies del navegador definidos en M2.

Datos analíticos enviados a terceros son aquellos datos que se envían a otro tipo de aplicaciones en otros servidores, para informar sobre el estado actual de la aplicación y su desempeño. Si este tercero tiene vulnerabilidades de accesos, podría acceder a dichos datos desde un tercero.

- **M5: Poor Authorization and Authentication – Autenticación / Autorización pobre.**

[33]Informe en página especializada Sgoliver - http://www.sgoliver.net/blog/content-providers-en-android-i-construccion/
[34]Informe especializado en InfosecInstitute
- ✓ http://resources.infosecinstitute.com/android-hacking-security-part-4-exploiting-unintended-data-leakage-side-channel-data-leakage/#gref

[35]Informe especializado en AnakamaSupport - https://support.ankama.com/hc/es/articles/203790076--Qu%C3%A9-es-un-log-

Las aplicacionesmóviles tienen factores de usabilidad diferentes que las que podríamos encontrar en los sistemas web y computadoras tradicionales. A menudo para acceder se necesita de un código de *PIN* de 4 dígitos y contraseñas cortas debido al factor de entrada del dispositivo móvil. Los requerimientos de autenticación para las aplicaciones móviles pueden ser muy diferentes de la forma tradicional de autenticación web. Esto sería muy fácil para un atacante realizar un ataquede fuerza bruta a estos códigos de *PINs* o contraseñas cortas en la aplicación si no hay un control para prevenir estos tipos de ataques. Podemos probar en los esquemas de autorización pobres para intentar acceder a funciones con más privilegios en la aplicación, para enviar peticiones maliciosas al servidor y observando si estas peticiones son almacenadas. [36]

Con este tipo de vulnerabilidad un atacante podría:

- ✓ Realizar pedidos a la api sin autenticar
- ✓ Realizar pedidos como un usuario con las cookies o claves de sesión de otro.
- ✓ Usar IMEI para la autenticación.
- ✓ Identificar autenticación persistente: guardar la contraseña del usuario en el dispositivo de manera permanente.

Realizar pedidos a la API sin autenticar hace referencia a encontrar rutas para hacer peticiones al servidor, en donde no tienen un control de autenticación.

Realizar pedidos como un usuario con las cookies o claves desesión de otro si en el dispositivo móvil se han logrado acceder a estos datos, podría hacer peticiones al servidor pasando el control de autenticación haciéndose pasar por otro usuario.

Usar email para la autenticación, existen rutas donde reciben como parámetro el código email para identificar al dispositivo al que pertenece y de esta manera realizaralguna acción. Si se logra acceder a códigos de email de otros dispositivos podría hacer peticiones haciéndose pasar por otro usuario.

[36] *Kotipalli& Imran, 2016.Hacking Android.Página 116.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*

Identificar autenticación persistente, si el *token,* para pasar el control de autenticación no es refrescado cada cierto periodo de tiempo, podría indicar una vulnerabilidad ya que este perduraráen el tiempo. Lo mismo ocurre si se guardan contraseñas o cookies de sesión permanentemente en la aplicación móvil, pueden ser accedidas a las mimas.

- **M6: BrokenCryptography – Mal uso de criptografía**

Los ataques de criptografía rotos entran en escena cuando un desarrollador quiere aprovechar el cifrado para su aplicación. Hay dos principales razones y son las siguientes:

- ✓ Usar algoritmos débiles de encriptación y desencriptación.
- ✓ Usar encriptación externo o creado por el mismo desarrollador pero implementando un método inseguro.

Usar algoritmos débiles de encriptación y desencriptación esto incluye el uso de algoritmos con debilidades importantes o de lo contrario, insuficiente para requisitos de seguridad modernos como DES, 3DES.

Usar encriptación externo o creado por el mismo desarrollador pero implementando un método inseguroesto incluye almacenar las llaves en los archivos de base de datos local, "harcodear" las llaves, y así sucesivamente. [37]

Otros algoritmos débiles de encriptación:

- ✓ MD4, MD5.
- ✓ SHA1.
- ✓ RC2.

- **M7: Client-SideInjection – Inyección del lado del cliente**

[37] *Kotipalli& Imran, 2016.Hacking Android.Página 117.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*

Inyección del lado del cliente resulta la ejecución de código malicioso en el dispositivo del cliente a través de la aplicación móvil. Normalmente este código malicioso se proporciona en forma de datos que el atacante ingresa de diferentes maneras.

La mejor manera de averiguar si las aplicaciones son vulnerables a la inyección es identificar las fuentes de entrada y validar que los datos de usuario / aplicación suministrada está sujeto a la validación de entrada, no permitir la inyección de código.

Los siguientes son algunos ejemplos de Inyecciones del lado del cliente en aplicaciones móviles:

- ✓ Inyeccion en la vista web (*WebView*)
- ✓ Inyecciones *SQL* usado con bases de datos *SQLite*.
- ✓ SQL inyección en los proveedores de contenidos.
- ✓ Recorrido de rutas en proveedor de contenidos. [38]

Inyección en la vista web[39], una de las técnicas más conocidas *XSS*es básicamente una técnica de inyección de código en un sitio mediante el uso de técnicas que permiten insertar específicamente código (*Javascript* o *VBscript*) en los URLs o en los campos de un formulario que no ha sido validado para tal efecto para que ejecute en el contexto de otro sitio.

Inyecciones tradicionales *SQL* usado con bases de datos *SQLite*, dado que SQLite permite hacer consultas*SQL* sobre la base de datos, podría inyectar código que permita modificar la consulta que realmente se está intentando ejecutar. Alterando así los resultados en la misma.

Los proveedores de contenidos[40]**:** como su propio nombre indica, es el encargado de proveer datos de una aplicación, gestionando el acceso a los mismos. Estos datos pueden estar almacenados en el sistema de ficheros, en una base de datos *SQLite*, en la Web o en

[38] *Kotipalli& Imran, 2016.Hacking Android.Página 117.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*
[39] Mauro Maulini R.; PublicaciónenTecnologías Web - http://tecnologiasweb.blogspot.com.ar/2008/10/cross-site-scripting-xss-otro-enemigo.html
[40] David González Verdugo; Informe especializado en Solid Gear
- ✓ https://solidgeargroup.com/inyeccion-sql-en-proveedores-de-contenido-de-android-y-como-protegerse?lang=es

cualquier otra ubicación de almacenamiento persistente accesible desde nuestra aplicación. Si está almacenado en una base de datos *SQLite*, podría inyectar código *SQL*.

Recorrido de rutas en proveedor de contenidos[41]: pueden ser vulnerables si no validan bien sus parámetros de entrada. Un atacante mediante una aplicación maliciosa puede proporcionar una dirección de *Url* para engañar a la aplicación y devolver un archivo fuera del directorio, lo que permitirá que un atacante acceda a cualquier archivo de la aplicación.

- **M8: Security DecisionsviaUntrusted Inputs – Decisiones de seguridad a través de entradas no confiables.**

Los desarrolladores deberían siempre asumir que las entradas pueden ser partes no autorizadas en donde un atacante abuza funcionalidades sensibles de una aplicación. Específicamente en Android, un atacante puede interceptar las llamadas a los webservices y observar los parámetros sensibles. Las implementaciones defectuosasde estas funcionalidades en una aplicación, podrían elevar los permisos para un atacante. [42]

La aplicación puede aceptar datos de diferentes tipos de fuentes, estas decisiones permiten saltear restricciones o modelos de seguridad. Si hay un requisito de negocio de comunicación IPC, la aplicación móvil debe restringir el acceso a una lista blanca de aplicaciones de confianza.

- **M9: *ImproperSessionHandling* – Manejo inapropiado de sesiones.**

Las aplicaciones móviles usan protocolos como *SOAP* o *REST* para conectar servicios. Estos protocolos son sin estados. Cuando una aplicación móvil cliente está usando estos protocolos, los clientes obtienen un token del servidor después de una autenticación. El *token* generado por el servidor, será usado por toda la sesión del usuario. El manejo inapropiado de sesiones de *OWASP*, habla del ataque y seguridad de estas

[41] Publicación de Google - https://support.google.com/faqs/answer/7496913?hl=en
[42] *Kotipalli& Imran, 2016.Hacking Android.Página 117.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*

sesiones. Un problema común que es frecuentemente ver en una aplicación móvil, es invalidar el *token* para usarlo en la aplicación móvil pero no en el servidor. Generalmente, el token recibido por una aplicación móvil podría ser alojada en un archivo del sistema del cliente usando archivos de preferencia (clave – valor) o bases de datos *SQLite*. Un usuario malicioso que obtiene acceso a este token, puede ser usado todo el tiempo si el token no es invalidado en algún momento por el servidor. Otro posible escenario son los *timeouts*, malas implementaciones de *token* y un *token* que ha expirado. [43]

- **M10:*Lack of Binary Protections* – ProtecciondelBinario.**

Ingeniería inversa es uno de los problemas más comunes que se pueden ver en la mayoría de las aplicaciones móviles (Android), Uno de los primeros pasos de un atacante cuando obtiene una aplicación binaria es descompilar o desamblar la aplicación. Esto permite observar secretos "hardcodeados", encontrar vulnerabilidades, e incluso modificar las funcionalidades reemplazando la aplicación desmontada. Aunque ofuscando el código de la aplicación no es difícil de realizar, la mayoría parece no hacerlo. Cuando el código de una aplicación no está ofuscado todo lo que un atacante necesita es una herramienta que me permita hacer dicho proceso. [44]

La ofuscación se refiere a encubrir el significado de una comunicación haciéndola más confusa y complicada de interpretar.

Estructuras Arquitectónicas

El presente proyecto tiene como objetivo abocarse a la seguridad en aplicaciones móviles nativas del sistema operativo Android. Por este motivo, es necesario definir la estructura del sistema operativo y de una aplicación móvil desarrollada para dicho sistema operativo.

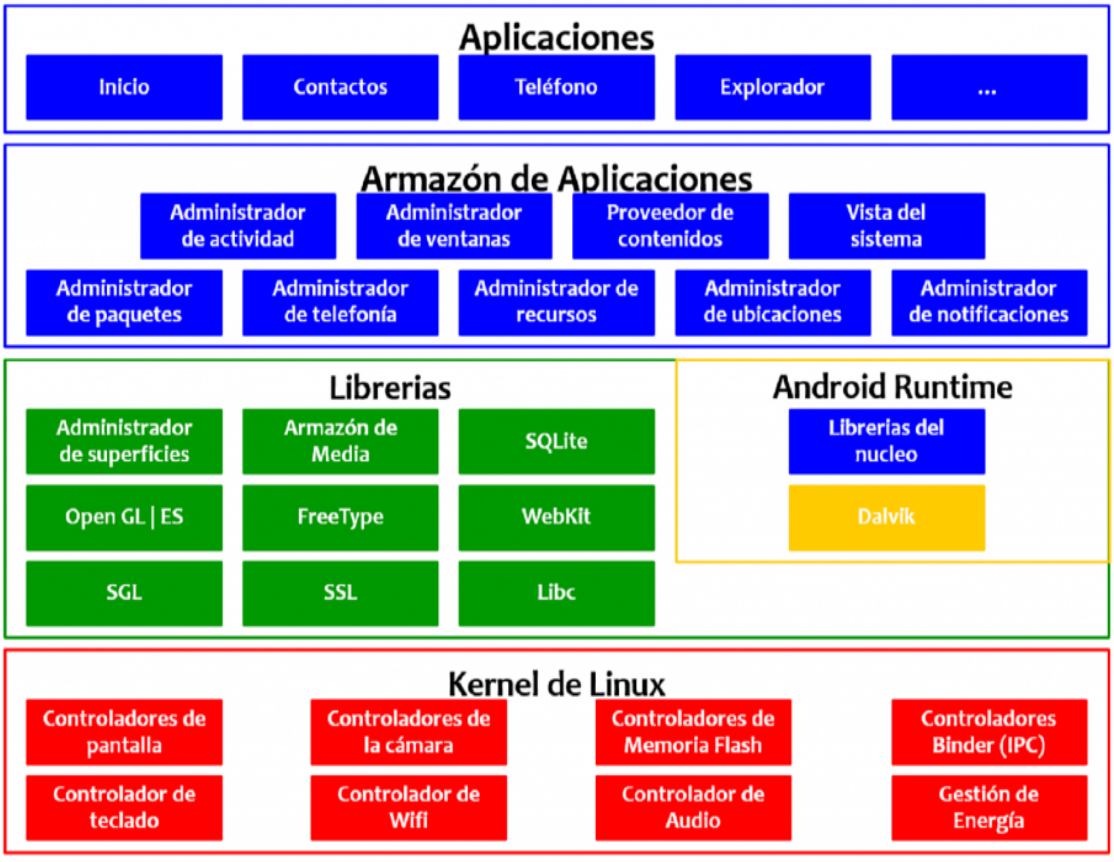
Estructura del sistema operativo Android

[43] *Kotipalli& Imran, 2016.Hacking Android.Página 118.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*
[44] *Kotipalli& Imran, 2016.Hacking Android.Página 118.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*

Android[45] es un sistema operativo basado en el núcleo de *Linux* con una plataforma abierta para dispositivos móviles adquirido por Google y la Open HandsetAllience, su finalidad es satisfacer la necesidad de los operadores móviles y fabricantes de dispositivos, además de fomentar el desarrollo de aplicaciones, cualidad que ningún otro sistema operativo incluye en sus conceptos (Google, 2010).

La estructura del sistema operativo ***Android***[46] se puede observar en el siguiente cuadro:

CUADRO Nº 2: Estructura del Sistema Operativo Android



Fuente: Universidad Carlos III de Madrid; Arquitectura Android - https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

[45]KristelMalave Polanco, José Luis BeauperthuyTaibo; Artículo científico especializado de NEGOTIUM
- ✓ http://ojs.revistanegotium.org.ve/index.php/negotium/article/view/248/235

[46]*Informe especializado en Universidad Carlos III de Madrid; Arquitectura Android*
- ✓ *https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android*

- **Aplicaciones**: Este nivel contiene, tanto las incluidas por defecto de *Android* como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las *API* y librerías de los niveles anteriores.
- ***Framework* de Aplicaciones**: Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de *API* y el mismo "*framework*", representado por este nivel.

Entre las **API** más importantes ubicadas aquí, se pueden encontrar las siguientes:

- ✓ ***ActivityManager*:** Conjunto de *API* que gestiona el ciclo de vida de las aplicaciones en *Android*.
- ✓ ***WindowManager*:** Gestiona las ventanas de las aplicaciones y utiliza la librería *SurfaceManager*.
- ✓ ***TelephoneManager*:** Incluye todas las *API* vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
- ✓ ***ContentProvider***: Permite a cualquier aplicación compartir sus datos con las demás aplicaciones de *Android*. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.
- ✓ ***ViewSystem*:** Proporciona un gran número de elementos para poder construir interfaces de usuario (*GUI*), como listas, mosaicos, botones, "check-boxes", tamaño de ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.

- ✓ ***LocationManager*:** Posibilita a las aplicaciones la obtención de información de localización y posicionamiento.

- ✓ ***NotificationManager*:** Mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una llamada entrante, un mensaje recibido, conexión *Wi-Fi* disponible, ubicación en un punto determinado, etc. Si llevan asociada alguna acción, en *Android* denominada *Intent*, (por ejemplo, atender una llamada recibida) ésta se activa mediante un simple clic.

- ✓ **XMPP *Service*:** Colección de *API* para utilizar este protocolo de intercambio de mensajes basado en *XML*.
  - **Librerías**: La siguiente capa se corresponde con las librerías utilizadas por Android. Éstas han sido escritas utilizando *C/C++* y proporcionan a *Android* la mayor parte de sus capacidades más características. Junto al núcleo basado en *Linux*, estas librerías constituyen el corazón de Android.

Entre las librerías más importantes ubicadas aquí, se pueden encontrar las siguientes:

- ✓ **Librería libc:** Incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.

- ✓ **Librería Surface Manager:** Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

- ✓ **OpenGL/SL y SGL:** Representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la

capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.

- ✓ **Librería Media Libraries:** Proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)

- ✓ **FreeType:** Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

- ✓ **Librería SSL:** Posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.

- ✓ **Librería SQLite:** Creación y gestión de bases de datos relacionales.

- ✓ **Librería WebKit:** Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

- **Tiempo de ejecución de Android:** Al mismo nivel que las librerias de Android se sitúa el entorno de ejecución. Éste lo constituyen las *CoreLibraries*, que son librerias con mulititud de clases Java y la máquina virtual *Dalvik*.

- **Núcleo Linux:** *Android* utiliza el núcleo de *Linux* 2.6 como una capa de abstracción para el *hardware* disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente *hardware* pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de *hardware*, lo primero que se debe realizar para que pueda ser utilizado desde *Android* es crear las librerias de control o drivers necesarios dentro de este *kernel* de *Linux* embebido en el propio Android.

