METODOLOGÍA DE TESTING DE SEGURIDAD PARA APLICACIONES MÓVILES ANDROID, EN EL CAMPO DE LA SALUD

Hermes Duvier Gaviria Pulgarín

Jhon Fernando Carmona

Programa Académico

Ingeniería de Sistemas

Director del trabajo de grado

Gabriel Taborda

INSTITUTO TECNOLÓGICO METROPOLITANO

OCTUBRE 2018

RESUMEN

El sorprendente incremento en la venta de la telefonía celular y el uso de aplicaciones móviles en los últimos años, han aumentado los riesgos que tienen los diversos tipos información y los datos en el mundo digital. En el mundo digital garantizar la seguridad de la información es muy importante, pero en el contexto de la salud el aseguramiento de la información es crítico; puesto que la vulneración de los datos privados de los usuarios (datos clínicos, enfermedades, etc.), podría provocar daños irreparables a la reputación e imagen de los mismos, así como dar cabida a la posibilidad de extorsiones y amenazas si se diera un uso indebido de esta información.

En la actualidad, son pocos los estudios en Colombia dedicados a cuestiones relacionadas con las aplicaciones móviles en el área de la salud, aun cuando se reconoce el beneficio de las mismas para los usuarios inmersos en dicho entorno. No obstante, para el correcto desarrollo de estas tecnologías es necesario la realización de pruebas claras, a partir de las que se pueda prescindir de las vulnerabilidades de seguridad de la información clínica y los datos, los cuales son considerados como críticos.

Por lo anterior, la finalidad de este proyecto es proponer una metodología de testing de seguridad en aplicaciones móviles para Android en el campo de la salud (mHealth), a través de una revisión bibliográfica sobre el tema, con el fin de obtener los aportes más concretos con los cuales se pueda realizar dicha tarea.

Palabras clave: Aplicaciones Móviles, Metodología de Seguridad, mHealth, Testing de seguridad, Aplicaciones en la Salud, Android.

RECONOCIMIENTOS

ACRÓNIMOS

FDA	Administración de Medicamentos y Alimentos
SO	Sistema Operativo.
FCC	Comisión Federal de Comunicaciones de EEUU
AMPS	Sistema Telefónico Móvil Avanzado
TACS	Sistema de Comunicaciones de Acceso Total
ETACS	Sistema de Comunicaciones de Acceso Total Extendido
GSM	Sistema Global para las Comunicaciones Móviles
CDM	Acceso Múltiple por Versión de Código
TDMA	Acceso Múltiple por División de Tiempo
DANE	Departamento Administrativo Nacional de Estadística
MinTIC	Ministerio de Tecnologías de la Información y las Comunicaciones
OHA	Alianza para los Dispositivos Móviles Abiertos
SDK	Kit de Desarrollo de Software
APK	Aplicación Empaquetada de Android
HTML	Lenguaje de Marcas de Hipertexto
URL	Localizador Uniforme de Recursos
WEB	Red Informática Mundial
OWASP	Proyecto Abierto de Seguridad de Aplicaciones Web
OSSTMM	Manual de la Metodología Abierta de Evaluación de Seguridad
ISSAF	Marco de Evaluación de Seguridad de Sistemas de Información
CEH	Certificación Ética Hacker
MASVS	Estándar de Verificación de Seguridad en Aplicaciones Móviles
IPC	Comunicación Entre Procesos

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Generalidades

Las aplicaciones móviles en el campo de la salud; denominadas mHealth, se definen a partir de “un término que se utiliza para referirse a la práctica de la medicina y la salud pública con el apoyo de dispositivos móviles” (Alonso-Arévalo, 2016, pág. 3), a su vez se deriva de la eHealth (procesamiento electrónico de la salud), pero mientras que el segundo se centra más en las instalaciones de computación fijos (por ejemplo, ordenadores de sobremesa), el primero tiene como objetivo explorar más intensamente los avances en la comunicación inalámbrica, computación ubicua y tecnologías de dispositivos portátiles (Iwaya, 2014, pág. 18). La implementación en la práctica médica de estos desarrollos tecnológicos produce beneficios significativos tanto para los proveedores como para los usuarios, pues las tecnologías de comunicación móvil permiten la comunicación inmediata independiente de las circunstancias de tiempo y lugar, teniendo beneficios directos en la prestación del servicio, por ejemplo, en la ampliación de la cobertura.

La expansión de la tecnología en el siglo XXI ha comenzado a desempeñar un papel relevante en el área de la salud, permitiendo que médicos, enfermeros y otros profesionales del campo puedan acceder a dispositivos y aplicaciones que les permitan ejecutar sus tareas con más facilidad y rapidez. La diversidad de herramientas mHealth incluyen el uso de dispositivos móviles en la recogida de datos de la comunidad y de salud clínica, la entrega y acceso a información de salud para los profesionales, los investigadores, y pacientes, el seguimiento en tiempo real de los pacientes, y la provisión directa de atención a través de la telemedicina móvil (Alonso-Arévalo, 2016, pág. 3).

Así mismo los usuarios interactúan con la información que provee la aplicación sobre el control de la salud, prevención e información de determinadas enfermedades.

Dos factores han sido interpretados como una oportunidad para el desarrollo de aplicaciones mHealth; por un lado, el constante incremento en los últimos años en el uso de telefonía celular; y, por otro lado, la necesidad de aumentar la cobertura de los servicios de salud, estos dos factores combinados han generado un incremento significativo en la provisión de este tipo de aplicaciones. En los países en vías de desarrollo el despliegue de soluciones de mHealth es particularmente prometedor (…), [puesto] que las autoridades de salud pueden aprovechar el floreciente mercado móvil para brindar atención médica adecuada a las comunidades sin servicio o desatendidas (Iwaya, 2014, pág. 19).

Pero en la práctica, a la vez que se registra este incremento, también se registra el de las amenazas a este tipo de aplicaciones.

Una preocupación importante se refiere a la seguridad, aunque los datos médicos suelen estar sujetos a una legislación muy estricta con el objetivo de prevenir el uso o divulgación no autorizados. Sin embargo, muchas propuestas de mHealth no emplean soluciones de seguridad robustas para cumplir con dichas leyes, lo que dificulta su capacidad de convertirse en implementaciones reales (Iwaya, 2014, pág. 20).

Esto es preocupante puesto que, en el campo de la salud, la vulneración de los datos privados de los usuarios podría provocar daños irreparables en la reputación e imagen de los sujetos, así como dar cabida a posibles extorsiones y amenazas en caso de que se hiciera un uso indebido de la información suministrada en la aplicación móvil.

En la actualidad, son pocos los estudios en Colombia dedicados a cuestiones relacionadas con las aplicaciones móviles en el área de la salud, aun cuando se reconoce el beneficio que están en capacidad de generar en los usuarios, en cuanto mejora de la calidad de vida, disminución de costos e incremento en el acceso a los servicios de salud. No obstante, las herramientas mHealth

> …puede[n] convertirse en un importante medio para proporcionar un mayor acceso a segmentos más amplios de la población en estos entornos, así como la mejora de la capacidad de los sistemas de salud (…) para proporcionar una atención médica de mejor calidad, mejora en la posibilidad de diagnosticar y hacer un seguimiento de enfermedades, informar y formar en salud a los ciudadanos y dotar de una mejor formación para los trabajadores de salud. (Alonso-Arévalo, 2016, pág. 3)

Sin embargo, para el correcto desarrollo de estas tecnologías es necesario la realización de pruebas asertivas, a través de las que se eviten los riesgos mencionados en referencia a la seguridad y privacidad de la información, más aún en este campo, en el que los datos son considerados como críticos.

A pesar de la evolución de los dispositivos móviles y el uso masivo que se le da en todo el mundo, el aumento de aplicaciones relacionadas con el campo de la salud y sus beneficios demostrados, también se evidencia el crecimiento de las posibles amenazas a la seguridad y la privacidad de los usuarios; “en Estados Unidos, el U.S. Department of Health and Human Services Food and Drug Administration (FDA) reconoce (…), el rápido ritmo de la innovación en las aplicaciones móviles, así como los beneficios y los riesgos potenciales para la salud pública” (Alonso-Arévalo, 2016, pág. 4), que pueden representar estas aplicaciones. En este mismo sentido, los usuarios dan muestras de desconfianza en la autenticidad de la información que proveen las aplicaciones, y en la confianza con la que suministran información de su vida íntima y datos en relación con su vida privada. Debido a ello, los beneficios que están en capacidad de proveer las aplicaciones mHealth para el campo de la salud, tanto para usuarios como para profesionales del sistema sanitario, no han aumentado significativamente por causa de la precaria confianza que se tiene sobre las mismas.

De acuerdo a lo anterior, es evidente como en las aplicaciones móviles de la salud, donde no solo se ingresa información general, sino personal y privada, existe la necesidad de crear herramientas que potencien la seguridad de los datos.