Aspectos fundamentales de una aplicación Android[47]

Las aplicaciones de *Android* se escriben en lenguaje de programación *Java*. Las herramientas de *AndroidSDK* compilan el código, junto con los archivos de recursos y datos, en un *APK*: un paquete de Android, que es un archivo de almacenamiento con el sufijo .apk. Un archivo de *APK* incluye todos los contenidos de una aplicación de Android y es el archivo que usan los dispositivos con tecnología *Android* para instalar la aplicación.

Una vez instalada en el dispositivo, cada aplicación de Android se aloja en su propia zona de pruebas de seguridad:

- ✓ El sistema operativo Android es un sistema Linux multiusuario en el que cada aplicación es un usuario diferente.
- ✓ De forma predeterminada, el sistema le asigna a cada aplicación una ID de usuario de Linux única (solo el sistema utiliza la ID y la aplicación la desconoce). El sistema establece permisos para todos los archivos en una aplicación de modo que solo el ID de usuario asignado a esa aplicación pueda acceder a ellos.
- ✓ Cada proceso tiene su propio equipo virtual (EV), por lo que el código de una aplicación se ejecuta de forma independiente de otras aplicaciones.
- ✓ De forma predeterminada, cada aplicación ejecuta su proceso de Linux propio. Android inicia el proceso cuando se requiere la ejecución de alguno de los componentes de la aplicación, luego lo cierra cuando el proceso ya no es necesario o cuando el sistema debe recuperar memoria para otras aplicaciones.

De esta manera, el sistema *Android* implementa el principio de mínimo privilegio. Es decir, de forma predeterminada, cada aplicación tiene acceso solo a los componentes que necesita para llevar a cabo su trabajo y nada más. Esto crea un entorno

[47] Publicación de Google en Developer Android - https://developer.android.com/guide/components/fundamentals?hl=es

muy seguro en el que una aplicación no puede acceder a partes del sistema para las que no tiene permiso.

Sin embargo, hay maneras en las que una aplicación puede compartir datos con otras aplicaciones y en las que una aplicación puede acceder a servicios del sistema:

- ✓ Es posible disponer que dos aplicaciones compartan la misma ID de usuario de *Linux* para que puedan acceder a los archivos de la otra. Para conservar recursos del sistema, las aplicaciones con la misma ID de usuario también pueden disponer la ejecución en el mismo proceso de Linux y compartir el mismo EV (las aplicaciones también deben estar firmadas con el mismo certificado).
- ✓ Una aplicación puede solicitar permiso para acceder a datos del dispositivo como los contactos de un usuario, los mensajes de texto, el dispositivo de almacenamiento (tarjeta SD), la cámara, *Bluetooth* y más. El usuario debe garantizar de manera explícita estos permisos. Para obtener más información, consulta Cómo trabajar con permisos del sistema.

Componentes de una aplicación

Los componentes de la aplicación son bloques de creación esenciales de una aplicación para *Android*. Cada componente es un punto diferente a través del cual el sistema puede ingresar a tu aplicación. No todos los componentes son puntos de entrada reales para el usuario y algunos son dependientes entre sí, pero cada uno existe como entidad individual y cumple un rol específico; cada uno es un bloque de creación único que ayuda a definir el comportamiento general de tu aplicación.

Hay cuatro tipos diferentes de componentes de una aplicación. Cada tipo tiene un fin específico y un ciclo de vida diferente que define cómo se crea y se destruye el componente.

- ➢ **Activity – Actividades**

Una actividad representa una pantalla con interfaz de usuario. Por ejemplo, una aplicación de correo electrónico tiene una actividad que muestra una lista de los correos electrónicos nuevos, otra actividad para redactar el correo electrónico y otra actividad para leer correos electrónicos. Si bien las actividades trabajan juntas para proporcionar una experiencia de usuario consistente en la aplicación de correo electrónico, cada una es independiente de las demás. De esta manera, una aplicación diferente puede inicial cualquiera de estas actividades (si la aplicación de correo electrónico lo permite). Por ejemplo, una aplicación de cámara puede iniciar la actividad en la aplicación de correo electrónico que redacta el nuevo mensaje para que el usuario comparta una imagen.

- **Services– Servicios**

Un servicio es un componente que se ejecuta en segundo plano para realizar operaciones prolongadas o tareas para procesos remotos. Un servicio no proporciona una interfaz de usuario. Por ejemplo, un servicio podría reproducir música en segundo plano mientras el usuario se encuentra en otra aplicación, o podría capturar datos en la red sin bloquear la interacción del usuario con una actividad. Otro componente, como una actividad, puede iniciar el servicio y permitir que se ejecute o enlazarse a él para interactuar.

Content Providers – Proveedores de contenido

Un proveedor de contenido administra un conjunto compartido de datos de la app. Puedes almacenar los datos en el sistema de archivos, en una base de datos *SQLite*, en la Web o en cualquier otra ubicación de almacenamiento persistente a la que tu aplicación pueda acceder. A través del proveedor de contenido, otras aplicaciones pueden consultar o incluso modificar los datos (si el proveedor de contenido lo permite). Por ejemplo, el sistema *Android* proporciona un proveedor de contenido que administra la información de contacto del usuario. De esta manera, cualquier app con los permisos correspondientes puede consultar parte del proveedor de contenido (como *ContactsContract.Data*) para la lectura y escritura de información sobre una persona específica.

Los proveedores de contenido también son útiles para leer y escribir datos privados de tu aplicación y que no se comparten. Por ejemplo, la aplicación de ejemplo Bloc de notas usa un proveedor de contenido para guardar notas.

- **BroadcastReceiver – Receptores de mensajes**

Un receptor de mensajes es un componente que responde a los anuncios de mensajes en todo el sistema. Muchos mensajes son originados por el sistema; por ejemplo, un mensaje que anuncie que se apagó la pantalla, que la batería tiene poca carga o que se tomó una foto. Las aplicaciones también pueden iniciar mensajes; por ejemplo, para permitir que otras aplicaciones sepan que se descargaron datos al dispositivo y están disponibles para usarlos. Si bien los receptores de mensajes no exhiben una interfaz de usuario, pueden crear una notificación de la barra de estado para alertar al usuario cuando se produzca un evento de mensaje. Aunque, comúnmente, un receptor de mensajes es simplemente una "puerta de enlace" a otros componentes y está destinado a realizar una cantidad mínima de trabajo. Por ejemplo, podría iniciar un servicio para que realice algunas tareas en función del evento.

- **Activación de componentes mediante Intent**

Tres de los cuatro tipos de componentes (actividades, servicios y receptores de mensajes) se activan mediante un mensaje asincrónico llamado ***intent***. Las *intents* enlazan componentes individuales en tiempo de ejecución (son como mensajeros que solicitan una acción de otros componentes), ya sea que el componente le pertenezca a tu aplicación o a otra.

Una *intent* define un mensaje para activar un componente específico o un tipo específico de componente; una *intent* puede ser explícita o implícita, respectivamente.

Para actividades y servicios, una *intent* define la acción a realizar (por ejemplo, "ver" o "enviar" algo) y puede especificar el URI de los datos en los que debe actuar (entre otras cosas que el componente que se está iniciando podría necesitar saber). Por ejemplo, una *intent* podría transmitir una solicitud para que una actividad muestre una imagen o abra una página web. En algunos casos, puedes iniciar una actividad para recibir un resultado; en cuyo caso, la actividad también devuelve el resultado en una *Intent* (por

ejemplo, puedes emitir una *intent* para que el usuario elija un contacto personal y te lo devuelva; la *intent* de devolución incluye un URI que apunta al contacto seleccionado).

Para los receptores de mensajes, la intent simplemente define el anuncio que se está transmitiendo (por ejemplo, un mensaje para indicar que la batería del dispositivo tiene poca carga incluye solo una string de acción conocida que indica "batería baja").

El archivo de manifiesto

Para que el sistema *Android* pueda iniciar un componente de la app, el sistema debe reconocer la existencia de ese componente leyendo el archivo *AndroidManifest.xml* de la app (el archivo de "manifiesto"). Tu aplicación debe declarar todos sus componentes en este archivo, que debe encontrarse en la raíz del directorio de proyectos de la aplicación.

El manifiesto puede hacer ciertas cosas además de declarar los componentes de la aplicación como:

- ✓ Identificar los permisos de usuario que requiere la aplicación, como acceso a Internet o acceso de lectura para los contactos del usuario.
- ✓ Declarar el nivel de *API* mínimo requerido por la aplicación en función de las *API* que usa la aplicación.
- ✓ Declarar características de hardware y software que la aplicación usa o exige, como una cámara, servicios de *bluetooth* o una pantalla multitáctil.
- ✓ Bibliotecas de la API a las que la aplicación necesita estar vinculada (además de las Android framework API), como la biblioteca Google Maps.

➢ **Declaración de componentes**

La tarea principal del manifiesto es informarle al sistema acerca de los componentes de la aplicación. Por ejemplo, un archivo de manifiesto puede declarar una actividad de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

En el elemento **<application>**, el atributo **android:icon** señala los recursos para un ícono que identifica la app.

En el elemento **<activity>**, el atributo **android:name** especifica el nombre de clase plenamente calificado de la subclase *Activity* y el atributo **android:label** especifican una *string* para usar como etiqueta de la actividad visible para el usuario.

Debes declarar todos los componentes de la aplicación de esta manera:

- ✓ Elementos **<activity>** para las actividades
- ✓ Elementos **<service>** para los servicios
- ✓ Elementos **<receiver>** para los receptores de mensajes
- ✓ Elementos **<provider>** para los proveedores de contenido

Las actividades, los servicios y los proveedores de contenido que incluyas en tu archivo de origen pero que no declares en el manifiesto no estarán visibles para el sistema y, por consiguiente, no se podrán ejecutar. No obstante, los receptores de mensajes pueden declararse en el manifiesto o crearse dinámicamente en forma de código (como objetos **BroadcastReceiver**) y registrarse en el sistema llamando a **RegisterReceiver()**.

2.C. Bibliografía que documentan el problema

- **Android TM Hacker'sHandbook**

Autores: Joshua J. Drake, Pau Oliva Fora, Zach Lanier, Collin Mulliner, Stephen A. Ridley, Georg Wicherski; Editorial: Wiley.

En el presente libro, se especifica la arquitectura de una aplicación móvil en *Android*, como es su ecosistema, diseño de arquitectura, encontrar vulnerabilidades y el abusode las mismas.

- **Application Security for the Android Platform**

  Autor: Jeff Six

  Editorial: O'reilly

El propósito del libro es inducir al desarrollar de aplicaciones móviles en android, acerca del conocimiento de seguridad en aplicaciones móviles, para que aprendan a construir aplicaciones seguras. El trabajo presenta, como es la arquitectura de una aplicación *Android*, permisos en las aplicaciones, seguridad en los componentes, proteger datos almacenados y seguridad al interactuar con el servidor.

- **Android Security Cookbook**

Autores: Keith Makan, Scott Alexander-Bown; Copyright 2013: PacktPublishing

Los autores abordan lasprácticas para desarrollar Aplicaciones móviles *Android*, mecanismos de seguridad, resolviendo vulnerabilidades comunes en aplicaciones móviles para esta plataforma. Abarcando: herramientas para el desarrollo de aplicaciones *Android*, inspección de firmas y certificados, herramientas para *Testing*, identificar vulnerabilidades, abusar de las vulnerabilidades, mecanismos de protección, ingeniería reversa, análisis de la red, encriptación, administración de políticas.

- **Hacking Android**

Autores: Srinivasa Rao Kotipalli, Mohammed A. Imran; Copyright 2013: PacktPublishing.

Estaobra tiene como objetivo analizar, identificar y solucionar, las vulnerabilidades que están presentes en aplicaciones móviles para el sistema operativo Android. Abarcando:

- ✓ Armado del laboratorio para el análisis de aplicaciones.
- ✓ Modo "*root*" en *android*. Qué significa.
- ✓ Fundamentos al construir aplicaciones móviles *Android* identificando los componentes, similitudes entre los mismos.
- ✓ Ataque de aplicaciones móviles identificando los tipos de aplicaciones móviles.
- ✓ **Guía para Testing de aplicaciones móviles *OWASP*** (primer trabajo descripto).
- ✓ Herramientas de automatización.
- ✓ Almacenamiento de datos seguros.
- ✓ Ataques al:
  - Servidor con su herramienta de automatización.
  - Cliente (aplicación) análisis estático (código).
  - Cliente haciendo análisis dinámico (aplicación en ejecución).
- ✓ *Malwareandroid*.
- ✓ Atacando el dispositivo.

3. DEFINICIÓN DEL PROBLEMA

3.A. Definición exacta del problema

En el ámbito de la seguridad de las aplicaciones móviles, se ha intentado mejorar los indicadores a lo largo de los años, poniendo atención a vulnerabilidades comunes que se presentan en diferentes tipos de App.En los últimos años se ha incrementado masivamente el número de desarrollo de las aplicaciones, sin enfocarse en la seguridad de las mismas, lo que ha generado consecuencias negativas en los propietarios de datos y accesos al sistema.

Se han desarrollado investigaciones que se han hecho a lo largo del tiempo, con políticas públicas destinadas a promover este cambio de paradigma al desarrollar las aplicaciones móviles. No obstante, los índices de vulnerabilidad en las aplicaciones actuales siguen en constante crecimiento. Se puede decir entonces que las empresas están enfocadas directamente al desarrollo de App vulnerables. Es por eso que resulta relevante visualizar profundamente esta práctica, analizarla y sentar las bases para que abra el camino a otras modalidades y territorios de atención.

Es preciso **relevar información** respecto a este tipo de problemática, ya que en la actualidad, si bien hay investigaciones realizadas respecto a esta práctica, no se están tomando las medidas necesarias debido al costo que implica, ya sea invertir en una empresa que se dedica a ello o la capacitación de los mismos desarrolladores.

Tal como se mencionó, el presente proyecto tiene como objetivo analizar aplicaciones destinadas al sistema operativo *Android*. Analizando las vulnerabilidades a través de un análisis estático de la aplicación. Por ende, se mencionaràn las vulnerabilidades más comunes.

- **Atacando actividades**

Las actividades exportadas son los problemas más comunes en las aplicaciones *android* que generalmente encontramos en los *pentesting.* **Una *activity* (o componente) que está exportada, puede ser llamada por una aplicación instalada en el**

mismo dispositivo. Imagine una situación en donde una aplicación tiene *activity* sensible exportada y el usuario tiene instalada una aplicación maliciosa que invoca esta *activity* en cualquier momento que conecta su cargador. Esto es posible cuando una aplicación tiene *activities*que no están protegidas con funcionalidades sensibles. [48]

Una *activity* puede ser iniciada por componentes de otras aplicaciónes, si el valor de "*exported*" es true, significa que si puede, caso contrario que no puede. Si el valor es *false,* significa que solo puede ser llamada únicamente por componentes de la misma aplicación o aplicaciones con el mismo *userID.* [49]

El valor por defecto depende de si la *activity*contiene *intentfilters*. La ausencia de algún filtro significa que la *activity*puede ser llamada únicamente especificando esta el nombre exacto de la clase. Esto quiere decir que el componente únicamente es usado de manera interna, ya que una aplicación externa no conoce el nombre de la clase. El valor por defecto en este caso es "*false*". Caso contrario, la presencia de uno de estos filtros implica que el componente está disponible para uso externo, aquí el valor por defecto es "*true*". [50]

Anteriormentese hizo referenciaque la declaración de todos los componentes se encuentra en el archivo *AndroidManifest.xml*. Este archivo refleja toda la información necesaria de una aplicación y de un componente. Analizando este archivo podemos observar si hay *activity*que están exportadas explícitamente. El siguiente código es un ejemplo de una declaración de una *activity*que está exportada explícitamente:

```
<activity
        android:label="@string/profile"
        android:name=".activities.ViewProfile"
        android:exported="true" />
```

[48] *Kotipalli& Imran, 2016.Hacking Android.Página 198.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*
[49] *Kotipalli& Imran, 2016.Hacking Android.Página 198.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*
[50] *Kotipalli& Imran, 2016.Hacking Android.Página 198.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*

Esta *activity* puede ser llamada por otra aplicación maliciosa que se está ejecutando en el mismo dispositivo.La solución a este problema, está en modificar el atributo "*android:exported"* a "*false*". Definiendo la *activity*de la siguiente manera:

```
<activity
    android:label="@string/profile"
    android:name=".activities.ViewProfile"
    android:exported="false" />
```

De esta manera, solo podrá ser llamada a través del nombre exacto de la clase y por ende por la misma aplicación o aplicación con el mismo ID, ya que internamente se conoce únicamente el nombre de dicha clase.