Una de ellas es la ejecución de una metodología de testing de seguridad centrada en las aplicaciones móviles de la salud, la cual proveería una ruta específica y válida para mitigar o disminuir el riesgo de seguridad de dichas aplicaciones, y proteger los datos de la salud considerados actualmente como críticos.

1.2. Objetivos

General

Proponer una metodología de testing de seguridad para aplicaciones móviles Android, en el campo de la salud.

Específicos

- Realizar una revisión bibliográfica sobre la seguridad en aplicaciones móviles, con énfasis en el área de la salud.
- Categorizar las diferentes metodologías dedicadas al testing de seguridad de aplicaciones moviles.
- Clasificar las caracteristicas mas relevantes en referencia a la proteccion de los datos que deben poseer las aplicaciones moviles en el campo de la salud.
- Definir una metodologia de testing de seguridad para aplicaciones moviles dedicas a la salud con sistema operativo android.
- Construir los test de seguridad empleando la metodologia propuesta.

1.3. Organización de la tesis

Para dar cuenta de los objetivos propuestos el trabajo de investigación se organizó de forma estructurada y por capítulos.

En el primer capítulo denominado: Introducción, se presenta el tema del trabajo, los objetivos que se trazó y la estructura del trabajo.

En el segundo capítulo denominado: Desarrollo de las aplicaciones Android en salud, se realizó una descripción del desarrollo de la telefonía móvil y los sistemas

operativos (SO) que permiten controlar los dispositivos móviles; posteriormente se hizo énfasis en el sistema operativo Android y se presentó información general sobre las aplicaciones móviles, haciendo énfasis en las aplicaciones Android, y posteriormente se realizó un desarrollo conceptual sobre las categorías y características de las aplicaciones mHealth.

El tercer capítulo denominado: Metodología de testing de seguridad para aplicaciones móviles Android en salud, establece los elementos teóricos y conceptuales que nos permiten entender las metodologías de testing de seguridad, y se hizo énfasis en las que hablan específicamente sobre la plataforma Android. Este capítulo finaliza con la propuesta de metodología de gestión de seguridad para aplicaciones mHealth

El capítulo cuarto denominado: Metodología presenta las fases, técnicas e instrumentos a través de los que se realizó el trabajo de investigación.

En el capítulo quinto denominado: denominado: Resultados y Discusión, se presenta el test de seguridad para aplicaciones mHealth para el Sistema Operativo Android construido a partir de la metodología que se propuso el capítulo tercero.

En el capítulo sexto denominado: Conclusiones y recomendaciones, se hace una breve reflexión sobre los principales aprendizajes del trabajo de grado y se hacen algunas sugerencias que se derivan del proceso de investigación.

2. DESARROLLO DE LAS APLICACIONES ANDROID EN SALUD

Este capítulo describe el proceso de desarrollo de los dispositivos e innovaciones necesarias para que fuera posible implementar aplicaciones móviles en el campo de la salud o mHealth. En primer lugar, se hace una descripción del desarrollo de la telefonía móvil y los sistemas operativos (SO) que permiten controlar los dispositivos móviles; posteriormente se hace énfasis en el sistema operativo Android y se presenta información general sobre las aplicaciones móviles, haciendo énfasis en las aplicaciones Android y finalmente se realiza un estado del arte de las aplicaciones móviles en el campo de la salud.

2.1. Breve Historia de la Telefonía Móvil

El contexto socioeconómico de la globalización y la sociedad del conocimiento, implica el desarrollo sostenido de Tecnologías de la Información y Comunicación, y el auge masivo de intercambios de información, los cuales requieren desarrollos tecnológicos constantes para mantener el flujo de información, que es el medio y el motor que impulsa los intercambios de la aldea global. El desarrollo de este contexto socio-económico-cultural, es el ámbito en que tiene nacimiento la telefonía móvil.

Cuando se habla de teléfonos móviles se hace referencia a dispositivos electrónicos que llegan al mercado con la capacidad de ser inalámbricos o portables, es decir, que no necesitan de ninguna conexión por medio de un cable para su funcionamiento, y que son capaces de recibir y realizar llamadas desde cualquier lugar. Estos teléfonos móviles poseen además otras características que aumentan su usabilidad, que los hacen más útiles para los usuarios, debido a que cuentan con aplicaciones, entre las que se encuentran agendas, juegos o mensajería

instantánea, entre otras. Pero para que un dispositivo así pudiera existir debieron darse algunos desarrollos tecnológicos.

Dos grandes invenciones del siglo XIX fueron necesarias; por un lado, el teléfono, aunque popularmente se atribuye su invención a Bell, en realidad fue inventado “por el médico italiano Antonio Meucci” (Ruesca, 2016, párr. 5) en 1849; y por otro lado, la comunicación inalámbrica por ondas de frecuencia, que, como en el caso del teléfono, también se le atribuye a otra persona; la radio fue desarrollado por Nicola Tesla; pero “el italiano Guglielmo Marconi sobre distintos proyectos patentados por el austro-húngaro Nikola Tesla desarrolla la radio” (Ruesca, 2016, párr. 9), tras varias acciones legales interpuestas por Tesla, se resuelve otorgarle la patente de la radio en 1943.

Otro de los desarrollos necesarios para la invención de la telefonía móvil fue el de los radioteléfonos y los handies, esta tecnología requería una antena dentro del rango de transmisión para llevar a cabo la comunicación. En 1947 la empresa AT&T intentó adaptar esta tecnología para instalar teléfonos a los automóviles, pero la Comisión Federal de Comunicaciones de EEUU (FCC) “no aprobó la propuesta, y esto retrasó las investigaciones y estudios, dado que no existía interés en las empresas de investigar en un proyecto que no les podría ser redituable económicamente” (Ruesca, 2016, párr. 13). Sin importar esta circunstancia AT&T propuso un modelo de comunicación celular que fue el precedente directo de la telefonía móvil; este sistema “propone la descentralización de las comunicaciones, y un esquema de células base. Cada célula consta de una antena transmisora y el equipo tecnológico para que envíe y reciba ondas de radio” (Ruesca, 2016, párr. 14), este sistema descentralizado es primordial para el desarrollo de los equipos móviles, puesto que es éste el que permite que cuando el equipo sale de cobertura de una célula, la llamada pase a otra, permitiendo la capacidad de desplazarse sin

interrumpir la llamada, y la descentralización de las antenas (Ver Figura 1).

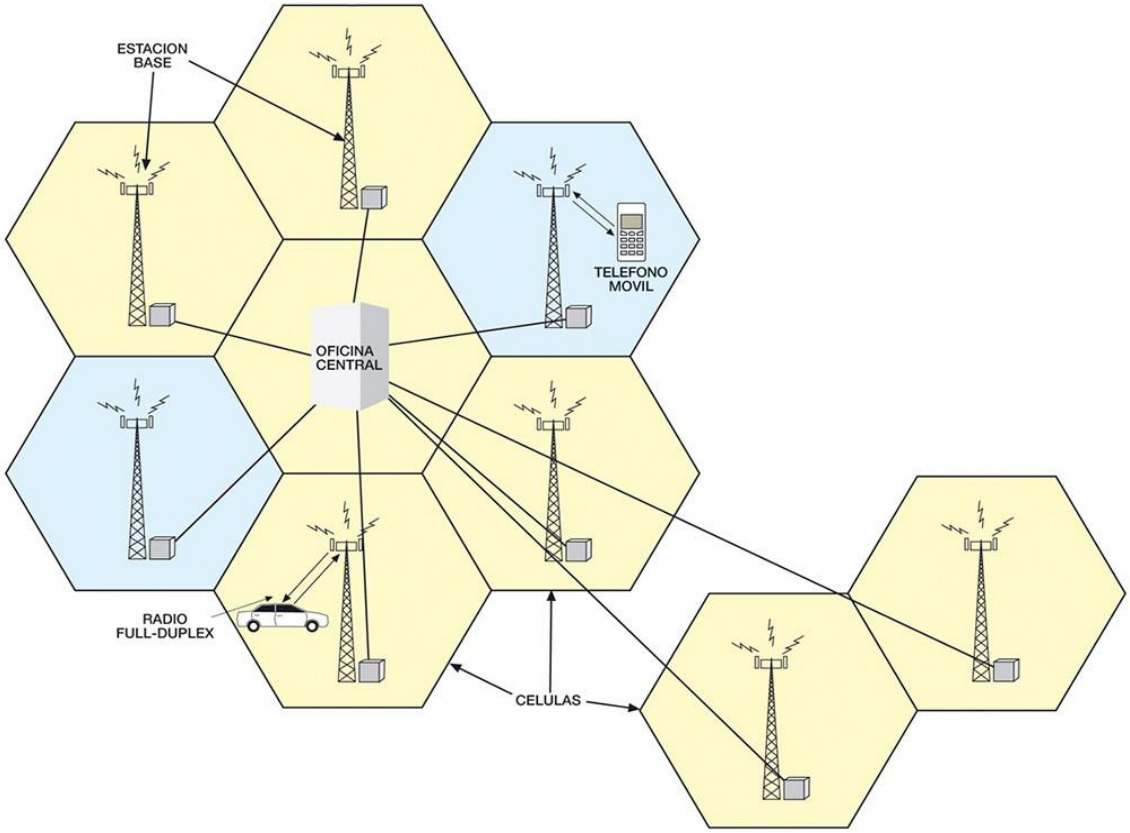


Figura 1 Esquema de comunicación a partir de células

Fuente: https://cursodecelulares.files.wordpress.com/2011/04/celulas.jpg

A partir de estas primeras innovaciones, en 1977 AT&T en asocio con Bell, desarrollaron un prototipo de comunicación móvil; tras lo cual en 1982 la FCC autorizó el uso comercial de las frecuencias de radio, todo esto permitió que "Ameritech [pusiera] a disposición en Chicago el primer sistema analógico de telefonía móvil celular de uso comercial" (Ruesca, 2016, párr. 16).