- ***Intent filters***

Un *intent filter* especifica qué tipo de *intent* puede inicializar un componente de una aplicación. Podemos añadir restricciones especiales para inicializar un componente de una aplicación. Se abre el componente para recibir *intents* de diferentes tipos, mientras que filtra aquellos que no son significativos para el componente. Un *intentfilter* no puede ser utilizado como un mecanismo de seguridad para proteger los componentes de la aplicación y siempre recuerda que el componente es exportado por defecto si se hace uso de un *intent filter*. [51]

El siguiente ejemplo muestra un código simple de una *activity*con un *intentfilter:*

```
<activity android:label="@string/apic_label" android:name="com.androidpentesting.PrivateActivity">
    <intent-filter>
        <action android:name="com.androidpentesting.action.LaunchPrivateActivity"/>
        <categoryandroid:name="android.intent.category.DEFAULT"/>
```

[51] *Kotipalli& Imran, 2016.Hacking Android.Página 204.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*

```
        </intent-filter>
</activity>
```

Como se puede ver en el ejemplo anterior, un elemento *action* esta declarado dentro del tag *<intent-filter>,* para pasar este filtro, la acción especificada en el *intent*debe ser utilizada para inicializar el componente debe coincidir con la acción declarada.

Esta misma lógica de componente exportado se aplica a los componentes restantes.

- **Atacando los servicios**

Comose ha mencionado anteriormente, un *service* es un componente que se ejecuta en segundo plano para realizar operaciones prolongadas o tareas para procesos remotos. Si el servicio está exportado, ya sea de manera explícita (colocando el atributo *"android:exported=true"*) o implícita a través de un *intentfilter*, este *service* podría ser llamado desde otra aplicación maliciosa especificando el componente de la aplicación.

```
<service android:name=".services.LocationService">
<intent-filter>
        <action android:name="org.owasp.goatdroid.fourgoats.services.LocationService" />
</intent-filter>
</service>
```

Una aplicación maliciosa podría inicializar este componente exportado.

- **Atacando *BroadcastReceiver***

Broadcast Receiver o receptores de mensajes son componentes muy usados por los desarrolladores de Android. Los desarrolladores pueden añadir funcionalidades increíbles

utilizando los receptores de mensajes. Estos también son propensos a ataques cuando están exportados, explícitamente o mediante un *intent-filter*. [52]

El siguiente ejemplo muestra la declaración de un recepto de mensaje que está siendo exportado mediante un *intent-filter*.

```
<receiver android:label="Send SMS"
android:name=".broadcastreceivers.SendSMSNowReceiver">
    <intent-filter>
        <action
        android:name="org.owasp.goatdroid.fourgoats.SOCIAL_SMS" />
    </intent-filter>>
</receiver>
```

- **Atacando proveedores de contenido.**

Similar a otros componentes de la aplicación que hemos discutido, los proveedores de contenido pueden ser abuzados cuando están exportados. Las aplicaciones con el target *SDK* versión menor a 17 son exportadas por defecto. Esto significa que debemos explicitar el atributo "*android:exported = false*" en el archivo *AndroidManifest.xml*, para que no sea exportado por defecto. Este comportamiento ha sido cambiado a partir del nivel de Api 17 (Android versión 4.2). Para ver en qué nivel de Api se encuentra una aplicación en el archivo AndroidManifest.xml podemos observar el atributo "*android:targetSdkVersion*". [53]

*android:targetSdkVersion es el v*alor entero que designa el nivel de API al cual se dirige la aplicación. Si no se configura, el valor predeterminado es igual al valor asignado a la *minSdkVersion*.[54]

[52] *Kotipalli& Imran, 2016.Hacking Android.Página 206.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*
[53] *Kotipalli& Imran, 2016.Hacking Android.Página 206.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*
[54] Publicación de Google en Developer Android - https://developer.android.com/guide/topics/manifest/uses-sdk-element?hl=ES#target

Este atributo informa al sistema que has realizado las pruebas en la versión prevista y el sistema no deberá habilitar ningún comportamiento de compatibilidad a fin de mantener la compatibilidad con versiones posteriores de tu aplicación y la versión prevista. La aplicación puede continuar ejecutándose en versiones más antiguas (anteriores a la *minSdkVersion*).

```
<provider android:name=".NoteProvider"
android:authorities="com.sonyericsson.notes.provider.Note" />
```

- **Aplicaciones depurables**

Las aplicaciones de *Android* tienen un atributo conocido como *android:debuggable* en el archivo *AndroidManifest.xml*. Esto se establece en true durante la etapa de desarrollo de la aplicacióny por defecto se establece en *false* una vez que la aplicación se exporta para su distribución. Esteatributo se usapara fines de depuración durante el proceso de desarrollo y no se supone queestablecerse en *true* en producción. Si un desarrollador establece explícitamente el valor de la depuracióna *true* se vuelve vulnerable. [55]

Una aplicación depurable es propensa a que un atacante con acceso físico al dispositivo pueda extraer información sensible administrada por la aplicación, ingresar al log de archivos de la aplicación y ejecutar código arbitrario en la misma.

3.B. Objetivos

El objetivo en **analizar, diseñar e implementar** un sistema que permita identificar vulnerabilidades, proporcionando medidas para lograr mayor consistencia e integridad de los datos.

[55] *Kotipalli& Imran, 2016.Hacking Android.Página 250.353 Páginas.Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.*

- ✓ Implementar un sistema que permita identificar vulnerabilidades en aplicaciones móviles.
- ✓ Desarrollo de aplicación de mecanismos de seguridad automatizados.
- ✓ Presentación de informe del análisis.
- ✓ Capacitación al desarrollador acerca de las vulnerabilidades encontradas y su solución.

3.C. Alcance

En el presente proyecto se pretende desarrollar un sistema que permita identificar las vulnerabilidades de **aplicaciones móviles para el sistema operativo *Android***, basándose en el **análisis estático** de la misma. Si bien es necesario hacer un análisis detallado acerca de las vulnerabilidades en una aplicación Android, el presente proyecto estará limitado a identificar vulnerabilidades correspondientes a comunicación entre procesos *IPC* (*Inter-Process-Communication*). Las mismas, se identifican analizando el archivo principal de la aplicación. Estas vulnerabilidades suelen provocar la fuga de datos confidenciales a otras aplicaciones instaladas en el mismo dispositivo, de las cuales se podrían identificar otro tipo de vulnerabilidades no pertenecientes a IPC si el desarrollador profundiza en el análisis de las vulnerabilidades encontradas. Está dirigido principalmente a desarrolladores de aplicaciones móviles nativas *Android.*

3.D. Recursos

Para el desarrollo del siguiente proyecto es necesario contar con los siguientes recursos

- ➢ Computadora *intelcore i5*. 8gb Ram. 64 bits.
- ➢ Sistema operativo (Preferentemente GNU/Linux).
- ➢ Lenguaje de programación de *scripting.*
- ➢ *Librería/Framework* para desarrollar entorno web.
- ➢ Código fuente de aplicación *android* nativa.

3.E. Alternativas tecnológicas.

- *Hardware*:
  - ✓ *Intel core i3. 8gb Ram.*
  - ✓ *Intel core i3. 4gb Ram.*

- Sistema operativo:
  - ✓ *GNU/Linux.*
  - ✓ *Windows.*
  - ✓ *Mac OS.*
  - ✓ *Arquitectura X86/x64.*

- Lenguaje de programación de scripting:
  - ✓ *Ruby.*
  - ✓ *Python.*
  - ✓ *Php.*
  - ✓ *JavaScript.*
  - ✓ *Flex.*
  - ✓ *Groovy.*

- Librería/Framework para desarrollar entorno web:
  - ✓ Angular.
  - ✓ React.
  - ✓ Vue.
  - ✓ .Net.

4. SOLUCIÓN PROPUESTA

4.A. Justificación de la solución aportada.

La motivación personal que impulsa el abordaje de éste tema, son los valores éticos y de responsabilidad profesionales con que considero se deben manejar los desarrolladores en el ámbito de la informática, especialmente teniendo en cuenta que, con las App, se expone a la sociedad en general como consecuencia del uso masivo de los dispositivos móviles. La problemática de la carencia de seguridad y los riesgos implícitos de las aplicaciones móviles vulnerables, es el objetivo de la propuesta de desarrollo del presente trabajo final de grado.

La importancia de resolver esta problemática, es la de conseguir que se desarrollen aplicaciones con un índice de vulnerabilidad menor.

Para resolver esta problemática, manifiesto gran interés en la posibilidad de desarrollar un sistema de *software*que permita identificar las vulnerabilidades en las aplicaciones móviles *android*, basandas en el análisis estático de la misma, proporcionando los mecanismos de seguridad automatizados para agilizar el proceso de seguridad y mostrar un informe de los resultados obtenidos, capacitando al programador en los aspectos de seguridad en las aplicaciones.

4.B. Conocimientos Teóricos Aportados

Para el desarrollo del software, se toman los conocimientos enunciados en las secciones anteriores del proyecto, teniendo presente los conceptos desarrollados en cuanto a:

- Top 10 vulnerabilidades "*Owasp mobile Project*".
- Tipos de aplicaciones móviles.
- Arquitectura del sistema operativo *android.*
- Arquitectura de una aplicación *android*.

- Vulnerabilidades en aplicación *android* (análisis estático).

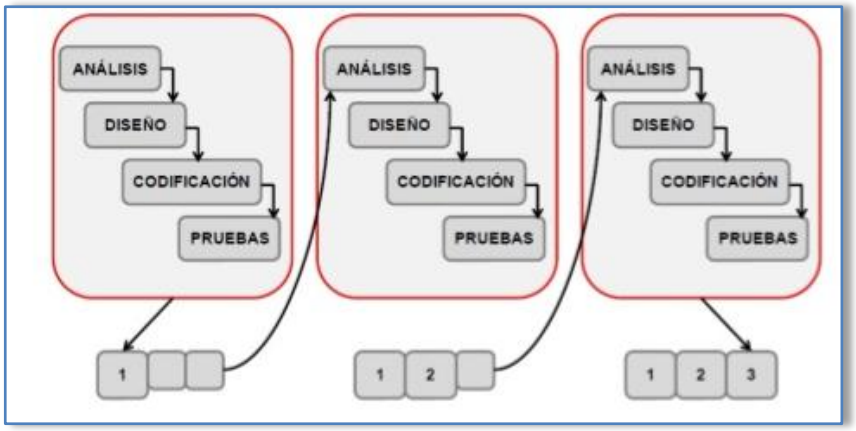
4.C. Metodología de la solución aportada

En primera instancia, se hará un análisis de riesgos basado en el estándar para la administración de proyectos PMBOK [56], el cual es desarrollado previamente al desarrollo de la solución, el cual contempla los siguientes pasos:

1. Indentificación.
2. Analisis cualitativos.
3. Análisis cuantitativo.
4. Planificacion de las respuestas.

De las diferentes metodologías de desarrollo de software estudiadas a lo largo de la carrera, se hace uso de la metodología "**Iterativo e incremental**". Esta es la metodología que más se adapta para el presente proyecto, permitiendo tener software funcional en cada iteración e incrementándolo por cada una de las mismas.[57]

CUADRO Nº 3: Desarrollo Iterativo e Incremental



Fuente: Slideshare.net – Desarrollo Iterativo e Incremental - https://image.slidesharecdn.com/desarrolloiterativoeincremental-120829050505-phpapp02/95/desarrollo-iterativo-e-incremental-6-728.jpg?cb=1346216786

[56] Project Management Institute, 2013. Guía de los Fundamentos para la Dirección de Proyectos (Guía de PMBOK). Quinta Edición. Página 309. 568 páginas. Project Management Institute, EE.UU. Inc. ISBN 978-1-62825-009-1.

[57] Slideshare.net – Desarrollo Iterativo e Incremental - https://image.slidesharecdn.com/desarrolloiterativoeincremental-120829050505-phpapp02/95/desarrollo-iterativo-e-incremental-6-728.jpg?cb=1346216786

Para la fase de análisis y diseño se hará uso de las técnicas de análisis y diseño estructurado moderno por *Edward Yourdon* [58], utilizando únicamente las herramientas necesarias para cada iteración.

Serán necesarias tres iteraciones para el desarrollo de software, constadas por:

1. **Identificación y solución de vulnerabilidades en la aplicación**: mediante un lenguaje de programación de *scripting*, con un análisis de expresiones regulares para identificar las vulnerabilidades de la aplicación *android*, exportando un archivo con las detecciones realizadas y dando la posibilidad al desarrollador de aplicar los mecanismos de seguridad necesarios.
2. **Informe detallado de las vulnerabilidades encontradas:** a través del entorno web se informará al desarrollador de las vulnerabilidades encontradas, como así también, de cuál es la solución que se ha proporcionado induciendo al desarrollador de aplicaciones móviles *android* en el ámbito de la seguridad.
3. **Integración de 1 y 2:** unificar los 2 entornos de manera que al terminar la identificación de vulnerabilidades (1), automáticamente inicie y muestre el informe del análisis (2).

Para la fase de codificación se hará uso de diferentes tecnologías que se adaptan para cada modulo correspondiente:

- **Lenguaje de programación de *scripting*: *Python*.**
- ***FrameworkWeb*: Angular.**

Se ha seleccionado por un lado *Python*, porque es un lenguaje de programación *opensource*, ampliamente utilizado en el ámbito de la seguridad informática, con un modulo en particular para hacer análisis de expresiones regulares.

Particularmente nunca he realizado un desarrollo con este lenguaje de programación. Lo que me facilitó aprender este lenguaje para dar solución al problema,

[58] *Edward Yourdon, 1993. Análisis Estructurado Moderno. 735 Páginas. Prentice Halls Hispanoamericana, S.A.,México. ISBN 0-13-598624-9.*

fueron los conocimientos aportados a lo largo de la carrera y a la agilidad adquirida de cambiar de tecnología para resolver problemas, según la situación que se presente.

Por otro lado se seleccionó el *Framework Angular (open source)*, para el entorno web, ya que tengo la experiencia de haberlo utilizando en otros desarrollos. Por otra parte, es uno de los más utilizados por desarrolladores y empresas para dar solución *web* a sus clientes.

Se hace uso de tecnologías *opensource*, porque a criterio personal, considero que en el ámbito académico se debería impartir conocimientos especialmente desde el enfoque del **software libre.** Cuando un estudiante hace uso de software libre para sus proyectos, amplia su visión y aprende con mayor calidad todo lo que se ejecuta por detrás del mismo, adquiriendode esta manera un conocimiento más especializado sobre la tecnología que está aprendiendo, dándole la posibilidad de analizar el entorno en el que trabaja, modificarlo cuando sea necesario, mejorarlo e inducirlo a que dicte su propio criterio sobre la tecnología.

Coincidiendo totalmente con *Richard Stallman*[59];quien manifiesta que las escuelas tienen una misión social, enseñar a los alumnos a ser ciudadanos de una sociedad fuerte, capaz, independiente, solidaria y libre. Deben promover el uso de software libre al igual que promueven la conservación y el voto.

El software libre permite a los alumnos aprender cómo funciona el software. Algunos alumnos son programadores natos, de adolescentes anhelan aprender absolutamente todo sobre los ordenadores y el software. Manifiestan una intensa curiosidad por leer el código fuente de los programas que usan a diario.

Las escuelas que utilicen software libre contribuirán al progreso de los alumnos más brillantes en programación. ¿De qué manera los programadores natos aprenden a convertirse en buenos programadores? Tienen que leer y comprender el código de programas reales que la gente de hecho usa. La manera de aprender a escribir código

[59]Richard Stallman; Artículo en GNU- https://www.gnu.org/education/edu-schools.es.html

bueno y claro es leyendo y escribiendo mucho código. "*Uno lee más código del que escribe*".

La razón más profunda para utilizar software libre en las escuelas es la educación moral. Esperamos que las escuelas enseñen hechos básicos y habilidades útiles, pero esa es solo una parte de su función. La tarea fundamental de las escuelas es enseñar a ser buenos ciudadanos, incluyendo el hábito de ayudar a los demás. En el ámbito informático, esto se traduce en enseñar a compartir el software. Las escuelas, a partir del jardín infantil, deberían decirle a sus alumnos: «Si traéis software a la escuela, debéis compartirlo con los demás niños. Y debéis mostrar el código fuente en clase, por si alguien quiere aprender. Por lo tanto, no está permitido traer a la escuela software que no sea libre, a menos que sirva para hacer algún trabajo de ingeniería inversa».