Los primeros dispositivos móviles que surgieron, eran de gran tamaño y bastante costosos y solo una parte de la población podía acceder a los mismos. "el nacimiento de la telefonía móvil, fue en 1973, cuando el directivo de Motorola Martin Cooper realizó la primera llamada con un DynaTAC 8000X" (Iglesias et al. 2017, pág. 227); un teléfono móvil que pesaba 794 gramos y medía 33 x 4,5 x 8,9 centímetros, a un costo de US$ 3995. (Ver Figura 2).

Figura 2 Primer teléfono celular Motorola DynaTAC 8000X
(Fuente: https://media.novinky.cz/840/52840-top_foto1-p8x2n.jpg?1452268805)

Tras una buena recepción por parte de los consumidores, y los desarrollos técnicos que se dieron en las décadas de 1980 y 1990, se "desarrollan distintos estándares de comunicación celular, y surgen los teléfonos móviles de segunda generación (2G), equipos digitales frente a los teléfonos de primera generación (1G) que eran analógicos" (Ruesca, 2016, párr. 18), la mejora que reportó la aparición de los equipos 2G fue la ampliación de frecuencias y de la banda de frecuencia en la cual pueden funcionar estos sistemas. La aparición de los dispositivos 3G, fundamentales para el desarrollo de aplicaciones, se dio en Japón, éstos son indispensables, puesto que "además de la transmisión de datos y voz (una llamada) también permiten la transmisión de datos no-voz (acceso al email, descarga de programas, etc.)" (Ruesca, 2016, párr. 19).

López (2013) describe dichas generaciones indicando las características más comunes que prevalecieron en cada una de ellas: (Ver Tabla 1)

Tabla 1 Algunas características de los móviles según su Generación (G)

Tipo de G	Características
1G	Los equipos 1G funcionaban por medio de tecnología analógica, a través de AMPS (Sistema Telefónico Móvil Avanzado): primer estándar para redes de dispositivos móviles. Otros teléfonos utilizaban TACS (Sistema de Comunicaciones de Acceso Total), el cual fue muy utilizado por países europeos y asiáticos como Inglaterra, Hong Kong y Japón. Posteriormente se dio uso de ETACS (Sistema de Comunicaciones de Acceso Total Extendido) creado por Reino Unido como una mejora de TACS;

2G	Los equipos 2G, pasaron de tecnología analógica a digital y utilizaron estándares como GSM (Sistema Global para las Comunicaciones Móviles), el más utilizado en Europa, CDM (Acceso Múltiple por versión de Código) la cual posibilitó que se pudiera transmitir a gran distancia una señal de radio y TDMA (Acceso Múltiple por División de Tiempo) permite transferir información digital y de voz pequeños;
3G	Los equipos 3G, se desarrollaron gracias a la intervención del IMT 2000 (Telecomunicaciones Móviles Internacionales para el año 2000), disponen de redes de alta velocidad de transmisión de datos, con compatibilidad de los servicios móviles de 3G con las redes de la segunda generación 2G
4G	Los equipos 4G, la generación actual, y hasta ahora la más avanzada, posibilita al usuario disfrutar de un sin número de beneficios como mejora de banda ancha, la transmisión digital de datos, entre otros.

Fuente: Datos tomados de (López, 2013, pág. 2). (Elaboración propia)

Una de las consecuencias más significativas de la incursión de las tecnologías 4G, es que en la medida en que las capacidades y las funcionalidades de los dispositivos móviles aumentan, la innovación y la incorporación de nuevas aplicaciones que permiten mayor variedad de interacciones con los teléfonos inteligentes generan mayores presiones en el mercado y atractivos para los usuarios, lo cual tiene como resultado que las personas requieran y se adapten con mayor facilidad a este tipo de tecnologías, por lo cual es de esperar que el uso del teléfono inteligente sea cada vez más frecuente entre la población.

Según el diario The Independent, para el año 2014, “oficialmente [habían] más dispositivos móviles que personas en el mundo. [Dicho diario argumenta que], el mundo alberga 7.200 millones de dispositivos, [y que estos] se multiplican cinco veces más rápido que nosotros” (Brick&Mobile, 2018); también a nivel global, se estima que más de un tercio de las personas en todo el mundo, es decir aproximadamente “2.560 millones de personas serán propietarios de smartphones. Esa cifra, (…) de 2018, representa más de la mitad (51.7%) de todos los usuarios de teléfonos móviles” (Brick&Mobile, 2018); otra cifra que nos puede dar un interesante contexto sobre la penetración de la tecnología móvil en el planeta, es que: “el 60% de los consumidores móviles globales usan sus dispositivos móviles como su fuente principal o exclusiva de Internet” (Brick&Mobile, 2018), en este mismo sentido, según estudios realizados, se espera que para el año 2020; “24 mil

millones de dispositivos estarán conectados a Internet [y que] la gran mayoría usará algún tipo de conexión inalámbrica para acceder" (Brick&Mobile, 2018).

Según proyecciones del Dane 2016-2017 y el Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC), en Colombia, en el tercer trimestre de 2017, se verifica la tendencia a nivel internacional según la cual se registran mayor cantidad de líneas telefónicas que población, según estas cifras, Colombia tiene un total de 60.783.374 líneas activas (2018, pág. 35). (Ver Figura 3)

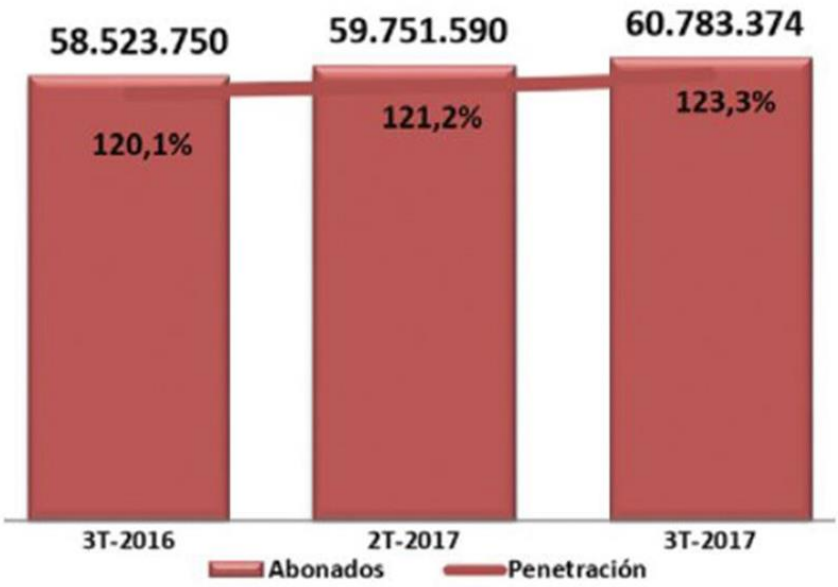


Figura 3 Abonados de telefonía móvil e índice de penetración

Fuente: Proyecto de población Dane 2016-2017 y datos reportados por los proveedores de redes y servicios a Colombia TIC. Fecha de consulta: 27 de diciembre de 2017.

Según la encuesta anual del consumo móvil, realizado por la firma Deloitte, en Colombia, de la gran variedad de dispositivos electrónicos que están en capacidad de reproducir contenidos, es más frecuente que las personas dispongan de teléfonos inteligentes, lo cual es bastante significativo, puesto que si nos atenemos a las cifras (Ver figura 4), es mucho más frecuente que la población colombiana cuente con un smartphone, que por ejemplo con un computador portátil o uno

escritorio (Deloitte, 2018, pág. 11). Lo cual hace que el mercado de las aplicaciones móviles en Colombia sea un mercado que puede tener bastante expansión.

Figura 4 ¿Cuál dispositivo posee o tiene acceso inmediato?