En otras palabras, darle la posibilidad al estudiante de modificar, ejecutar, copiar, distribuir, redistribuir, cambiar, mejorar, y lo más importante académicamente, **estudiar el software libremente.**

4.D. Planificación y desarrollo del Proyecto

Análisis de riesgos

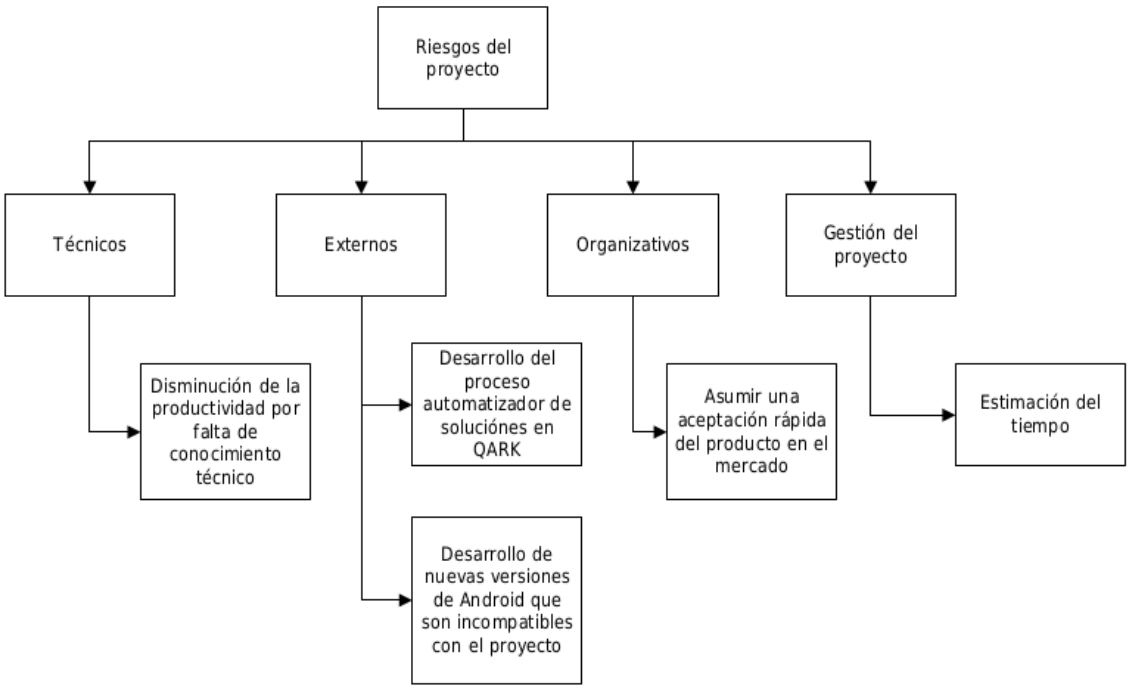
1. Identificar los riesgos:
    - Entradas:
        - Estimacion de la duración de actividades.
        - Estimacion de los costos de la actividad.
    - Herramientas y técnicas:
        - Análisis FODA.

FODA		
Análisis interno	**Fortalezas:** • Seguridad en los datos por trabajo en cloud.	**Debilidades:** • Escasos recursos humanos para afrontar el proyecto.

	• Buena organización por estrategias y objetivos bien definidos. • Proyección de consecución del desarrollo bien planificada en el tiempo. • Recusos Humanos con habilidades de innovación. • Escasa incidencia de tiempo en desarrollo.	• Escasa disponibilidad monetaria para la incorporación de mayor cantidad de Analistas Programadores. • Excesiva incidencia de tiempo destinado a la investigación.
Análisis externo	**Oportunidades:** • Estimación de buena y rápida aceptación del producto por desarrolladores, como herramienta que permite disminuir el índice de vulnerabilidad para aplicaciones *Android*. • Sociedad, en calidad de usuarios, expuesta a carencia de seguridad y riesgos implícitos en aplicaciones móviles.	**Amenazas:** • Existencia de un proyecto competidor, denominado "*QARK*". • Cambios en la estructura de aplicaciones móviles por nuevas actualizaciones lanzadas al mercado, por lo que nuevos desarrollos en ella podría ser incompatible con el proyecto.

- Salida:
  - Registro de los riesgos:
    - ✓ Qark podría desarrollar el módulo diferenciador (automatización de solución).
    - ✓ En caso de ausencia del Analista programador por casos particulares el proyecto se atrasaría.
    - ✓ Los proyectos desarrollados en las nuevas versiones de Android podría no ser compatible con el proyecto.
    - ✓ La productividad podría disminuirse en caso que el desarrollador no comprenda la tecnología.
    - ✓ Asumir una rápida aceptación del producto en el mercado.

2. Análisis cualitativo de los riesgos:
   - Entradas:
     - Registro de riesgos.
   - Herramientas y técnicas:
     - Categorización de riesgos.



- Salida:
  - Riesgos categorizados

3. Planificación de la respuesta:
   - Entradas:
     - Registro de riesgos.
   - Herramientas y técnicas:
     - Estrategias para riesgos negativos o amenazas.
       - ✓ Aumento del tiempo de desarrollo por falta de conocimiento técnico: **MITIGAR.** Existe disponible un tiempo de investigación destinado a la capacitación técnica del personal, que puede no ser suficiente pero reduce la probabilidad de retrasar el proyecto por falta del conocimiento técnico.
       - ✓ Desarrollo del proceso automatizador de soluciones en QARK: **ACEPTAR (Activo).** En este riesgo no disponemos de control sobre el desarrollo de QARK, aceptar el riesgo es la estrategia a seguir y posteriormente desarrollar un plan de marketing para competir con este software.
       - ✓ Incompatibilidad nuevas versiones de Android: **ACEPTAR (Activo).** En caso de ocurrencia, posteriormente a la implementación del proyecto se debería realizar un mantenimiento para soportar las aplicaciones en las nuevas versiones de Android.
       - ✓ Estimación del tiempo: **EVITAR.** Con el desarrollo del diagrama Gantt se puede visualizar la estimación del tiempo de todas las tareas puntuales del proyecto que un desarrollador debe seguir.
     - Estrategias para riesgos positivos u oportunidades:
       - ✓ Asumir un aceptación rápida del producto en el mercado: **Explotar.** Las empresas de desarrollo de aplicaciones

móviles Android son numerosas y amplias. Desarrollar este mercado con un análisis de marketing a definir.

- Salida:
  - Actualización a los documentos del proyecto.

Para el desarrollo del proyecto, represento los tiempos en un diagrama Gantt, dado que permite obtener una visión general del proyecto total en un gráfico.

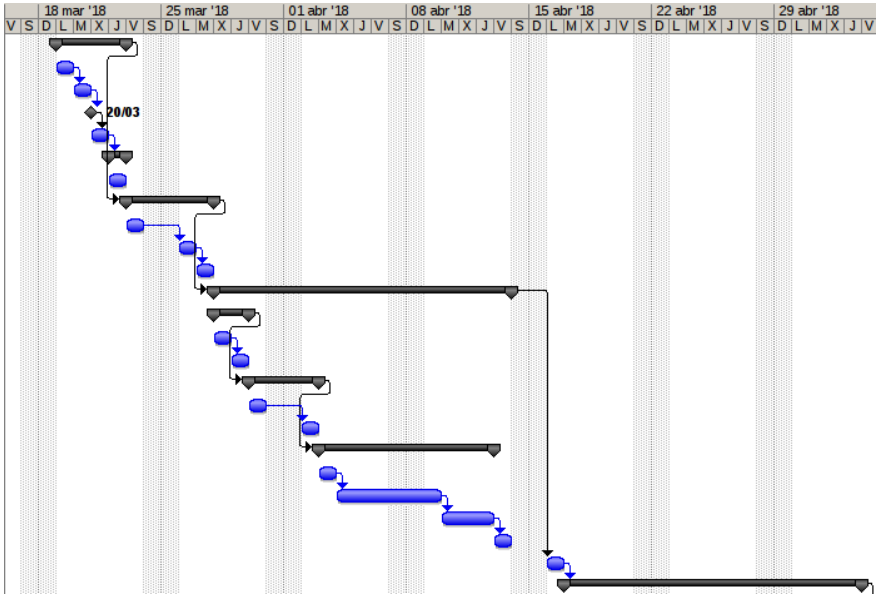
CUADRO Nº 4: Planificación del Proyecto

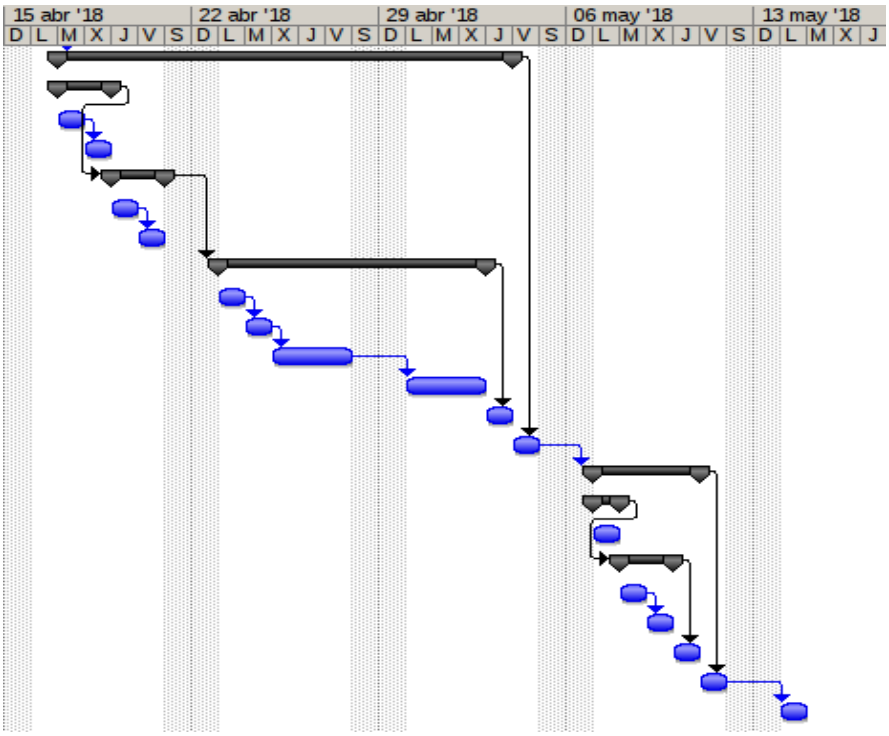
	Nombre de tarea	Duración	Comienzo	Fin	Pred
1	**Planificación de proyecto**	**4 días**	**lun 19/03/18**	**jue 22/03/18**	
2	Estudio de la problemática	1 día	lun 19/03/18	lun 19/03/18	
3	Establecimiento de tiempo	1 día	mar 20/03/18	mar 20/03/18	2
4	Establecimiento de costos	0 días	mar 20/03/18	mar 20/03/18	3
5	Análisis de riesgos	1 día	mié 21/03/18	mié 21/03/18	4
6	**Viabilidad del producto**	**1 día**	**jue 22/03/18**	**jue 22/03/18**	**5**
7	Cálculo VAN y TIR	1 día	jue 22/03/18	jue 22/03/18	
8	**Investigación de tecnologías**	**3 días**	**vie 23/03/18**	**mar 27/03/18**	**1**
9	Lenguaje de programación Scripting	1 día	vie 23/03/18	vie 23/03/18	
10	Framework web	1 día	lun 26/03/18	lun 26/03/18	9
11	Integración de ambos desarrollos	1 día	mar 27/03/18	mar 27/03/18	10
12	**Iteración 1 - Scripting**	**13 días**	**mié 28/03/18**	**vie 13/04/18**	**8**
13	**Análisis**	**2 días**	**mié 28/03/18**	**jue 29/03/18**	
14	Recopilación de información	1 día	mié 28/03/18	mié 28/03/18	
15	Diagramación: DFD - DD'S - EP	1 día	jue 29/03/18	jue 29/03/18	14
16	**Diseño**	**2 días**	**vie 30/03/18**	**lun 02/04/18**	**13**
17	Definición del modelo de diseño	1 día	vie 30/03/18	vie 30/03/18	
18	Diseño de interfaz	1 día	lun 02/04/18	lun 02/04/18	17
19	**Codificación**	**8 días**	**mar 03/04/18**	**jue 12/04/18**	**16**
20	Preparación ambiente de desarr	1 día	mar 03/04/18	mar 03/04/18	
21	Identificación de vulnerabilidades	4 días	mié 04/04/18	lun 09/04/18	20
22	Mecanismos de seguridad	3 días	mar 10/04/18	jue 12/04/18	21
23	Prueba	1 día	vie 13/04/18	vie 13/04/18	22
24	Implementación	1 día	lun 16/04/18	lun 16/04/18	12

25	**Iteración 2 - Web**	**13 días**	**mar 17/04/18**	**jue 03/05/18**	**24**
26	**Análisis**	**2 días**	**mar 17/04/18**	**mié 18/04/18**	
27	Recopilación de información	1 día	mar 17/04/18	mar 17/04/18	
28	Diagramación: DFD - DD'S - EP	1 día	mié 18/04/18	mié 18/04/18	27
29	**Diseño**	**2 días**	**jue 19/04/18**	**vie 20/04/18**	**26**
30	Definición del modelo de diseño	1 día	jue 19/04/18	jue 19/04/18	
31	Diseño de interfaz	1 día	vie 20/04/18	vie 20/04/18	30
32	**Codificación**	**8 días**	**lun 23/04/18**	**mié 02/05/18**	**29**
33	Preparación de ambiente de des	1 día	lun 23/04/18	lun 23/04/18	
34	Lectura de archivo de vulnerabil	1 día	mar 24/04/18	mar 24/04/18	33
35	Vistas de componentes	3 días	mié 25/04/18	vie 27/04/18	34
36	Integración de vulnerabilidades e	3 días	lun 30/04/18	mié 02/05/18	35
37	Prueba	1 día	jue 03/05/18	jue 03/05/18	32
38	Implementación	1 día	vie 04/05/18	vie 04/05/18	25
39	**Iteración 3 - Integración**	**4 días**	**lun 07/05/18**	**jue 10/05/18**	**38**
40	**Análisis y Diseño**	**1 día**	**lun 07/05/18**	**lun 07/05/18**	
41	Diagrama de estructuras	1 día	lun 07/05/18	lun 07/05/18	
42	**Codificación**	**2 días**	**mar 08/05/18**	**mié 09/05/18**	**40**
43	Preparación de ambiente de inte	1 día	mar 08/05/18	mar 08/05/18	
44	Integración	1 día	mié 09/05/18	mié 09/05/18	43
45	Prueba	1 día	jue 10/05/18	jue 10/05/18	42
46	Implementación	1 día	vie 11/05/18	vie 11/05/18	39
47	Demo final	1 día	lun 14/05/18	lun 14/05/18	46

Fuente: Elaboraciòn propia.

GRÁFICO Nº 1: Cronograma del Proyecto





Fuente: Elaboraciòn propia.

A continuación se detallará la fase de análisis y diseño de cada iteración, la codificación y prueba serán adjuntadas al proyecto.

Lista de eventos:

1. Desarrollador de aplicaciones *android* identifica vulnerabilidades.
2. Desarrollador de aplicaciones *android* aplica mecanismos de seguridad a las vulnerabilidades detectadas.
3. Desarrollador de aplicaciones *android* genera reporte del análisis.

Dado que se utilizará, adicionalmente, herramienta pertenecientes a Lenguaje Unificado de Modelado, a continuación se procede a escribir el listado de eventos mencionado anteriormente en listado de casos de uso.

Listado de casos de uso:

1. Identificar vulnerabilidades.
2. Aplicar mecanismos de seguridad.

3. Generar reporte del análisis.

De esta manera podremos hacer uso de herramientas de Análisis estructurado moderno combinado con UML para un mejor entendimiento del comportamiento del sistema.

➢ Iteración 1 – Scripting

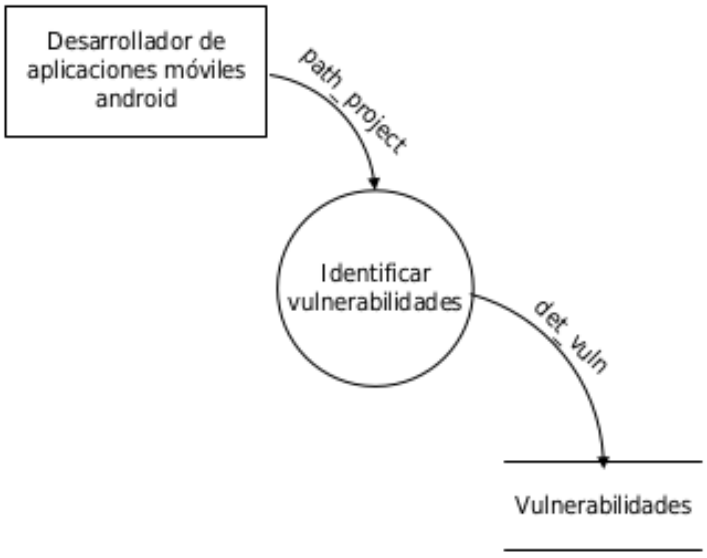
Análisis:

Herramientas a utilizar:

- ✓ **DFD:** Diagrama de flujo de datos. Especifica los procesos o funciones.
- ✓ **Especificación de caso de uso:** se combinará con esta herramienta perteneciente a Lenguaje Unificado de Modelado para una mejor comprensión del comportamiento del sistema.
- ✓ **DD'S:** Diccionario de datos. Significado y composición de los datos.
- ✓ **EP:** Especificación de procesos. Acciones que suceden en cada proceso.

➢ **Evento**: Desarrollador de aplicaciones *android* identifica vulnerabilidades.

- ✓ **DFD:**



✓ Especificación de caso de uso:

Caso de uso: Identificar vulnerabilidades
Actor: Desarrollador de aplicaciones móviles
Pre condiciones: Código y Path de proyecto Android.
Post condiciones: Vulnerabilidades detectadas

Escenario principal de éxito

Actor	Sistema
1- Iniciar nuevo C.U. Iniciar análisis de vulnerabilidades. 3- Ingresar Path.	 2- Solicita Path proyecto Android. 4- Buscar AndroidManifest.xml. 5-Identificar componentes exportados explícitamente. 6- Identificar componentes exportados por defecto con intent filter. 7- Identificar vulnerabilidades provider. 8-Identificar si es aplicación debuggable 9- Escribir json File Salida. 10- Fin del C.U.

Escenarios alternativos
4- Validar búsqueda AndroidManifest.xml

Actor	Sistema
	4.1- Mostrar mensaje: "Android Manifest.xml no encontrado"

✓ DD'S:

- path_project = [Path_Proyecto_Android | Path_Android_Manifest.xml]
- det_vuln = { id + type + { (action) **Especifica el action del intent que exporta el componente por defecto ** + (type_exported) + name_component } }

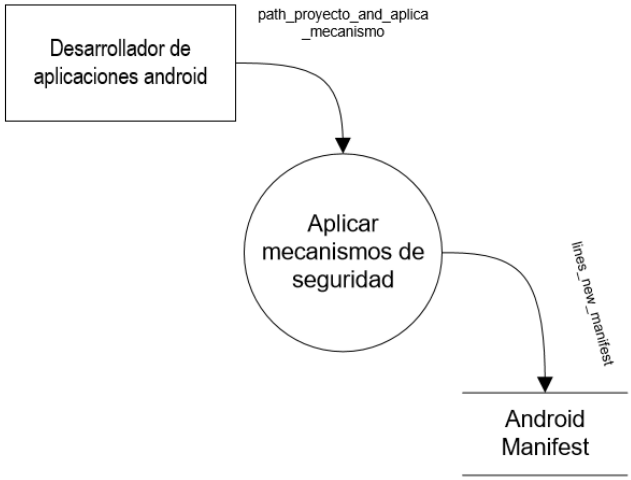
✓ EP:

```
path = SolicitarPathProyecto()

if (not path)
```

```
exit()


path = BuscarPathAndroidManifest(path)
with open(path) as f:
for line in f.readlines():



#Obtener el tipo de componente
component = getComponent(line);


    # Chequeamos los componentes que estan exportados manualmente con android:exported = true
checkExported(line);


    # Ver si esta exportado por defecto con un intentfilter adentro (Si tiene android:exported = false no estara exportado aunque tenga un intentfilter)
checkIntent(line);


    # Analizar si los providerestan exportados por defecto
if (isProvider(line))
analizeProvider(line);


# Analizar si es una aplicaciondebuggable
isDebuggable(line);


# Escribimos el json file de salida
writeOutPut()
exit()
```

- **Evento:** Desarrollador de aplicaciones *android* aplica mecanismos de seguridad a las vulnerabilidades detectadas.

✓ **DFD:**



✓ **Especificación de caso de uso:**

Caso de uso: Aplicar mecanismos de seguridad.
Actor: Desarrollador de aplicaciones móviles.
Pre condiciones: Código y Path de proyecto Android.
Post condiciones: Vulnerabilidades corregidas.

Escenario principal de éxito

Actor	Sistema
1- Iniciar nuevo C.U. Iniciar aplicar mecanismos de seguridad. 3- Ingresar Yes.	2- Solicita aplicar mecanismos de seguridad. 4- Validar solicitud. 5-Identificar tipo de componente. 6-Identificar componentes exportados por defecto con intent filter. 7-Cambiar componentes exportados explícitamente a no exportado. 7-Identificar componentes exportados implícitamente con intent filter 8- Establecer explicitamente componente no exportado. 9- Identificar provider exportados 10-Cambiar componente provider a no

	exportado. 11-Identificar si es aplicación debuggable 12-Cambiar a aplicación no debuggable. 10- Fin del C.U.

Escenarios alternativos
4- Validar Solicitud aplicar mecanismos de seguridad (Entrada distinta a y/n)

Actor	Sistema
	4.1- Mostrar mensaje: "La entrada ingresada no es correcta"

Escenarios alternativos
4- Validar Solicitud aplicar mecanismos de seguridad (Entrada = n)

Actor	Sistema
	4.1- Fin Caso de uso.

- ✓ **DD'S:**

path_proyecto_and_aplica_mecanismo = path_proyecto + [s | n] **aplicar mecanismos de seguridad**

lines_new_manifest = { linea }

- ✓ **EP:**

```
mecanismo = raw_input();
if (mecanismo == n)
exit()

with open(path) as f:
for line in f.readlines():

#Obtener el tipo de componente
component = getComponent()

# Chequeamos los componentes que estan exportados manualmente con android:exported = true. Si es así, cambiarlo a false.
if (checkExported(line)):
modificarExportedAFalse(line)
```

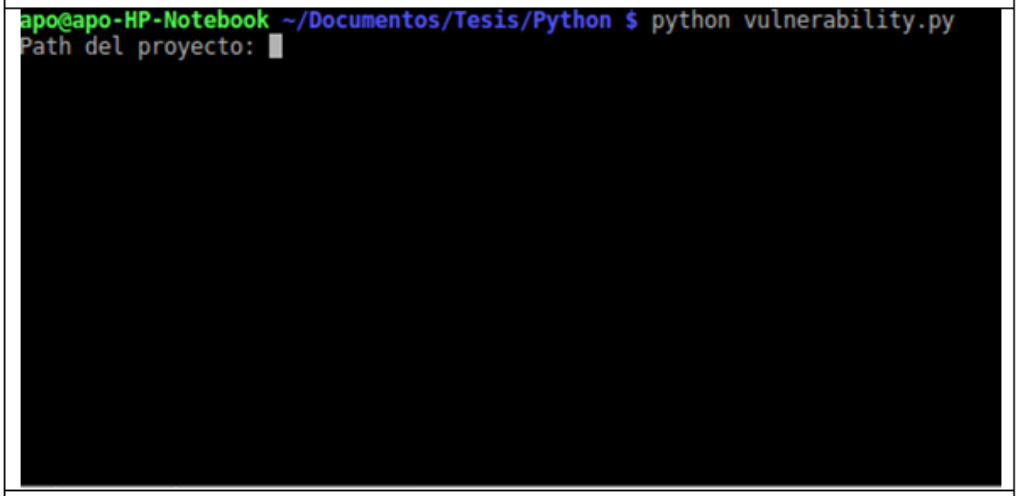
 # Ver si esta exportado por defecto con un intentfilter adentro (Si tiene android:exported = false no estara exportado aunque tenga un intentfilter). Si es así, obtener el componente y colocar exported en false.

```
if (checkIntent(line)):
agregarExportedFalse(componente_del_intent)
#Guardarlo previamente en variable

# Analizar si los providerestan exportados por defecto. AgregarExported false
if (isProvider(line))
if (analizeProvider(line)):
agregarExportedFalse(line)

# Analizar si es una aplicaciondebuggable. Modificarla a false
if (isDebuggable(line)):
modificarDebuggableAFalse(line)

#Escribir archivo final
writeNewAndroidManifestXml()
exit()
```

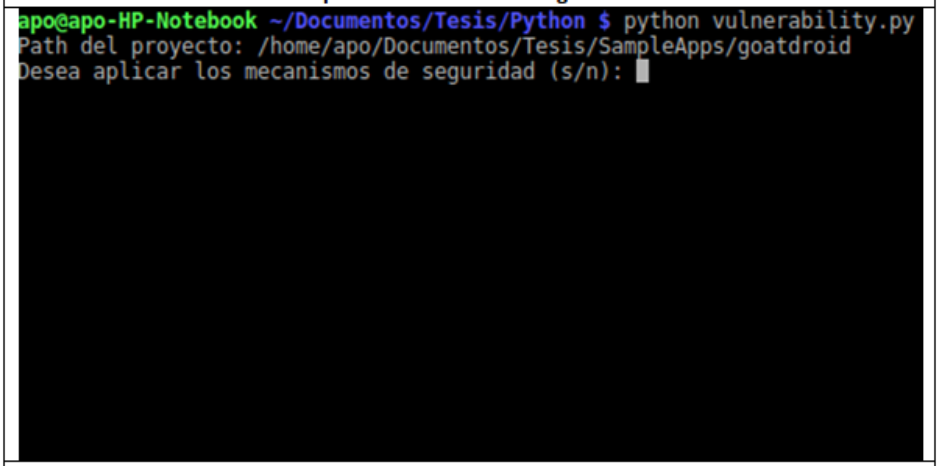
Diseño:

Para la fase de diseño, se desarrollará en las 2 primeras iteraciones el diseño de interfaz, y en la última iteración el diseño arquitectónico. Para la presente iteración se hará uso de la herramienta de diseño ***Layout***. Esta herramienta me permite crear el diseño de la vista, el cual se debe implementar en la fase de programación.

CUADRO Nº 5: Layout. Proceso: Desarrollador de aplicaciones *android* identifica vulnerabilidades

Sistema: Sistema de Hacking y Seguridad en aplicaciones móviles ***Proceso:*** Desarrollador de aplicaciones *android* identifica vulnerabilidades	**P01**
Identificar vulnerabilidades	
	
*1: El path no puede estar vacio.	

CUADRO Nº 6: Layout. Proceso: Desarrollador de aplicaciones *android* aplica mecanismos de seguridad a las vulnerabilidades detectadas.

Sistema: Sistema de Hacking y Seguridad en aplicaciones móviles ***Proceso:*** Desarrollador de aplicaciones *android* aplica mecanismos de seguridad a las vulnerabilidades detectadas	**P02**
Aplicar mecanismos de seguridad	
	
*1: El valor solo puede ser "s" o "n".	

➢ Iteración 2 – Web

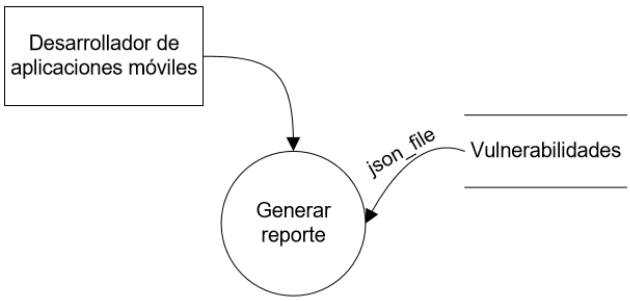
Análisis:

Herramientas a utilizar:

- ✓ **DFD:** Diagrama de flujo de datos**.**
- ✓ **DD'S:** Diccionario de datos**.**
- ✓ **EP:** Especificación de procesos**.**

➢ **Evento**: Desarrollador de aplicaciones *android* genera reporte del análisis.

✓ **DFD:**



✓ **Especificación caso de uso:**

Caso de uso: Generar reporte del análisis
Actor: Desarrollador de aplicaciones móviles
Pre condiciones: Archivo .Json con vulnerabilidades identificadas.
Post condiciones: Reporte de vulnerabilidades generado.

Escenario principal de éxito

Actor	Sistema
1- Iniciar nuevo C.U. Generar reporte del análisis.	2- Iniciar página principal. 4- Identificar Archivo .Json. 5-Identificar vulnerabilidades detectadas.

7- Visualiza reporte correctamente.	6- Mostrar información en vista. 8-Fin del C.U

- ✓ **DD'S:**

  json_file ={ id + type + { (action) **Especifica el action del intent que exporta el componente por defecto ** + (type_exported) + name_component } }

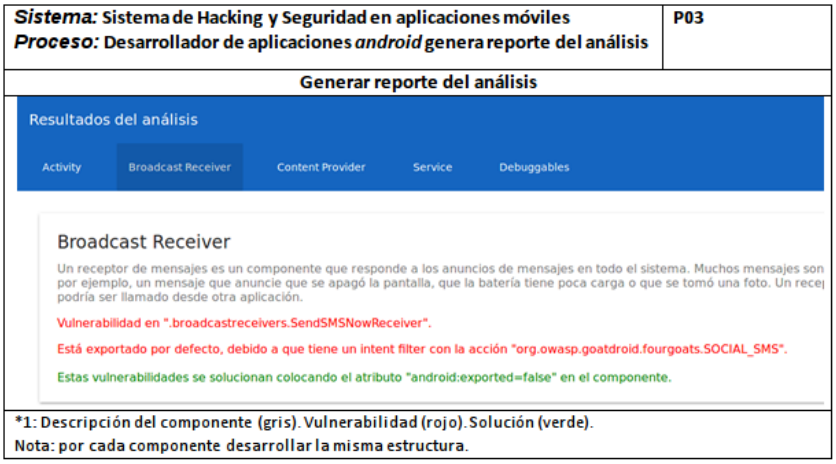
- ✓ **EP:**

```
vulnerabilities = getDatos()

if (not vulnerabilities):
exit()

# Por cada tipo de componente android, crear un componente angular que ejecute
foreach vulnerability as vulnerabilities:
if (vulnerability.type == componente):
mostrarVulnerabilidad(vulnerability)
mostrarSolucion(vulnerability)
```

Diseño:

Para representar el diseño de interfaz, haré uso de la herramienta de diseño ***Layout***, el cual me permite definir el diseño de la vista del reporte generado. El mismo se debe implementar en la fase de programación.

CUADRO Nº 7: Layout. Proceso: Desarrollador de aplicaciones *android* genera reporte del análisis.



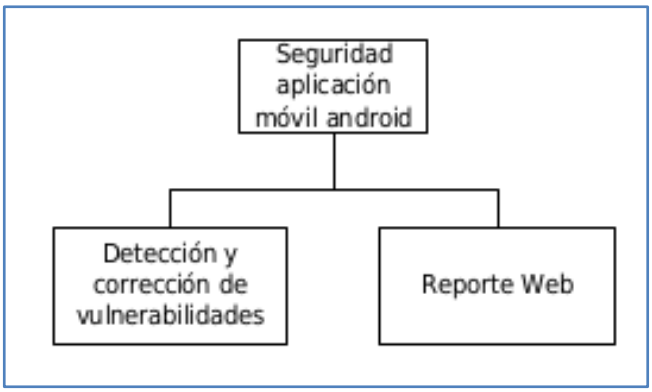
➢ Iteración 3 – Integración

El objetivo de esta iteración es modificar el diseño arquitectónico del sistema de manera de integrar el script desarrollado en la iteración 1 y el reporte web desarrollado en la iteración 2.

Diseño:

Diseño arquitectónico: Diagrama de estructuras.

CUADRO Nº 8: Diagrama de Estructura



Nota: el archivo *.json* exportado desde el *script* (Iteración 1) debe ser importado para leer el mismo y generar el reporte (iteración 2).

Consideraciones a tener en cuenta para implementar:

- Deberá implementarse en un servidor web local (*apache*).
- Crear una carpeta con el nombre "application" dentro de */var/www/html/*.
- Generar el proyecto de angular *"ngbuild –prod"*.
- Copiar la carpeta generada "*dist*" en a*pplication/.*
- Copiar el script *vulnerability.py* en la carpeta *application*/.

Consideraciones a tener en cuenta para hacer uso del sistema:

- El archivo manifiesto de la aplicación deberá tener cada etiqueta en una línea diferente.
- Si un componente tiene un *intent-filter* deberá seguir la siguiente secuencia de líneas:
  1. Componente.
  2. *Intent-Filter*.
  3. *Action*.
- El archivo manifiesto no debe contener líneas vacías.

4.D. Evaluación Económica Financiera

Los costos de inversión para el presente proyecto están reflejados en el cuadro que se expone a continuación:

CUADRO Nº 9: Costos de Inversiòn

Concepto	Cantidad	Días	Precio Unitario	Precio Total
Computadora Hp intelcore i5.	1	-	$15.358,62	$15.358,62
RRHH	**Cantidad**	**Días (8 Hs / Día)**	**Precio / Hs[60]**	**Precio Total**
Analista programador Ssr.	1	41	$200	$65.600,00
Totales	**2**	**41**	**-**	**$80.958,62**

Fuente: http://www.copaipa.org.ar/informatica/. *https://www.hponline.com.ar/c/notebooks*

El mismo, puede ser comparado con el costo que implicaría contratar mensualmente un personal de Administrador de Seguridad Informática.

CUADRO Nº 10: Costos de contratar un personal de seguridad

RRHH	Cantidad	Días (4 Hs / Día)	[61]Precio / Hs	Precio Total / Mes
Administrador de Seguridad Informática	1	20	$240	$19.200,00
Total	**-**	**-**	**-**	**$19.200,00**

Fuente: http://www.copaipa.org.ar/informatica/.

Se debería considerar una contratación mensual, ya que constantemente las empresas están realizando actualizaciones o desarrollando nuevas aplicaciones para sus

[60]Publicación deCopaipa - http://www.copaipa.org.ar/informatica/
[61]Publicación de Copaipa - http://www.copaipa.org.ar/informatica/

clientes. Observamos entonces, que al finalizar el quinto mes, estaría superando el costo del proyecto. Con lo cual, el desarrollo del sistema tiende a reducir los costos empresariales.