Fuente: Deloitte (2018)

2.2. Sistemas Operativos para Dispositivos Móviles

En general un sistema operativo (OS) es un programa que después de ser cargado por un programa de arranque, controla todos los otros programas que existen en el equipo; en el caso de los dispositivos móviles, se encarga de controlar el celular, a la vez que facilita la correcta interacción entre el usuario y las aplicaciones que posee el teléfono. El Sistema Operativo (SO) es el programa principal de cualquier dispositivo móvil, puesto que es el encargado de que todos los subsistemas (Aplicaciones, tiempo prudente para la ejecución de tareas, velocidad y demás) funcionen adecuadamente, permitiendo que el usuario pueda trabajar con cada herramienta del teléfono sin problemas (Polanco & Beauperthuy, 2011). En palabras más técnicas, el sistema operativo:

Es una capa compleja entre el hardware y el usuario, concebible también como una máquina virtual, que facilita al usuario o al programador las herramientas e interfaces adecuadas para realizar sus tareas informáticas, abstrayéndole de los complicados procesos necesarios para llevarlas a cabo. Podemos deducir que el uso de uno u otro SO determinarán las capacidades multimedia de los dispositivos, y la forma de éstas de interactuar con el usuario (Baz et al., 2011).

Los celulares o teléfonos móviles han experimentado numerosos cambios en los últimos años. El avance de la tecnología en relación con las necesidades del ser humano ha propiciado que un teléfono celular haya pasado de tener una única función específica, como lo es la comunicación, a poseer muchas otras funciones, hasta el intento incluso de querer igualarse a las capacidades y herramientas que provee una computadora. Esta capacidad la tienen gracias a las posibilidades que les brindan los diferentes sistemas operativos; entre los sistemas operativos actualmente más utilizados se encuentran: iOS, Windows Phone, BlackBerry OS y Android. En la siguiente Tabla (Ver Tabla 2), destacamos algunas características de estos Sistemas Operativos, pero posteriormente hacemos énfasis en la plataforma Android, pues es ella la que tiene mayor interés para este trabajo de investigación.

Tabla 2 Sistemas Operativos (SO) más comunes para dispositivos móviles

SO	Logo	Ranking	Características
Android		1°	Diseñado para dispositivos móviles, smartphones y tabletas por Android Inc. fue posteriormente adquirido por Google. Basado en Linux, es un SO libre, gratuito y multiplataforma
iOS		2°	Sistema Operativo exclusivo de la empresa Apple; funciona con iPhone, Ipod Touch e iPad. Es un derivado de Mac OS X y una interfaz intuitiva, con buena funcionalidad y sólida estabilidad. Una de sus desventajas es la de la compatibilidad de las aplicaciones.
Windows Phone		3°	Anteriormente Windows Mobile, fue desarrollado por Microsoft para smartphones, tabletas y otros dispositivos de esa marca, de apariencia similar a equipos de escritorio o

			portátiles, su interfaz es intuitiva y muy amable.
BlackBerry OS		4°	Es el SO diseñado por RIM para la marca BlackBerry y para uso en computadoras de mano, como: trackwheel, trackball, touchpad y pantallas táctiles. Fue uno de los primeros en hacer presencia en el mercado, pero perdió su dominio y ahora su presencia en el mercado es mínima.

Fuente: (Elaboración Propia)

2.3. Historia del Sistema Operativo Android

Son varios los hitos que es necesario destacar para entender el desarrollo del sistema operativo Android; el primero de ellos evidentemente es su fundación, la empresa Android Inc. fue fundada en el año 2003 por “Andy Rubin, Rich Miner, Nick Sears y Chris White” (Betancur & Eraso, 2015, pág. 18), Andy Rubin había participado en varios proyectos anteriores sin mayor éxito en sus esfuerzos.

Otro hito importante es que en el año 2005, Google adquiere la empresa Android Inc. con el objetivo de iniciar un proyecto de desarrollo de un sistema operativo más funcional que el de sus competidores, y para dar comienzo a dicho plan en el año de 2007 se reúne con empresas importantes de tecnología reconocidas como Toshiba, Intel, Motorola, Samsung, entre otros, con el fin de emplear nuevos instrumentos más innovadores y adecuados a las necesidades actuales y deciden formar una alianza que fue denominada como Open Handset Alliance (OHA) (Aponte & Dávila, 2011).

De esta forma el SO Android se constituye como “un entorno de software integrado para dispositivos móviles que incluye un sistema operativo, un middleware y un conjunto de aplicaciones bases” (Rodríguez, 2010, pág. 33), que enriquecido por los aportes de los grupos de investigación de la OHA se transforma en lo que fue llamado en su primera versión “SDK Android” en noviembre del año 2007.

Posteriormente, en septiembre del 2008, se anuncia la salida al mercado del primer smartphone con este sistema operativo, y en el mismo año, Google crea el primer

portátil que contiene aplicaciones de Android. A partir de allí, Google sigue mejorando la plataforma, corrigiendo errores y produciendo nuevas versiones enfocadas en la mejora de todos los aspectos de los dispositivos móviles" (Aponte & Dávila, 2011).

El SO Android dispone de una interfaz de programación Java y es "el principal producto de la Open Handset Alliance, está basado en Linux, [tiene] un núcleo del Sistema Operativo libre, gratuito y multiplataforma, la mayoría del código fuente de Android ha sido liberado bajo la licencia de Software Apache" (Largo, 2016, pág. 3), es decir, que el licenciamiento permite al usuario del software la libertad de usar el software para cualquier propósito, para distribuirlo, modificarlo y distribuir versiones modificadas, una de sus particularidades más importantes es que es libre y de código abierto, peculiaridad que permite al programador crear nuevas aplicaciones o incluso transformar su propio sistema operativo.

2.4. Participación de Android en el Mercado

El auge que la plataforma Android ha tenido en los últimos años en el mercado es evidente y es notablemente el incremento del sistema operativo en los dispositivos móviles, ya que "cuatro de cada cinco terminales comprados durante el segundo trimestre del 2016 poseen el sistema operativo Android" (Tabares, 2016, p. 16). Así, se evidencia el aumento en compra y venta de dispositivos móviles con dicha plataforma, e incluso su posición principal entre los demás. Según datos de la web Statista, el dominio de Android del mercado ha sido claro desde 2012 y se ha profundizado a través del tiempo, ya para 2016 Android representaba 84.8% de los SO instalados, seguido de muy de lejos por iOS con una participación del 14,4% (Ver Figura 5).

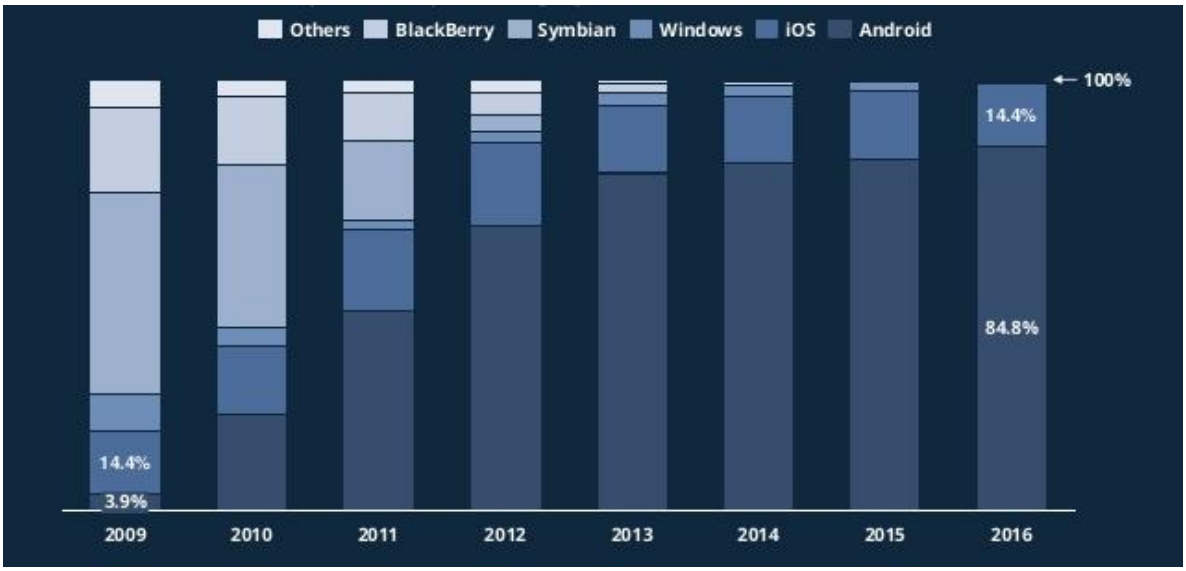


Figura 5 Evolución de la participación de Android en el mercado

Fuente:https://image.slidesharecdn.com/statistadigitaleconomycompass2017-170426105732/95/statista-digital-economy-compass-2017-13-638.jpg?cb=1496749226

El dominio de los SO Android en el mercado se debe en parte a la estrategia de mejora continua de la empresa, cada una de las versiones del SO es una versión mejorada del anterior. Una de las peculiaridades del SO, es que cada versión estable de Android, lleva el nombre de un postre; así, la versión de 2008 fue denominada Apple Pie (Alpha en la Figura 6), la primera de 2009 Banana Bread (Beta en la Figura), y así hasta la última versión lanzada en agosto de 2017 denominada Oreo. La justificación de los nombres de las versiones de Android, es que según la empresa, los SO Android nos hacen la vida más dulce, de ahí que tengan nombres de postres y dulces.

Figura 6 Nombres de versiones del SO Android

Fuente: https://medium.com/@firatkarababa/the-green-revolution-a un-look-on-the-history-of-android-30dbccc7afa9

Tal y como se evidencia en la gráfica de la web Statista (Ver Figura 7), la versión del SO Android más usada en el mundo es la Nougat, nombre que hace alusión a una especie de turrón, esta versión lanzada en agosto de 2016 está instalada en el 31,1% de los dispositivos controlados por el SO de Google. Para este trabajo éste no es un dato menor, puesto que al ser Nougat la versión que esta dominando el mercado, los criterios de seguridad de los que se hablan en este trabajo de investigación se refieren a la versión Nougat.

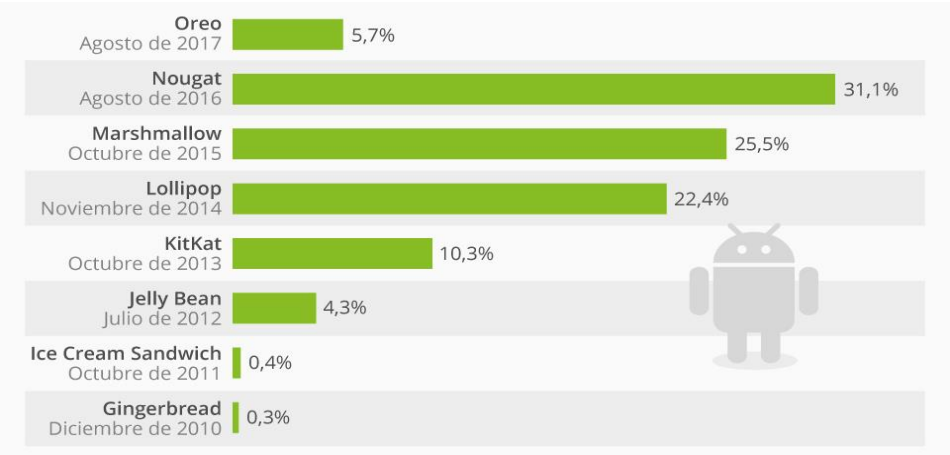


Figura 7 Versiones de Android más usadas

Fuente: https://es.statista.com/grafico/13861/nougat-todavia-la-version-de-android-mas-utilizada/

2.5. Características del SO Android Nougat

El SO Android es un sistema jerarquizado por capas, en el que los niveles más bajos se encargan de la interacción con el hardware y los más altos con los usuarios. La arquitectura del sistema operativo Android está basada en cuatro niveles: Kerner Linux, bibliotecas o librerías, framework o el armazón de aplicaciones y la capa de aplicaciones (Ver Figura 8).

Figura 8 Arquitectura del SO Android

Fuente: (OWASP, 2018)

El Kerner Linux tiene como función gestionar o controlar el hardware, la red, el acceso al sistema de archivos y la gestión de procesos; en general, las cuestiones más elementales del sistema, como la operación de drivers o funciones de la seguridad. El núcleo Kernel es el encargado de intermediar entre el software y el hardware y viceversa, de esta forma cuando requerimos hacer una llamada, ajustar el brillo de pantalla o poner una canción el software hace una petición al Kernel, y éste, por medio de los controladores cumple el requerimiento.

Las bibliotecas o librerías son el componente encargado de cumplir los servicios asociados a la capa de aplicaciones y cada una de ellas cumple una tarea específica; entre ellas se cuenta: Qlite necesaria para compilar datos, la OpenGL para el uso de gráficos en tercera dimensión, Webkit para navegar en internet, la biblioteca FreeType que trabaja con diferentes tipos de fuentes, entre otras.

La capa Framework o armazón de aplicaciones, proporciona servicios para el desarrollador de aplicaciones, en esta capa se encuentran aplicaciones Java destinadas entre otros fines, a interactuar con el hardware del dispositivo, y tener acceso a las bases de datos, además esta capa tiene a su cargo diferentes funciones, como la entrada a los datos de mensajería, la gestión telefónica o de las aplicaciones, etc.

La capa de aplicaciones, es el nivel más alto de la arquitectura del dispositivo Android y la conforman todas las aplicaciones con las que el usuario interactúa directamente, entre ellas se cuentan, la aplicación de mensajes de texto, el navegador de internet, los contactos, el calendario, entre otros. Todas estas aplicaciones se ejecutan con Java.

La versión de Android 7.0 o Nougat, fue lanzado el 22 de agosto de 2016 y a la fecha es el SO más usado en dispositivos móviles, Google hizo el lanzamiento de este SO para dispositivos de su marca como el “Nexus 5X, Nexus 6, Nexus 6P, Pixel C y Nexus 9” (Guevara, 2016, pág. 2), según algunas revisiones publicadas en revistas especializadas este sistema operativo es el más rápido y fluido de la familia Android, “Google introduce ahora más de 250 prestaciones nuevas, la mayoría a nivel de software, no visibles para el usuario” (Guevara, 2016, pág. 2-3). Una de las particularidades de esta versión es una funcionalidad a través de la cual se optimiza el consumo de memoria RAM lo que evita que la capa de aplicaciones y las actividades en segundo plano sature las funciones del sistema operativo, además de aumentar el rendimiento de la batería. Otra de las mejoras importantes del SO Nougat es la posibilidad de tener dos aplicaciones abiertas en la misma pantalla lo cual mejora la experiencia de un SO multitarea como lo es Android

2.6. Aplicaciones Móviles

Una aplicación móvil o App, como comúnmente se le denomina, es “un software diseñado para funcionar en teléfonos inteligentes y otros dispositivos móviles” (San Mauro et al. 2014), generalmente se acepta que no todas las aplicaciones son programas y que no todos los programas aplicaciones; una aplicación es una pieza de software diseñada para cumplir un propósito específico o para facilitar una tarea determinada, mientras que un programa es una pieza de software que se diseña para cumplir una serie de tareas o para suplir un requerimiento más general.

El desarrollo de aplicaciones móviles se ha expandido en los últimos 10 años, y las empresas encargadas de diseñarlas intentan realizar cada vez más mejoras al punto que hay una gran competitividad entre ellas. De esta manera, las aplicaciones y la telefonía móvil se han expandido permanente y los individuos se ven abocados a su uso en la mayoría de los entornos que normalmente frecuentan.

El principal tipo de clasificación de aplicaciones móviles son los tipos, entre ellas se encuentran las nativas, las hibridas y las web, y se extiende a todos las aplicaciones sin importar la empresa que las distribuye, ya sea Google, Apple, Blackberry, etc., un resumen de sus principales características aparece en la siguiente tabla (Ver Tabla 3).

Tabla 3 Tipos de aplicaciones

	Nativas	**Web**	**Hibridas**
Definición	Es una aplicación desarrollada especialmente para un determinado SO, ya sea Android, iOS, Blackberry, etc., no requieren de conexión a la red para desempeñar su función.	También denominadas webapp, son aplicaciones y un pintor desarrolladas con lenguajes de programación en red, como es el HTML, Javascript y CSS. Este tipo de aplicaciones nativas interactúan con sitios web y se ejecutan dentro del propio navegador web del dispositivo a través de una URL.	Este conjunto de apps son una combinación de las anteriores. Se desarrollan con leguajes propios de la aplicación web. Y dan la posibilidad de acceder parte de las características del hardware del dispositivo.

Características	Se descargan e instalan desde las tiendas de aplicaciones. Tienen actualizaciones frecuentes. Usan las aplicaciones del SO para mostrar avisos al usuario. No requieren internet para funcionar. Y disponen de una interfaz basada en las guías de cada sistema operativo.	No necesita instalarse. No requieren realizar actualizaciones. Necesitan conexión a internet para funcionar. Poseen algunas restricciones en factores como gestión de memoria. Tienen una interfaz más genérica e independiente de la apariencia del sistema operativo.	Permiten el acceso a las capacidades del teléfono. Su diseño visual, no se identifica en gran medida con el del SO. Son de uso gratuito
Ventajas	Tienen Acceso completo al dispositivo. Proveen una mejor experiencia al usuario. Son accesibles a través de la tienda de aplicaciones. Generan notificaciones o “avisos” de actividad a los usuarios. Las actualizaciones son constante	Tienen el mismo código base que es reutilizable en varias plataformas. Su proceso de desarrollo es más sencillo y económico. No necesitan aprobación externa para publicarse. El usuario dispone siempre de la última versión.	Es posible acceder a ellas en las tiendas de apps. Su instalación es nativa pero construida con JavaScript, HTML y CSS. Tienen acceso aparte del hardware del dispositivo.
Desventajas	Diferentes habilidades, idiomas, herramientas para cada plataforma de destino. Estas apps tienden a ser más caras de desarrollar. El código del cliente no es reutilizable entre las diferentes plataformas	Requiere de conexión a internet. Tienen acceso limitado a los elementos y características del hardware del usuario y el tiempo de respuesta es menor que en una APP nativa. Requiere un mayor esfuerzo en promoción y visibilidad.	La experiencia del usuario está más asociada a la app WEB que de a app nativa. Su diseño visual no siempre relacionado con el sistema operativo en el que se muestre.