VAN

Es importante analizar la rentabilidad y viabilidad del proyecto, el cual se hará uso de la fórmula del VAN y TIR para analizar estos aspectos.

$$VAN = -I + \sum_{n=1}^{N} \frac{Qn}{(1 + k)^n}$$

I: inversión inicial = 15.358,62 (Computadora)

K: tasa de descuento = 5% / 12 (meses) = 0,004167

N: cantidad de meses / años

Qn: flujo de caja en el mes / año n =

Cabe destacar, que el proyecto tiene como desarrollo 41 días hábiles, el mismo, implica 2 meses y 1 día. Significa, que la empresa que desee contratar el proyecto deberá hacer la inversión inicial y seguir invirtiendo en el personal de seguridad informática por este período de tiempo, hasta el momento de la implementación del proyecto. Se asume, que se toma como flujo de caja únicamente el valor de contratar un personal de seguridad informática y el costo de seguir contratando el personal de RRHH hasta la implementación del proyecto, independientemente de los ingresos de la empresa.

El costo de seguir invirtiendo en personal de seguridad informática es 19.200. Mientras que el costo de RRHH para el presente proyecto para el mes 1 es

RRHH (mes 1) = 20 * 200 * 8 = 32.000

Dado que estos 2 valores son egresos, queda como flujo de caja:

- **Q1: - 51.200**
- **Q2: - 51.200**

El costo de contratar 1 día el servicio de seguridad informática se calcula como 19.200 / 20 = 960. A partir de este período se considera ingresos a los 19.200. Restandole 960 queda como ahorro del tercer mes 18.240 (ingresos).

El costo de RRHH para el tercer mes (1 dia) es 1.600 (egreso). Haciendo el calculo 18.240 – 1.600 = 16.640.

- **Q3: 16.640**

A partir del mes 4 en adelante se considera ingresos a 19.200, sin costo de RRHH ya que el sistema está implementado para este período.

- **Q4 a Q12: 19.200**

Con estos parámetros el cálculo de VAN para el mes 12 es:

VAN = 66.467,71

Dado que la suma de todos los flujos esperados a futuros actualizados a la tasa de descuento de 5% son mayores a la inversión inicial, asumimos que el proyecto es rentable.

TIR

Indica cual es la rentabilidad del proyecto en el período n. Cuando VAN = 0. Entonces, TIR > r (usado para calcular VAN) para que el proyecto sea viable.

TIR = 7,19% > 5%

En este caso, la tasa de rendimiento interno que obtenemos es superior a la tasa mínima de rentabilidad exigida a la inversión.

5. RESULTADOS O VERIFICACIÓN EXPERIMENTAL

5.A. Experimento y prueba realizada.

GoatDroid es una aplicación A*ndroid* desarrollada por *OWASP*, específicamente para realizar todo el proceso de *pentesting*. Dicha aplicación, tiene vulnerabilidades que han sido desarrolladas con el propósito de estudiar los mismos. Es de código abierto y puede ser descargada del siguiente vínculo:

https://github.com/downloads/jackMannino/OWASP-GoatDroid-Project/OWASP-GoatDroid-0.9.zip

Para hacer uso del sistema desarrollado, tenemos 2 opciones:

1. Trabajar directamente con código fuente de la aplicación móvil.
2. Descargar el archivo *.apk* y hacer un proceso de ingeniería inversa para obtener el código fuente.

En èstecaso en particular, se ha descargado el archivo *.apk* de *GoatDroid*, realizando un proceso de ingeniería inversa para obtener los archivos y códigos de la aplicación.

El archivo *.apk* de *GoatDroid* lo podemos obtener de:

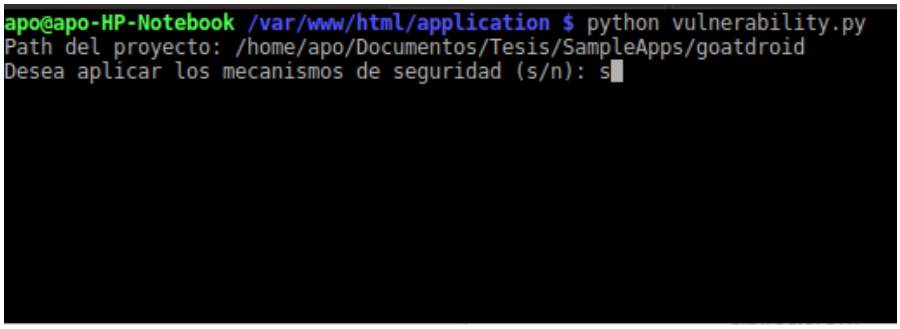
https://github.com/linkedin/qark/blob/master/qark/sampleApps/goatdroid/goatdroid.apk

Para realizar el proceso de ingeniería inversa, se utiliza la herramienta *Apktool*[62].

Verificar que el archivo AndroidManifest.xml pueda ser leído. Un vez que hemos realizado este proceso podemos comenzar a hacer uso del sistema.

Luego, en una terminal, cambiando el directorio a la carpeta del proyecto para comenzar a usar el *script* en *Python*, llamando al intérprete de Python y al archivo *vulnerability.py*. Introduzco el *path*de *GoatDroid*. Para este caso en particular, aplicaré los mecanismos de seguridad automáticos colocando la opción **"s"**.

[62] Anexo 1



5.B. Resultados Obtenidos

Una vez colocado estos parámetros, el *script* comenzará a buscar vulnerabilidades y a corregirlas automáticamente. Generando un reporte con el análisis. El mismo se presenta a continuación:

***Activity*:**

Ampliando los resultados observo las vulnerabilidades:

```
Vulnerabilidad en ".activities.ViewCheckin".
Está exportado explicitamente, debido a que tiene el atributo "android:exported = true".

Vulnerabilidad en ".activities.ViewProfile".
Está exportado explicitamente, debido a que tiene el atributo "android:exported = true".

Vulnerabilidad en ".activities.SocialAPIAuthentication".
Está exportado explicitamente, debido a que tiene el atributo "android:exported = true".

Estas vulnerabilidades se solucionan colocando el atributo "android:exported=false" en el componente.
```

- El componente ".activities.ViewCheckin" está exportado explícitamente.
- El componente "*.activities.ViewProfile*" está exportado explícitamente.
- El componente "*.activities.SocialApiAuthentication*" está exportado explícitamente.

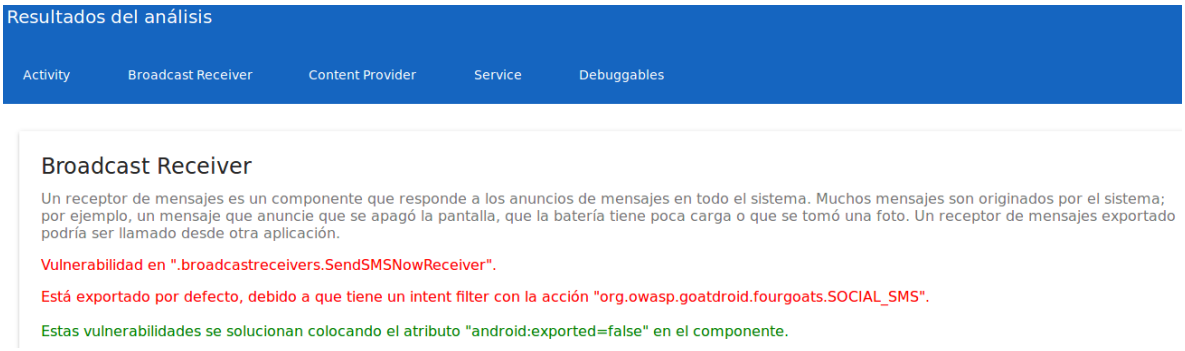
Si esta aplicación estaría en producción, un usuario podría evitar el *login* de la aplicación ejecutando cualquiera de estas actividades.

Verifico el archivo *AndroidManifest.xml* para observar si estas vulnerabilidades han sido corregidas correctamente.

```
AndroidManifest.xml
<activity android:label="@string/rewards" android:name=".activities.Rewards"/>
<activity android:exported="false" android:label="@string/view_checkin" android:name=".activities.ViewCheckin"/>
<activity android:exported="false" android:label="@string/profile" android:name=".activities.ViewProfile"/>
<activity android:label="@string/friend_request" android:name=".activities.ViewFriendRequest"/>
<activity android:label="@string/my_rewards" android:name=".fragments.MyRewards"/>
<activity android:label="@string/available_rewards" android:name=".fragments.AvailableRewards"/>
<activity android:label="@string/preferences" android:name=".activities.Preferences"/>
<activity android:label="@string/about" android:name=".activities.About"/>
<activity android:label="@string/send_sms" android:name=".activities.SendSMS"/>
<activity android:label="@string/comment" android:name=".activities.DoComment"/>
<activity android:label="@string/history" android:name=".activities.UserHistory"/>
<activity android:label="@string/destination_info" android:name=".activities.DestinationInfo"/>
<activity android:label="@string/admin_home" android:name=".activities.AdminHome"/>
<activity android:label="@string/admin_options" android:name=".activities.AdminOptions"/>
<activity android:label="@string/reset_user_passwords" android:name=".fragments.ResetUserPasswords"/>
<activity android:label="@string/delete_users" android:name=".fragments.DeleteUsers"/>
<activity android:label="@string/reset_user_password" android:name=".activities.DoAdminPasswordReset"/>
<activity android:exported="false" android:label="@string/authenticate" android:name=".activities.SocialAPIAuthentication"/>
<activity android:label="@string/app_name" android:name=".activities.GenericWebViewActivity"/>
<provider android:exported="false" android:name=".NoteProvider" android:authorities="com.sonyericsson.notes.provider.Note" />
<service android:exported="false" android:name=".services.LocationService">
<intent-filter>
```

Los 3 componentes se han corregido correctamente con el atributo *"android:exported:false"*.

***Broadcast Receiver*:**



Ampliación de los resultados de las vulnerabilidades:

- El componente *".broadcastreceivers.SendSMSNowReceiver"* está exportado por defecto, debido a que tiene un *intent-filter*.

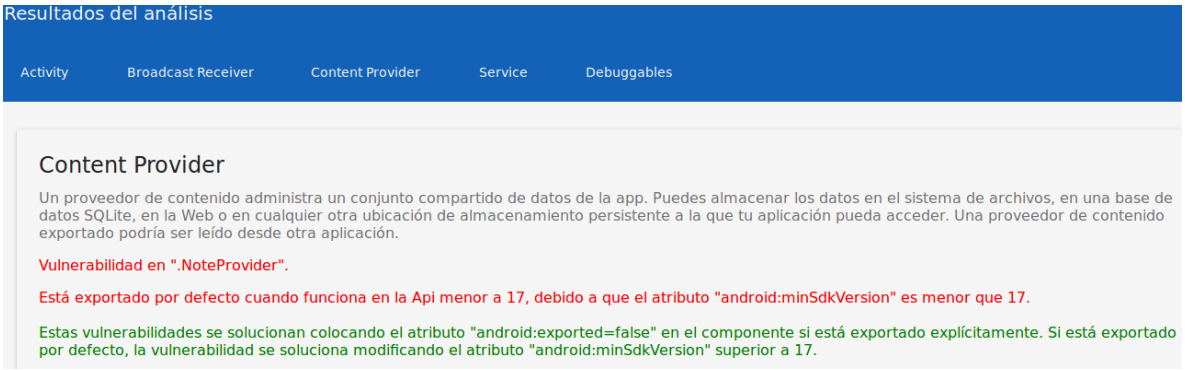
La descripción del nombre del componente nos indica que podría (analizar el código fuente del componente) llegar a ser un receptor de mensajes para enviar **sms**. Una aplicación maliciosa podría llamar a este componente y enviar mensajes indefinidamente, provocándole costos al usuario.

Observando el archivo manifiesto, verifico que esta vulnerabilidad se haya corregido correctamente:

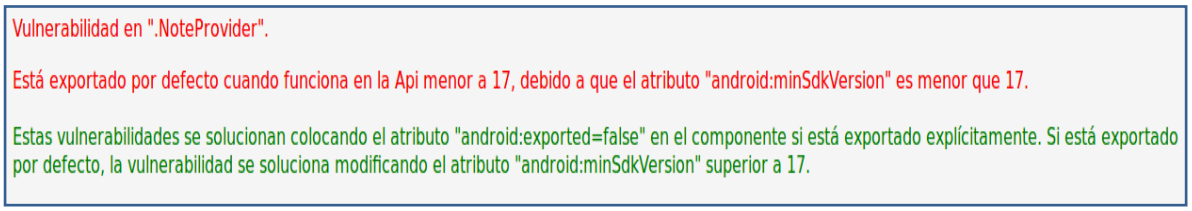
```
<intent-filter>
<action android:name="org.owasp.goatdroid.fourgoats.services.LocationService"/>
</intent-filter>
</service>
<receiver android:exported="false" android:label="Send SMS" android:name=".broadcastreceivers.SendSMSNowReceiver">
<intent-filter>
<action android:name="org.owasp.goatdroid.fourgoats.SOCIAL_SMS"/>
</intent-filter>
</receiver>
</application>
```

La vulnerabilidad ha sido corregida debido a que en el componente se asigna *"android:exported=false"*.

Content Provider:



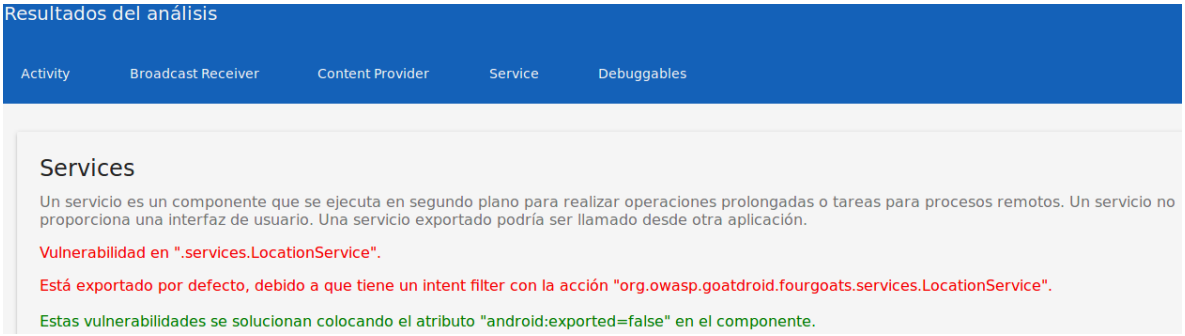
Ampliación de los resultados de las vulnerabilidades:



- El componente ".NoteProvider" está exportado por defecto para las versiones de *android* que ejecutan la Api menor a 17.

Esta vulnerabilidad debe corregirlo manualmente, dado que modificar el atributo *"android:minSdkVersion"* podría tener problemas al volver a compilar la aplicación dado que pueden existir paquetes que haya incluido que requieren ser ejecutadas en esta versión.

***Services*:**

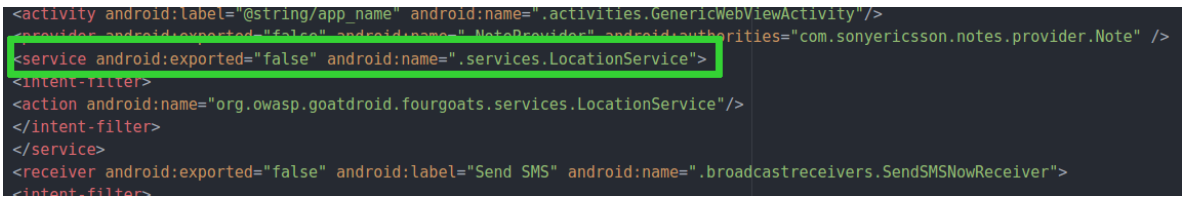


Ampliaciòn de los resultados de las vulnerabilidades:

- El componente *".services.LocationService"* está exportado por defecto, debido a que tiene un *intent-filter*.

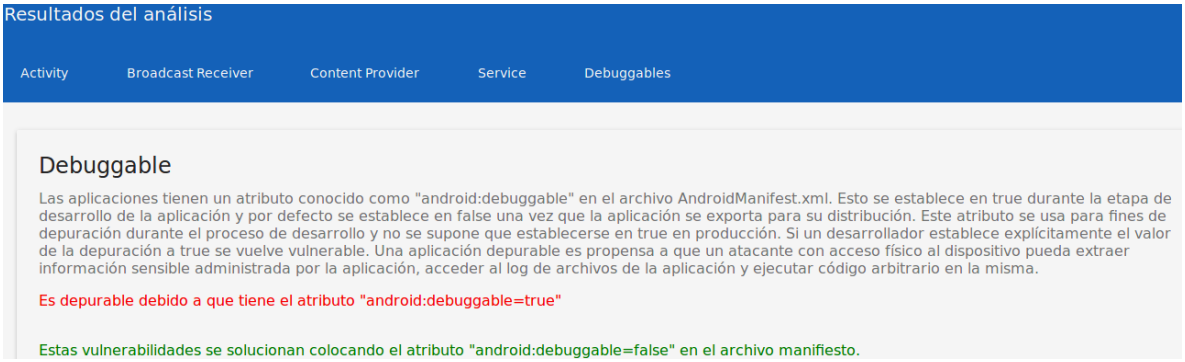
La descripción del nombre del componente nos indica que podría tratarse de un servicio que administra las posiciones via GPS del usuario. Un usuario con una aplicación maliciosa podría acceder a dicho servicio de manera externa.