Fuente: (Flórez, 2015, pág. 29-30) (Elaboración propia)

El tipo de desarrollo de aplicaciones, se refiere a la naturaleza del fabricante, es decir, si las aplicaciones fueron diseñadas para el SO del dispositivo o por el contrario fueron diseñadas para ser compatibles con cualquier SO. A partir de la gráfica, pueda inferirse que las aplicaciones nativas son superiores a los otros tipos de aplicaciones ya que ellas se desarrollan en un lenguaje definido por la misma empresa para para un sistema operativo específico, por lo cual resulta más veloces que otras aplicaciones que fueron diseñadas para ser compatibles con varios SO.

2.7. Historia de las Aplicaciones Móviles

En la década de los 1990 aparecen las primeras aplicaciones móviles, éstas eran herramientas básicas, bastante simples en términos de usabilidad y diseño, entre ellas había apps diseñadas para cumplir las funciones más elementales del teléfono como gestores de contactos, editores de ringtones, entre otros.

En el año 2007 la reconocida empresa Apple lanza su teléfono iPhone, a partir de allí, se dio comienzo a un mercado que cada vez está más extendido y que progresivamente presenta mejores y más funcionales aplicaciones. En “el año 2008 (…) se lanzó la App Store de Apple, una tienda virtual de apps donde los desarrolladores podían publicar sus aplicaciones y los usuarios descargarlas” (Herranz, 2017, pág. 7), mediante este programa Apple brinda la posibilidad de obtener aplicaciones externas y es así como Apple permite a los desarrolladores de aplicaciones desde cualquier país, promover su app y las ventas de la misma.

A partir de las innovaciones de Apple, otras empresas descubren un mercado con altísimo potencial, es así como Google con su SO Android emerge el dicho mercado y desarrolla la tienda Play Store, cuya ventaja es que muchas de sus aplicaciones son libres y tienen alta compatibilidad, no así las de su competencia de Apple. Dicha circunstancia conduce a un aumento en el alcance de las aplicaciones Android y contribuye a establecer un claro dominio del mercado de las apps.

Dadas estas circunstancias, lo que comienza con unas pocas aplicaciones con una funcionalidad restringida, termina con millones de aplicaciones que cumplen múltiples funciones que favorecen a los usuarios, e incluso la creación de apps por parte de los usuarios de las mismas, lo que incide positivamente en el crecimiento del mercado que muy probablemente seguirá profundizándose en los próximos años.

2.8. Aplicaciones Móviles para Android

Todas las aplicaciones para dispositivos Android, “están encapsuladas en un formato específico, conocido como APK «Application PacKage File»” (Eraso & Betancur, 2015, pág. 30), el formato APK es el utilizado para instalar y distribuir las aplicaciones de esta plataforma y aunque es un formato exclusivo para SO Android, es compatible con múltiples dispositivos móviles.

El sistema operativo Android clasifica sus aplicaciones en 3 clases: “Primer plano o foregroung, segundo plano o background, y lo que es denominado como widget o app widget” (Gallego, 2014). Las aplicaciones de primer plano, también son denominadas actividades, cada aplicación tiene una actividad principal que se representa en la pantalla a la que tiene acceso el usuario al ser iniciada desde la lista de aplicaciones disponibles; son aplicaciones que requieren una interfaz de usuario para desarrollar las actividades para las que fueron diseñadas; un ejemplo típico de este tipo de apps es el reproductor multimedia. Las aplicaciones de segundo plano son generalmente denominadas como servicios y son aplicaciones que no disponen de interfaz de usuario. Y finalmente, los widgets, son aplicaciones que cuentan con una pequeña interfaz gráfica que se ubica en el escritorio de Android y que se actualiza cada cierto intervalo de tiempo, un ejemplo es el del reloj, el calendario o una aplicación de información climática.

2.9. Aplicaciones móviles en el campo de la salud

Actualmente, se evidencia un creciente uso y promoción de la información mediada por los medios digitales. Los cambios y la rapidez con que se recibe y se emite la información se debe principalmente a la llegada del internet, a la era digital y tecnológica. La expansión de las herramientas digitales y las aplicaciones móviles se extienden a casi todos los aspectos de la vida. Muchos de los cambios experimentados en el contexto de esta expansión son benéficos, un ejemplo fehaciente de esto es el experimentado en el área de la salud, en la que se ha evidenciado un crecimiento exponencial de las herramientas digitales disponibles

para el tratamiento de los pacientes, permitiendo la promoción de ésta por medio de la tecnología.

La proliferación de la tecnología en este campo se demuestra incluso con los mismos profesionales de la salud (médicos y enfermeras) que comparte información por medio de las redes sociales acerca de temas relacionados con la salud física y mental. De la misma manera, gran cantidad de pacientes ven en las redes sociales una forma de compartir sus experiencias respecto a una enfermedad y permitir que otros reciban información y consejos para no recaer en los mismos hábitos y evitar una enfermedad determinada. De esta forma, en el área de la salud se experimenta una nueva época en la que la tecnología permite a los profesionales recurrir a otros espacios y transformar la salud por medio de la creatividad y la actualización (Fernández, 2013).

Una de las herramientas tecnológicas que se destaca actualmente en el campo de la salud son las aplicaciones móviles, las cuales han tenido gran aumento en su utilización en muchas partes de mundo. Notablemente, dichas herramientas ayudan a los usuarios en el sentido que les permite mantenerse informados sobre los procesos médicos que están siguiendo; así mismo a los médicos les posibilita el recurrir a instrumentos para abordar un diagnóstico y estar enterados de su historia clínica, entre otras ventajas (Santamaría, Hernández, 2015).

El auge de las dinámicas tecnológicas en el área de la salud ha conllevado a que se creen incluso términos específicos como la eSalud o eHealth, que resaltan el impacto de la tecnología en este campo. Este término surgió ya hace algún tiempo, en la década de los 90, siendo utilizado actualmente con mucha frecuencia para hacer alusión al uso de equipos de cómputo estacionarios, la tecnología informática y el internet en el sistema sanitario (RCCS, 2016).

Otra de las nociones actualmente en boga en esta área, es lo que se denomina como salud móvil, cuya abreviatura es msalud, mhealth, un término usualmente utilizado para indicar la práctica de la medicina y a la salud pública con el apoyo de

dispositivos móviles. “El término fue usado por primera vez por Robert Istepanian como el uso emergente de las comunicaciones móviles y las tecnologías de red para la salud” (Alonso-Arévalo, & Mirón-Canelo, 2017, pág. 3), más específicamente se define como:

> El ejercicio de la medicina y la salud pública con apoyo de los dispositivos móviles, como teléfonos móviles, dispositivos de monitoreo de pacientes y otros dispositivos inalámbricos Sus aplicaciones son variadas y dependen de los objetivos con los que se desarrollen y los usuarios a los que se destine (Gavilondo, Vialart, 2016).

La mHealth o mSalud hace referencia en general, al uso de los dispositivos móviles y de smartphones en el área de la salud por parte de los médicos y pacientes y a las aplicaciones que facilitan el trabajo a los prestadores de salud y benefician a los usuarios.

2.10. Estado del Arte de Aplicaciones mHealth

El uso de aplicaciones móviles en el campo de la salud va en aumento en los últimos años y por ser un mercado en expansión existe una fuerte competencia entre desarrolladores de aplicaciones para satisfacer las necesidades de los clientes, lo que ha contribuido a crear una infraestructura en el mercado de la mHealth. Ejemplo de esto es que ahora se cuenta con una categoría por medio de la cual se clasifican las aplicaciones, que aunque no es una clasificación general y aceptada por todo el mundo, sí ha tenido un impacto significativo, y constituye un avance con respecto a cómo deben categorizarse cada una de las aplicaciones disponibles en el mercado. Según The App Intelligence, (2014), las aplicaciones se clasifican según su funcionalidad, así:

Tabla 4 Clasificación de las Aplicaciones Móviles

Categoría	Descripción
Información	Son las aplicaciones que tiene como principal función aportar información completa y detallada sobre alguna enfermedad o un área de especialización médica.

Educación y sensibilización	Son aplicaciones que también aportan información actualizada sobre alguna patología, pero que va más allá, facilitando la educación del paciente un público.
Registro y monitorización	Son aquellas que se centran en el registro de parámetros físicos y en el seguimiento de determinada actividad o comportamiento por parte del usuario.
Ayuda al diagnóstico	Son aquellas que facilitan el proceso de identificación de una enfermedad, aportando datos de valor para el profesional en salud.
Seguimiento del tratamiento	Aquellas que sirven de apoyo al paciente para mejorar su adherencia al tratamiento o tener control sobre cómo está llevando el mismo
Gestión y utilidades	Las que aportan información útil relacionada con la gestión sanitaria, citas médicas, localización de centros y profesionales de la salud entre otros.

Fuente: (The App Intelligence, 2014, pág. 14) (Elaboración Propia)

Por otro lado en el proceso de expansión del mercado de aplicaciones móviles se han desarrollado una inmensa cantidad de aplicaciones móviles en esta área. Evidencia de esta expansión es que para "2017 existían más de 150.000 aplicaciones médicas en Google Play, representando un aumento de 50% respecto al año anterior. [Por su parte] iOS, creció un 20% en este tipo de apps" (Vega, 2018, pág. 1). Otra evidencia del acelerado crecimiento de este tipo de aplicaciones es que para el año 2018, los reportes de IQVIA[1] indican que más

> …de 318,000 aplicaciones de salud están disponibles en las principales tiendas de apps y agregándose cada día unas 200 aplicaciones de salud. Esta situación tiene un impacto evidente en la medicina, ya que el 80% de los médicos en EEUU utilizan alguna aplicación de tipo mHealth" (Vega, 2018, pág. 1).

Este inusitado crecimiento se fundamenta en las posibilidades económicas que brinda un mercado en crecimiento, "para el año 2018 se calcula que los ingresos por *eSalud* alcanzarán un volumen económico de 21.000 millones de dólares en todo el mundo, 7.1000 millones solo en Europa" (Ramudo, 23 febrero, 2018, párr. 5). Las posibilidades económicas en este campo entrañan una dificultad, y es que muchas de las aplicaciones que se desarrollan, cuando demuestran que no son rentables son abandonadas por las empresas desarrolladoras sin retirarlas de las

[1] IQVIA es una compañía mundial de servicios de información, tecnología e investigación para ayudar a los grupos de interés del cuidado de la salud a encontrar mejores soluciones para sus pacientes.

plataformas de descarga, por lo que muchas de ellas quedan sin actualizaciones y sin cubrimiento, esto se debe a que no fueron bien diseñadas, o no respondieron a la necesidad real, o no fueron conocidas por los usuarios a los que iban dirigidas, y en consecuencia, un porcentaje muy pequeño ofrece un servicio sanitario eficiente o algún resultado útil, por lo que implican riesgos para los usuarios.

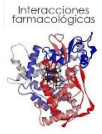
Con el propósito de enfrentar estas dificultades, varias organizaciones a nivel mundial han implementado estrategias de validación de las aplicaciones en salud existentes. Una de estas estrategias de validación es el Distintivo AppSaludable otorgado por la Agencia de Calidad Sanitaria de Andalucía, que es el primer distintivo que reconoce la calidad y seguridad de las apps de salud, el distintivo es gratuito y abierto a todas las aplicaciones de iniciativas públicas y privadas, tanto españolas como de cualquier otro país.

Por medio del distintivo se garantizan las aplicaciones en varias categorías, entre las que se cuentan, requisitos de diseño, usabilidad, privacidad, seguridad, calidad de la información, actualizaciones y frecuencia de actualización de fuentes. El mismo contribuye a categorizar cuales aplicaciones tienen mejor desempeño y tienen capacidad de contribuir a mejorar el bienestar de los usuarios.

Por las razones expresadas anteriormente, las aplicaciones que cuentan con la certificación de la Agencia de Calidad Sanitaria de Andalucía resultan ser las más importantes en el mercado hispanohablante, y es por esto que en este trabajo de investigación destacamos las aplicaciones que en el catálogo del Distintivo AppSaludable que tienen mayor calificación (Ver Tabla 5).

Tabla 5 Aplicaciones mHealth certificadas por AppSaludable.

Imagen	Aplicación mHealth	Clasificación	Descripción
	ReHand	Seguimiento de tratamiento	Es una herramienta de rehabilitación de muñeca, mano y dedos a través del dispositivo móvil. Cuenta con ejercicios con base a evidencia científica actualizada, consiguiendo una mejora en variables manuales tales como la destreza, la fuerza o la

			funcionalidad de la mano, necesarias para una óptima recuperación.
	Healthy Jeart	Educación y sensibilización	Aplicación orientada a adolescentes con el objetivo de aprender y mejorar hábitos de salud. La aplicación combina el juego con la formación saludable, ya que aparecen tips de salud, retos y más apartados.
	MásCaminos	Seguimiento del tratamiento	Es una aplicación diseñada como herramienta de intervención en salud mental, que ofrece un sistema proactivo enfocado en la prevención del suicidio a través de una red organizada de contactos.
	e_SaludAble	Gestión y utilidades	Esta aplicación ayuda a usar, navegar y manejar los recursos socio-sanitarios en el sistema de la Comunidad Autónoma de Andalucía, de manera que el usuario pueda promocionar y mantener una buena salud.
	Actuación Sanitaria al Maltrato	Educación y sensibilización	Muestra al personal sanitario cómo actuar ante un caso de violencia de género.
	Salud Mental Jaén Norte	Gestión y utilidades	Contiene información y recursos para usuarios o potenciales usuarios del Servicio, para facilitar la accesibilidad y la atención
	Guía de Antídotos en Intoxicaciones Agudas	Ayuda al diagnóstico	Una guía para el personal sanitario sobre identificación y diagnóstico de intoxicaciones y sus tratamientos.
	Interacciones Farmacológicas	Seguimiento del tratamiento	Permite la selección de un principio activo para posteriormente seleccionar uno o varios principios activos y determinar sus posibles interacciones.
	Guía Farmacológica 061	Seguimiento del tratamiento	Esta app facilita el acceso a una guía farmacológica especializada en la medicación utilizada en situaciones de urgencias y emergencias sanitarias.
	Hipot-Cnv	Ayuda al diagnóstico	Facilita la comunicación a personas que sufren de dificultad en la expresión oral. La aplicación consiste en un lenguaje audiovisual compuesto por pictogramas y voces sintetizadas. Inicialmente diseñado para su uso hospitalario en pacientes con Afasia.

	Tradassan	Ayuda al diagnóstico	Facilita la comunicación entre los trabajadores de un Servicio de Salud y los usuarios extranjeros que tienen dificultades con el idioma local. Idiomas: Español, Inglés, Francés, Alemán, Chino, Árabe
	Enfermeria Med Iv	Registro y monitorización	Aplicación dirigida a profesionales y estudiantes de enfermería o medicina, que incluye la descripción y modo de uso de fármacos con acceso a la ficha técnica
	Painometer v2	Registro y monitorización	Es una aplicación que sirve para evaluar y registrar la intensidad del dolor a lo largo del tiempo
	Primeros Auxilios Fáciles	Educación y sensibilización	Esta app te ayudará a conocer y a aprender las técnicas básicas de Primeros Auxilios de una forma fácil y sencilla, mediante animaciones e ilustraciones. Pensada para todos, niños, jóvenes y adultos.
	Asistente de RCP	Información	Esta aplicación guía a los usuarios de forma simple, eficaz y en tiempo real en situaciones de emergencias médicas graves, en especial en la muerte súbita.
	BCX Braden	Seguimiento del tratamiento	Es una aplicación pensada para su uso como apoyo a la actividad diaria de los profesionales sanitarios interesados en la prevención y tratamiento de las lesiones por presión
	Lady Pill Reminder	Seguimiento del tratamiento	Esta app está destinada a usuarias que toman la píldora anticonceptiva. Proporciona un paquete de píldoras virtual para visualizar el estado de las tomas del ciclo actual, y un sistema inteligente de notificaciones.
	iContraception	Seguimiento del tratamiento	Herramienta para ayudar a los profesionales para elegir el método anticonceptivo con los criterios médicos de elegibilidad
	iDoctus	Registro y monitorización	Es la primera herramienta móvil en español de consulta y referencia médica, exclusivamente para médicos. Ayuda a los médicos en el diagnóstico y tratamiento de sus pacientes

Fuente: (Distintivo AppSaludable, 2018) (Elaboración Propia)

Como vimos a lo largo del capítulo el campo de las aplicaciones mHealth está en constante desarrollo y expansión, pero este desarrollo es mucho más evidente en Estados Unidos y Europa. En el medio latinoamericano el modelo de la mHealth todavía tiene que progresar, y pasar de la fase de innovación a una fase más madura. “La mHealth se presenta como un factor clave en el desafío de avanzar hacia una sanidad más sostenible, mejorando la eficacia y eficiencia, reduciendo costes y atendiendo a las principales necesidades de nuestra sociedad (The App Intelligence, 2014, pág. 33). De ahí que, en el medio latinoamericano, sea evidente que es necesario potenciar el uso de las apps en el sistema de salud, pero ello debe ir acompañado de un proceso de mejoramiento y constante evaluación de la calidad de las aplicaciones desarrolladas, en este sentido, la mejora de los estándares de seguridad es una labor fundamental para alcanzar estos objetivos. Abordar estos elementos es el propósito del capítulo posterior.