Verifico el archivo manifiesto para observar si la vulnerabilidad se ha corregido correctamente:

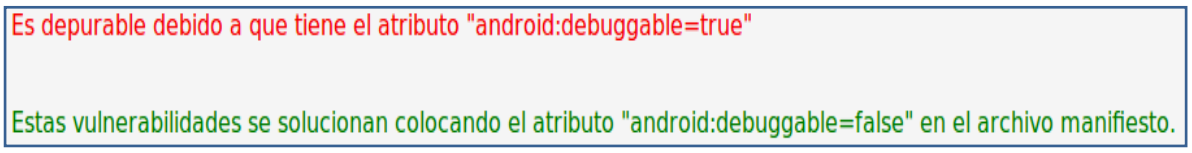


El atributo *"android:exported=false"* está presente en el componente ahora, corrigiendo de manera exitosa la vulnerabilidad.

Debuggable:

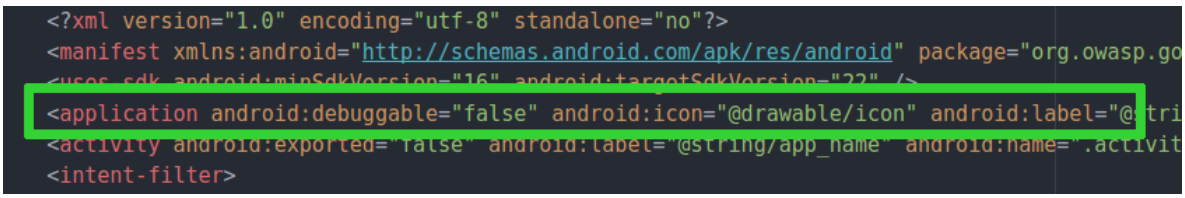


Ampliaciòn de los resultados de la vulnerabilidad:



- En el archivo manifiesto se observa el atributo *"android:debuggable=true"* con lo cual un usuario malicioso con acceso al dispositivo podría extraer información sensible, acceder al registro de eventos y ejecutar código arbitrario en la misma.

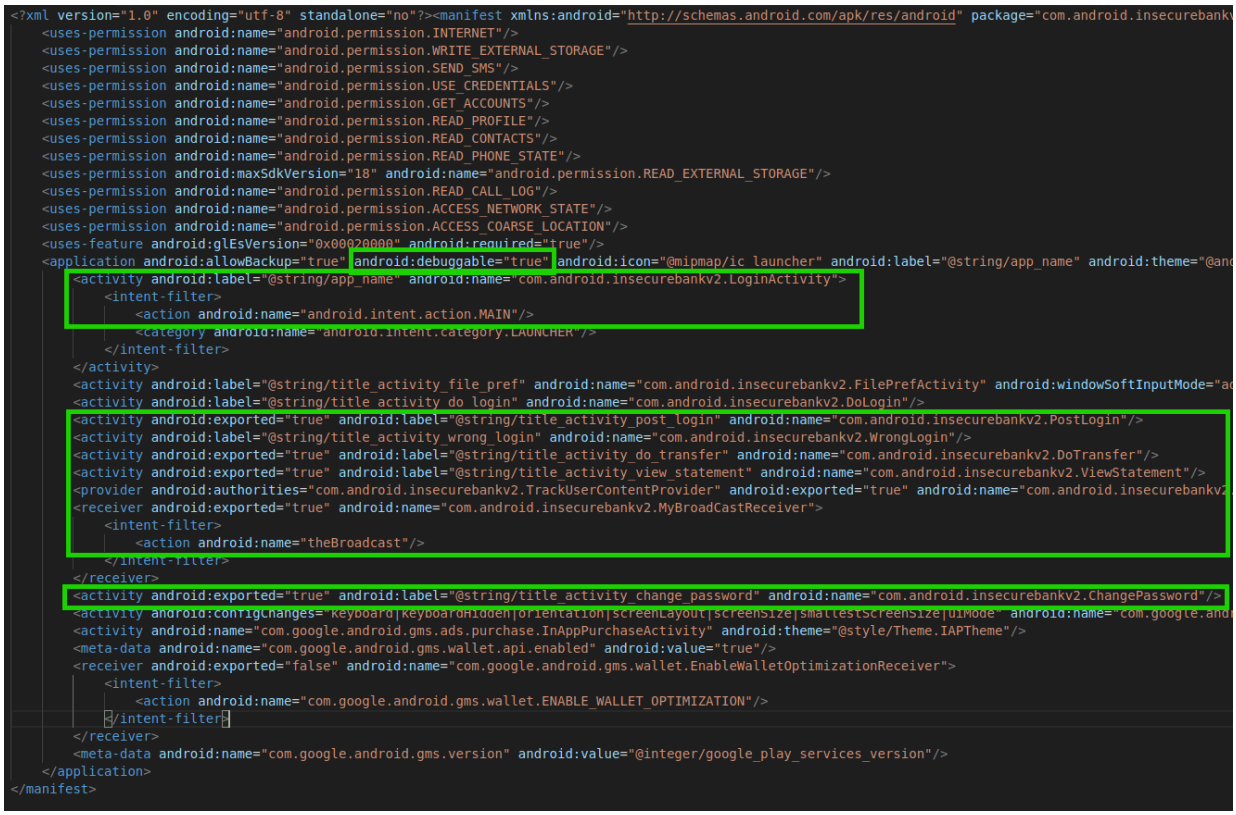
Verificando el archivo manifiesto:



Observo que el atributo *"android:debuggable=true"* ha sido modificado a false, solucionando esta vulnerabilidad.

A continuación se hará la prueba con la aplicación "*Insecure Bank V2*" [63], aplicación vulnerable desarrollada para realizar proceso de pentesting. Dado que anteriormente se ha detallado como instalar y ejecutar el sistema desarrollado, a continuación se procederá a mostrar las vulnerabilidades identificadas, el reporte obtenido y la solución implementada para *Insecure Bank V2.*

Vulnerabilidades identificadas:



- ✓ *Aplicación debuggable*
- ✓ *Activities:*
  - *LoginActivity*
  - *PostLogin*
  - *DoTransfer*
  - *ViewStatement*
  - *WrongLogin*
  - *ChangePassword*
- ✓ *BroadcastReceiver:*

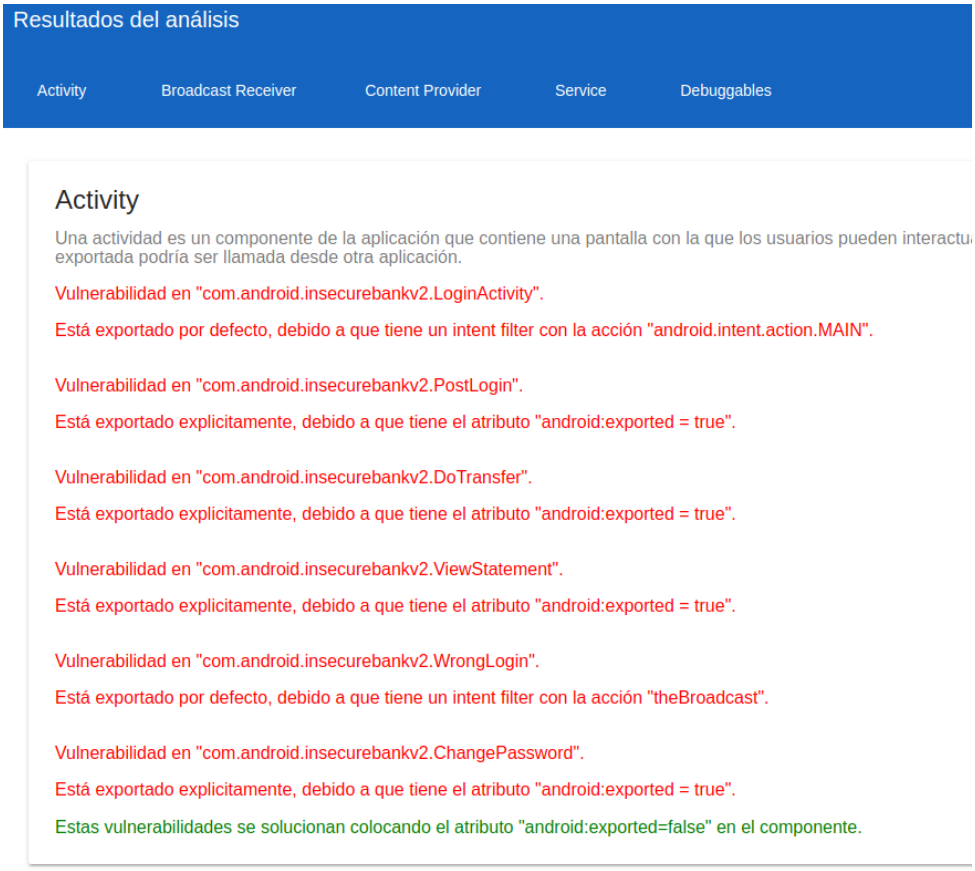
[63] Publicación en GitHub de Dinesh Shetty: https://github.com/dineshshetty/Android-InsecureBankv2

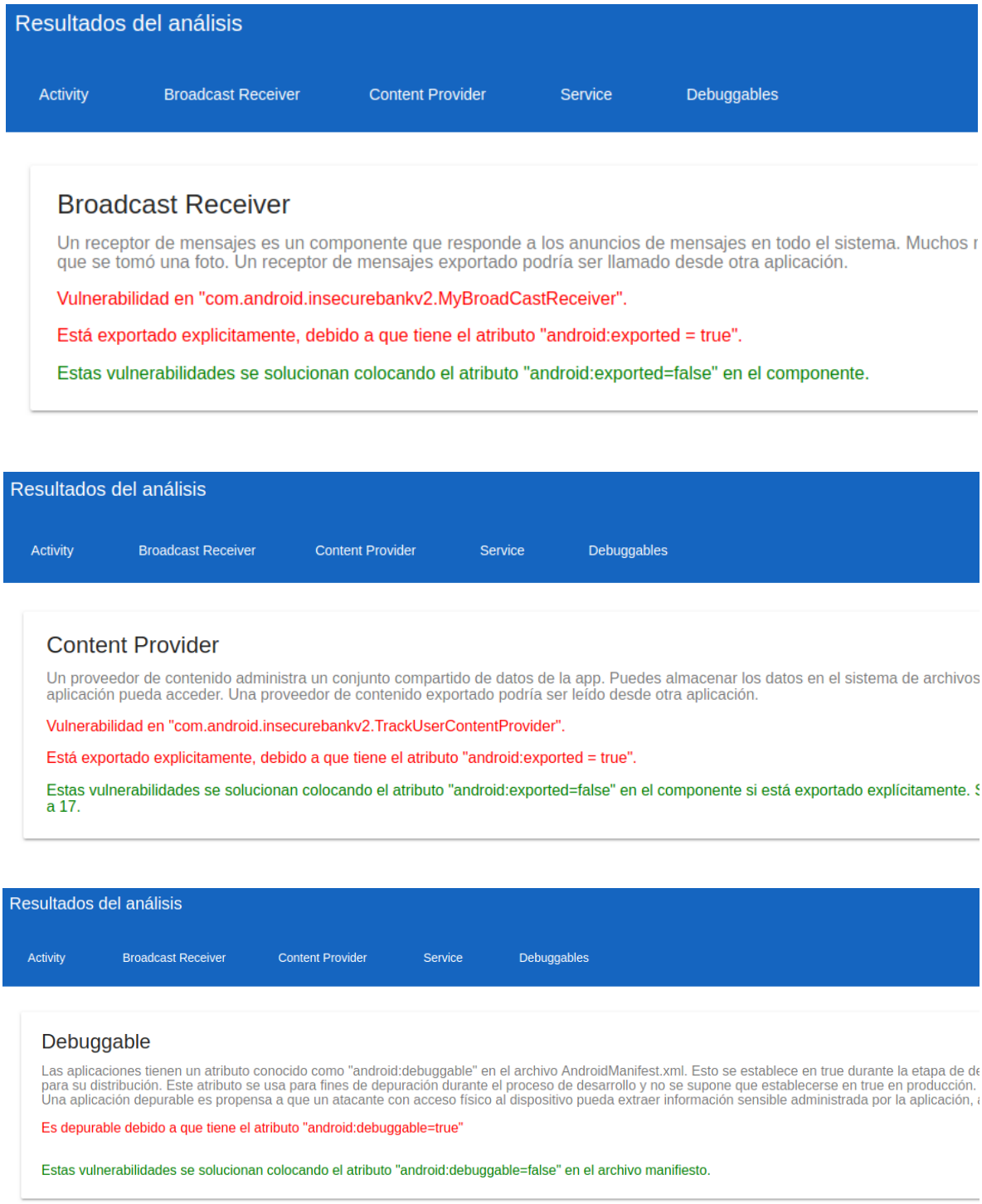
- *MyBroadCastReceiver*

✓ *Content Provider:*

- *TrackUserContentProvider*

Reporte obtenido luego del análisis:





Al examinar el nuevo AndroidManifest.xml Generado observamos la solución a las vulnerabilidades encontradas:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.android.insecu
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.USE_CREDENTIALS"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.READ_PROFILE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:maxSdkVersion="18" android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
<application android:allowBackup="true" android:debuggable="false" android:icon="@mipmap/ic_launcher" android:label="@string/app_name" android:theme="
<activity android:exported="false" android:label="@string/app_name" android:name="com.android.insecurebankv2.LoginActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:label="@string/title_activity_file_pref" android:name="com.android.insecurebankv2.FilePrefActivity" android:windowSoftInputMode="adj
<activity android:label="@string/title_activity_do_login" android:name="com.android.insecurebankv2.DoLogin"/>
<activity android:exported="false" android:label="@string/title_activity_post_login" android:name="com.android.insecurebankv2.PostLogin"/>
<activity android:label="@string/title_activity_wrong_login" android:name="com.android.insecurebankv2.WrongLogin"/>
<activity android:exported="false" android:label="@string/title_activity_do_transfer" android:name="com.android.insecurebankv2.DoTransfer"/>
<activity android:exported="false" android:label="@string/title_activity_view_statement" android:name="com.android.insecurebankv2.ViewStatement"/>
<provider android:authorities="com.android.insecurebankv2.TrackUserContentProvider" android:exported="false" android:name="com.android.insecurebank
<activity android:exported="false" android:label="@string/title_activity_wrong_login" android:name="com.android.insecurebankv2.WrongLogin"/>
<intent-filter>
<action android:name="theBroadcast"/>
</intent-filter>
</receiver>
<activity android:exported="false" android:label="@string/title_activity_change_password" android:name="com.android.insecurebankv2.ChangePassword"/>
<activity android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|screenSize|smallestScreenSize|uiMode" android:name="com.google.andro
<activity android:name="com.google.android.gms.ads.purchase.InAppPurchaseActivity" android:theme="@style/Theme.IAPTheme"/>
<meta-data android:name="com.google.android.gms.wallet.api.enabled" android:value="true"/>
<receiver android:exported="false" android:name="com.google.android.gms.wallet.EnableWalletOptimizationReceiver">
<intent-filter>
<action android:name="com.google.android.gms.wallet.ENABLE_WALLET_OPTIMIZATION"/>
</intent-filter>
</receiver>
<meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version"/>
</application>
</manifest>
```

Como tercer y última prueba, haremos la ejecución en la aplicación "*Sieve*" [64], aplicación destinada para realizar proceso pentesting.

Vulnerabilidades identificadas:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android"
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <permission android:label="Allows reading of the Key in Sieve" android:name="com.mwr.example.sieve.READ_KEYS" android:p
    <permission android:label="Allows editing of the Key in Sieve" android:name="com.mwr.example.sieve.WRITE_KEYS" android
    <application android:allowBackup="true" android:debuggable="true" android:icon="@drawable/ic_launcher" android:label="
        <activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:exported="true" android:finish
        <activity android:excludeFromRecents="true" android:label="@string/app_name" android:launchMode="singleTask" androi
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:exported="true" android:finish
        <activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="true" andr
        <activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="true" andr
        <activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="true" andr
        <activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="true" andr
        <activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="true" andr
        <service android:exported="true" android:name=".AuthService" android:process=":remote"/>
        <service android:exported="true" android:name=".CryptoService" android:process=":remote"/>
        <provider android:authorities="com.mwr.example.sieve.DBContentProvider" android:exported="true" android:multiproces
            <path-permission android:path="/Keys" android:readPermission="com.mwr.example.sieve.READ_KEYS" android:writePer
        </provider>
        <provider android:authorities="com.mwr.example.sieve.FileBackupProvider" android:exported="true" android:multiproc
    </application>
</manifest>
```

[64] Publicacion de MWR Labs https://labs.mwrinfosecurity.com/

- ✓ *Aplicación debuggable*
- ✓ *Activities*:
  - *FileSelectActivity*
  - *MainLoginActivity*
  - *PWList*
- ✓ *Content Provider*:
  - *DBContentProvider*
  - *FileBackupProvider*
- ✓ *Services*:
  - *AuthService*
  - *CryptoService*

Reporte obtenido luego del análisis:

Activity

Una actividad es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden i
exportada podría ser llamada desde otra aplicación.

Vulnerabilidad en ".FileSelectActivity".

Está exportado explicitamente, debido a que tiene el atributo "android:exported = true".

Vulnerabilidad en ".MainLoginActivity".

Está exportado por defecto, debido a que tiene un intent filter con la acción "android.intent.action.MAIN".

Vulnerabilidad en ".PWList".

Está exportado explicitamente, debido a que tiene el atributo "android:exported = true".

Estas vulnerabilidades se solucionan colocando el atributo "android:exported=false" en el componente.

Debuggable

Las aplicaciones tienen un atributo conocido como "android:debuggable" en el archivo AndroidManifest.xml. Esto
para su distribución. Este atributo se usa para fines de depuración durante el proceso de desarrollo y no se supo
Una aplicación depurable es propensa a que un atacante con acceso físico al dispositivo pueda extraer informac

Es depurable debido a que tiene el atributo "android:debuggable=true"

Estas vulnerabilidades se solucionan colocando el atributo "android:debuggable=false" en el archivo manifiesto.

Content Provider

Un proveedor de contenido administra un conjunto compartido de datos de la app. Puedes almacenar los datos
aplicación pueda acceder. Una proveedor de contenido exportado podría ser leído desde otra aplicación.

Vulnerabilidad en ".DBContentProvider".

Está exportado explicitamente, debido a que tiene el atributo "android:exported = true".

Vulnerabilidad en ".FileBackupProvider".

Está exportado explicitamente, debido a que tiene el atributo "android:exported = true".

Estas vulnerabilidades se solucionan colocando el atributo "android:exported=false" en el componente si está ex
a 17.

Services

Un servicio es un componente que se ejecuta en segundo plano para realizar operaciones prolongadas o tareas para procesos remotos.

Vulnerabilidad en ".AuthService".

Está exportado explicitamente, debido a que tiene el atributo "android:exported = true".

Vulnerabilidad en ".CryptoService".

Está exportado explicitamente, debido a que tiene el atributo "android:exported = true".

Estas vulnerabilidades se solucionan colocando el atributo "android:exported=false" en el componente.

Al examinar el nuevo AndroidManifest.xml Generado observamos la solución a las vulnerabilidades encontradas:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/a
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<permission android:label="Allows reading of the Key in Sieve" android:name="com.mwr.example.sieve.READ_KEYS
<permission android:label="Allows editing of the Key in Sieve" android:name="com.mwr.example.sieve.WRITE_KE
<application android:allowBackup="true" android:debuggable="false" android:icon="@drawable/ic_launcher" andr
<activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:exported="false" andro
<activity android:exported="false" android:excludeFromRecents="true" android:label="@string/app_name" andro
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:exported="false" andro
<activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="tr
<activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="tr
<activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="tr
<activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="tr
<activity android:clearTaskOnLaunch="true" android:excludeFromRecents="true" android:finishOnTaskLaunch="tr
<service android:exported="false" android:name=".AuthService" android:process=":remote"/>
<service android:exported="false" android:name=".CryptoService" android:process=":remote"/>
<provider android:authorities="com.mwr.example.sieve.DBContentProvider" android:exported="false" android:mu
<path-permission android:path="/Keys" android:readPermission="com.mwr.example.sieve.READ_KEYS" android:write
</provider>
<provider android:authorities="com.mwr.example.sieve.FileBackupProvider" android:exported="false" android:mu
</application>
</manifest>
```

Queda demostrado, que el sistema ha detectado y corregido correctamente las vulnerabilidades detectadas en las aplicaciones probadas.

5.C. Calidad de los resultados en comparación con otras soluciones

Qark[65] es una herramienta muy utilizada en la actualidad para hacer análisis estático de una aplicación *android*. Esta herramienta fue desarrollada y es mantenida por el equipo de seguridad de la empresa L*inkedinCorporation*. El mismo, es de código abierto y cualquier persona puede leer el código y modificarlo para adaptarlo a su uso.

Qark realiza además otros tipos de análisis, como *backup*, explotación de *WebViews*, análisis de certificados de validación, análisis de criptografía, entre otros. *Qark* y mi proyecto presentado tienen similitudes y diferencias. La similitud es que hacen un análisis de detección de vulnerabilidades basándose en el análisis estático, generando un reporte. La diferencia fundamental con esta herramienta es que**, mi proyecto concretaademás de hacer el análisis, la correcciónde estas vulnerabilidades de manera automática, brindando además, capacitación al usuario sobre las vulnerabilidades encontradas y como solucionarlas.**

5.D. Dificultades no previstas encontradas y sus soluciones

Al momento de implementar me he encontrado con la dificultad de cómo integrar ambas partes del desarrollo (*script y web*), dado que el *frameworkAngular* me permitía leer archivos externos (generado por Python) a través del protocolo http (local). Mi idea en un principio, era que el usuario final del sistema no tenga que instalar un servidor web local, para facilitar el uso de la misma. Buscando soluciones a este problema, encontré que al momento de generar el proyecto *Angular*introducia este archivo en su faz interna. Como consecuencia, siempre sería el mismo y sin posibilidad de modificarlo o cambiarlo. Por éste motivo,decidí solucionar dicho problema, leyendo archivos externos a través del protocolo Http e instalando un servidor web local para su uso.

[65] Publicación en GitHub de LinkedinCorporation. https://github.com/linkedin/qark

6. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

En los albores de este proyecto, he tenido como propósito mejorar las condiciones en que se desarrollan las aplicaciones móviles, automatizando procesos para minimizar errores y darle mayores beneficios a los desarrolladores de aplicaciones, dado que estas condiciones exponen a la sociedad en general.

El objetivo principal se centróen brindar una herramienta para permitr desarrollar aplicaciones con un índice de vulnerabilidad menor.A lo largo del presente proyecto, desarrollé un sistema que me permite solucionar el problema con las características planteadas.

Dado que mi solución se abordó desde el punto de vista del análisis estático de la aplicación, **como futuras investigaciones se podría desarrollar el análisis dinámico de las aplicaciónes**, es decir, en tiempo de ejecución. Partiendo principalmente, de una aplicación con componentes que no están exportados, analizar si es posible que al modificar el código lograr exportarlos en tiempo de ejecución y ser llamados desde una aplicación maliciosa para así brindar una solución. Además, se podría hacer otro tipo de análisis, tales como, verificación de componente encontrado, chequeo de memoria, bases de datos, copias de seguridad encontrados y análisis de permisos de accesos.

Finalmente deseo destacar quehaciéndome eco de los valores éticos y morales de mi formación, como así también respondiendo a los preceptos de la Responsabilidad Social de nivel profesional que me competen, es que se desarrolló el presente proyecto, con la sola intención de concretar un servicio de seguridad informática destinada a la población con fines netamente sociales. Por tal motivo, es que dejo establecido que el mismo no tendrá fines de lucro y el código fuente queda a disposición de la sociedad, con el objetivo de que los mismos puedan, utilizarlo, modificarlo y/o mejorarlo. Con ésta intenciónse deja a disposición un CD con los archivos originales de desarrollo.

7. Bibliografía y Fuentes de Información

BIBLIOGRAFÍA:

- Edward Yourdon, 1993. Análisis Estructurado Moderno. 735 Páginas. Prentice Halls Hispanoamericana, S.A., México. ISBN 0-13-598624-9.
- Kotipalli& Imran, 2016. Hacking Android. 353 Páginas. Packt Publishing Ltd. Birmingham B3 2PB, UK. ISBN 978-1-78588-314-9.
- Project Management Institute, 2013. Guía de los Fundamentos para la Dirección de Proyectos (Guía de PMBOK). Quinta Edición. 568 páginas. Project Management Institute, EE.UU. Inc. ISBN 978-1-62825-009-1.
- Andrew S. Tanenbaum, 2009. Sistemas Operativos Modernos. Tercera Edición. 1076 páginas. Pearson Education, México. ISBN: 978-607-442-046-3.
- Kenneth C. Louden, 2004. Construcción de compiladores: principios y práctica. 582 páginas. Thomson. ISBN 9706862994, 9789706862990.

WEBGRAFÍA Y HEMEROGRAFÍA:

- Biblioteca virtual Universidad Católica de Chile.
  - ✓ http://repositorio.uchile.cl/bitstream/handle/2250/144529/Evaluaci%C3%B3n-de-la-seguridad-de-aplicaciones-m%C3%B3viles-bancarias.pdf?sequence=1
- Biblioteca virtual Universidad Nacional de la Plata. Facultad de informática. Laboratorio de investigación de nuevas tecnologías informáticas
  - ✓ http://sedici.unlp.edu.ar/bitstream/handle/10915/43678/Documento_completo.pdf?sequence=1
- Boris Balacheff et ali., "Trusted computing platforms. TCPA technology in context". Prentice Hall PTR 2003
- Casos de “Crackers” de aplicaciones móviles en el 2018
  - ✓ http://www.abc.es/tecnologia/redes/abci-instagram-hackean-instagram-y-roban-cuentas-celebrities-culpa-fallo-seguridad-__201708311230_noticia.html
  - ✓ https://www.clarin.com/economia/hackean-aplicacion-marca-ropa-deportiva-difunden-datos-150-millones-usuarios_0_rJt9laiqG.html
- Dante Odín Ramírez López, Carmina Cecilia Espinosa Madrigal; Informe especializado en Universidad Autónoma de México
  - ✓ http://revista.seguridad.unam.mx/numero-10/el-cifrado-web-ssltls
- David González Verdugo; Informe especializado en Solid Gear
  - ✓ https://solidgeargroup.com/inyeccion-sql-en-proveedores-de-contenido-de-android-y-como-protegerse?lang=es
- Dennis K. Branstad,"Report of the Nist Workshop on Digital Signature Certificate anagement". U.S. Department of Commerce 1983

- Ignacio Pérez; Informe especializado en Welivesecurity
  - ✓ https://www.welivesecurity.com/la-es/2015/04/29/vulnerabilidad-xss-cross-site-scripting-sitios-web/
- Informe de HOOTSUITE
  - ✓ https://hootsuite.com/es/pages/digital-in-2018
- Informe en página especializada Sgoliver
  - ✓ http://www.sgoliver.net/blog/content-providers-en-android-i-construccion/
- Informe especializado en AnakamaSupport
  - ✓ https://support.ankama.com/hc/es/articles/203790076--Qu%C3%A9-es-un-log-
- Informe especializado en InfosecInstitute
  - ✓ http://resources.infosecinstitute.com/android-hacking-security-part-4-exploiting-unintended-data-leakage-side-channel-data-leakage/#gref
- Informeespecializadoen Massachusetts Institute of Technology
  - ✓ http://web.mit.edu/rhel-doc/3/rhel-sag-es-3/ch-logfiles.html
- Informe especializado en Universidad Carlos III de Madrid; Arquitectura Android
  - ✓ *https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android*
- Informe especializado en Universidad Internacional de Valencia
  - ✓ https://www.universidadviu.es/la-seguridad-informatica-puede-ayudarme/
- Kenneth Amaditz; Informe especializado en Trustdimension
  - ✓ http://www.trustdimension.com/la-importancia-de-la-seguridad-informatica/
- KristelMalave Polanco, José Luis BeauperthuyTaibo; Artículo científico especializado de NEGOTIUM
  - ✓ http://ojs.revistanegotium.org.ve/index.php/negotium/article/view/248/235
- Liliana CújarBahamón; Publicación en liliseguridadinformatica
  - ✓ https://liliseguridadinformatica.webnode.es/guia-de-buenas-practicas/diseno/a9/
- Mauro Maulini R.; Publicación en Tecnologías Web
  - ✓ http://tecnologiasweb.blogspot.com.ar/2008/10/cross-site-scripting-xss-otro-enemigo.html
- Mobile Top Ten Contributions - https://www.owasp.org/index.php/Mobile_Top_Ten_Contributions
- Publicación de SEARS
  - ✓ http://www.sears.com.mx/celulares/la-importancia-de-los-celulares-en-la-vida-cotidiana/
- Publicación de Apktool - https://ibotpeaches.github.io/Apktool/documentation/
- Publicación de Babylon
  - ✓ http://thesaurus.babylon-software.com/hard-coded
- Publicación de Forodeseguridad - http://www.forodeseguridad.com/artic/discipl/4163.htm
- Publicación de Google - https://support.google.com/faqs/answer/7496913?hl=en
- Publicación de Google en Developer Android

- ✓ https://developer.android.com/guide/components/fundamentals?hl=es

- Publicación de Google en Developer Android
  - ✓ https://developer.android.com/guide/topics/manifest/uses-sdk-element?hl=ES#target

- Publicación de Instituto Tecnológico y de Estudios Superiores de Monterrey
  - ✓ http://www.scielo.org.mx/scielo.php?pid=S0188- 52X2010000100008&script=sci_arttext
- Publicación de OWASP
  - ✓ https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks
- Publicación de OWASP
  - ✓ https://www.owasp.org/index.php/Top_10_2007-Autenticaci%C3%B3n_y_Gesti%C3%B3n_de_Sesiones_Disfuncional
- Publicación en GitHub de LinkedinCorporation. https://github.com/linkedin/qark

- Publicación en página especializada Php
  - ✓ http://php.net/manual/es/security.database.sql-injection.php
- Publicación en página especializada Sqlite - http://www.sqlite.org/about.html

- Publicación en Segu.Info - https://blog.segu-info.com.ar/2015/03/owasp-mobile-security-espanol.html
- Publicaciòn especializada de DiarioTi.com
  - ✓ https://diarioti.com/el-50-de-los-grandes-desarrolladores-de-apps-moviles-no-invierte-en-seguridad/86664
- Rafael Vindel Amor; Publicación en Adictos al trabajo
  - ✓ https://www.adictosaltrabajo.com/tutoriales/introduccion-a-owasp/

- Responsabilidad Social Empresaria desde la Informática
  - ✓ https://www.eleconomista.com.mx/empresas/Seguridad-informatica-como-responsabilidad-20150406-0135.html
  - ✓ https://es.slideshare.net/avanet/la-responsabilidad-social-de-la-ingeniera-de-software
- Richard Stallman; Artículo en GNU - https://www.gnu.org/education/edu-schools.es.html
- Slideshare.net – Desarrollo iterativo e Incremental
  - ✓ https://image.slidesharecdn.com/desarrolloiterativoeincremental-120829050505-phpapp02/95/desarrollo-iterativo-e-incremental-6-728.jpg?cb=1346216786
- Publicación en GitHub de Dinesh Shetty: https://github.com/dineshshetty/Android-InsecureBankv2

- Publicacion de MWR Labs https://labs.mwrinfosecurity.com/

8. Anexos

Anexo 1

Ingeniería inversa con *Apktool*[66]

Un archivo *apk* es simplemente un archivo *zip* que contiene recursos y ensablado*java*. Si simplemente desempaquetamos el archivo con unzip observaríamos lo siguiente:

```
$ unzip testapp.apk
Archive:  testapp.apk
 inflating: AndroidManifest.xml
 inflating: classes.dex
 extracting: res/drawable-hdpi/ic_launcher.png
 inflating: res/xml/literals.xml
 inflating: res/xml/references.xml
 extracting: resources.arsc
```

Si tratamos de leer el archivo *AndroidManifest.xml* observaremos lo siguiente:

```
P□4F□□0□\□f□n□□versionCode□versionName□android*http://schemas.android.com/apk/res/android□package□platfor
mBuildVersionCode□platformBuildVersionName□manifest□brut.apktool.testapp□1.0□21□APKTOO
L□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

Obviamente, editar o ver un archivo compilado es imposible. Aquí es donde entra en juego *Apktool*:

```
$ apktool d testapp.apk
I: Using Apktool 2.0.0 on testapp.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: 1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
$
```

Observando *AndroidManifest.xml* nuevamente, observamos que el archivo es más legible para seres humanos:

[66] Publicación de Apktool - https://ibotpeaches.github.io/Apktool/documentation/

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="https://schemas.android.com/apk/res/android" package="brut.apktool.testapp"
platformBuildVersionCode="21" platformBuildVersionName="APKTOOL" />
```

Ademas de los archivos *XML*, recursos como imágenes, *layout*, string y otros recursos se decodifican correctamente.