3. METODOLOGÍA DE TESTING DE SEGURIDAD PARA APLICACIONES MÓVILES ANDROID EN SALUD

La seguridad de los dispositivos móviles es un tema que ha logrado cada vez más relevancia a consecuencia del aumento en la frecuencia de los ataques y sus secuelas. El incentivo para que se diera dicho incremento es el auge del uso de los dispositivos móviles, el aumento de información personal y confidencial almacenada en dichos dispositivos y el tipo de operaciones que son realizadas a través de ellos; por ejemplo, las que tienen que ver con actividades relacionadas con el área de la salud.

El sistema operativo Android es uno de los que tiene mayor cantidad de aplicaciones disponibles, por ende, es uno de los que más vulneraciones presenta. No obstante, este sistema operativo tiene unos buenos estándares de seguridad para responder ante posibles amenazas; sin embargo, los desarrolladores de aplicaciones tienen la opción de someter o no la app a regulaciones y permisos para el aseguramiento de la información, control que muchas veces no se lleva a cabo. Esto tiene como resultado las problemáticas más comunes, como dificultades en la transmisión de los datos e inconvenientes en la privacidad y almacenamiento de los mismos (Toro, Vargas & Hernández, 2015).

3.1. Pruebas de Seguridad

Existen muchos tipos de pruebas o testing de software, ellas se realizan con el objetivo de evaluar una aplicación móvil próxima al lanzamiento en el mercado, para evitar los riesgos y la probabilidad de fallas o vulnerabilidad en un asunto determinado que se esté evaluando, y que la aplicación pueda ser lanzada sin

mayores complicaciones. Esencialmente dichas pruebas son “investigaciones empíricas y técnicas cuyo fin es proporcionar información objetiva e independiente sobre la calidad del producto” (Hadfeg & Vega, 2017, pág. 84), Fernández (2005) hace referencia al testing como “una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas, los resultados se observan, registran y se realiza una evaluación de algún aspecto” (p. 44).

El objetivo general de las pruebas es asegurar que se tienen un software de calidad, es decir que se tiene un software que puede ser modificado, que es confiable, eficaz y que puede ser usado fácilmente. Entre estos tipos de pruebas, están las que evalúan los requisitos funcionales y las que lo hacen en términos de los no funcionales. Las primeras evalúan el software en función de “los criterios que éste debe cumplir para que éste sea adecuado para su propósito” (PMOinformatica, 2013, párr. 6), mientras que las no funcionales “especifican los criterios que debe cumplir para que sea adecuado para su uso” (PMOinformatica, 2013, párr. 6). Una de las circunstancias del mercado a partir de la que aumenta la vulnerabilidad de las aplicaciones, es que muchos desarrolladores ponen más empeño en cumplir con los criterios funcionales, dejando a un lado los criterios no funcionales, y aunque ya no es tan generalizado debido a las regulaciones las pruebas que evalúan la calidad de este tipo de criterios se dejan a un lado. Las pruebas o testing de seguridad, por ejemplo, son pruebas no funcionales, es decir, que se encargan de evaluar que las medidas de seguridad adoptadas tengan criterios adecuados para el óptimo uso de las aplicaciones.

El testing o la prueba de un software, es uno de elementos más importantes si se desea desarrollar una aplicación móvil de calidad. No obstante, para ello es necesario en primera instancia estudiar o conocer diversas maneras de crear las pruebas en relación a la app que se quiere crear, y no simplemente llevar a cabo un cúmulo de ellas sin saber si podrán ser de ayuda o no para la ejecución de la aplicación. Por lo anterior, es importante previamente realizar una “metodología de

desarrollo" que se enfoque en las necesidades específicas de la aplicación, para que en los resultados de las pruebas se pueda realizar un contraste entre lo que se esperaba y los resultados actuales (Aristegui, 2010).

La metodología para testing posibilita que cualquier aplicativo móvil pueda ser evaluado y pueda salir al mercado con la seguridad de que sea una herramienta práctica y funcional para el usuario. Es por ello que dicha metodología debe ser tomada en serio, como un trabajo de compromiso y responsabilidad debido a que es la vía para que los errores que pueda tener en el dispositivo puedan ser eliminados o disminuido previamente, sin que esto acarree problemas mayores más adelante. Además, si dichos errores son ubicados anticipadamente, se posibilitara que haya menores dificultades en relación al costo los mismos, y ventajas en su productividad (Franco, 2010).

En el área del testing o pruebas de software, una metodología de testing, conceptualmente puede ser lo mismo que una estrategia para la creación de éste. Algunos autores hacen referencia a las estrategias como las "técnicas o metodologías" que se llevaran a cabo para la creación de las pruebas. Otros, como la planeación de las mismas. Sea como fuere, lo importante es que antes que se realice la prueba de software de una aplicación móvil, se debe realizar todo un proyecto que dé cuenta de que las pruebas que se ejecutarán, y evaluarán el funcionamiento de aquello que se desea testear (seguridad, funcionalidad, Etc.) (Cálad, Ruiz, 2009).

3.1.1. Tipos de pruebas de seguridad

Según la clasificación de OSSTMM existen seis tipos de pruebas de seguridad, en general las pruebas difieren en la relación existente entre el evaluador y la aplicación que debe evaluar, más específicamente, difieren de la cantidad de información que tiene el evaluador sobre las aplicaciones que debe evaluar y su infraestructura (target), lo que el target de la prueba sabe sobre el probador o las expectativas que

tiene de la prueba misma, además de la legitimidad que tenga la prueba. Según esta clasificación los equipos de prueba o testing serían:

Tabla 6 Tipos de testing de seguridad

Tipo		Descripción
1	Ciega	El analista se compromete con el target del testing sin conocimiento previo de sus defensas. El target está preparado para la auditoría, sabiendo de antemano los detalles de la auditoría. Una auditoría ciega principalmente prueba las habilidades del analista. La amplitud y profundidad de una auditoría ciega solo puede ser tan amplia como lo permita el conocimiento y la eficiencia aplicables del Analista.
2	Doble ciega	El analista se compromete con el target sin conocimiento previo de sus defensas. El target no recibe una notificación previa del alcance de la auditoría, los canales probados o los vectores de prueba. Una auditoría a doble ciego evalúa las habilidades del analista y la preparación del objetivo frente a variables desconocidas de agitación. La amplitud de profundidad de cualquier auditoría ciega solo puede ser tan amplia como lo permita el conocimiento y la eficiencia aplicables del Analista. Esto también se conoce como prueba de caja negra o prueba de penetración.
3	Caja gris	El analista se compromete con el target con un conocimiento limitado de sus defensas y activos y un conocimiento completo de los canales. El objetivo está preparado para la auditoría, a sabiendas de todos los detalles de la auditoría. Una auditoría de caja gris pone a prueba las habilidades del analista. La naturaleza de la prueba es la eficiencia. La amplitud y profundidad dependen de la calidad de la información proporcionada al Analista antes de la prueba, así como del conocimiento aplicable del Analista. Este tipo de prueba a menudo se denomina prueba de vulnerabilidad y la mayoría de las veces es iniciada por el objetivo como una autoevaluación.
4	Doble caja gris	El analista se compromete con el target con un conocimiento limitado de sus defensas y activos y un conocimiento completo de los canales. El target se notifica por adelantado del alcance y el marco temporal de la auditoría, pero no los canales probados ni los vectores de prueba. Una auditoría de doble caja prueba las habilidades del analista y la preparación del objetivo para identificar las desconocidas variables de agitación. La amplitud y profundidad dependen de la calidad de la información

		proporcionada al Analista y del objetivo antes de la prueba, así como del conocimiento aplicable del Analista. Esto también se conoce como una prueba de caja blanca.
5	Tándem o en equipo	El analista y el target están preparados para la auditoría, ya que ambos conocen de antemano todos los detalles de la auditoría. Una auditoría en tándem prueba la protección y los controles de la meta. Sin embargo, no puede probar la preparación del objetivo ante variables desconocidas de agitación. La verdadera naturaleza de la prueba es la minuciosidad, ya que el analista tiene una visión completa de todas las pruebas y sus respuestas. La amplitud y profundidad dependen de la calidad de la información proporcionada al Analista antes de la prueba (transparencia), así como del conocimiento aplicable del Analista. Esto se conoce a menudo como un auditor interno, una prueba de Crystal Box y el analista a menudo forma parte del proceso de seguridad.
6	Inversión	El analista se compromete con el target con pleno conocimiento de sus procesos y seguridad operacional, pero el objetivo no sabe nada de qué, cómo o cuándo el analista apostará. La verdadera naturaleza de esta prueba es auditar la preparación de las variables objetivo y los vectores de agitación conocidos. La amplitud y profundidad dependen de la calidad de la información proporcionada al Analista y al conocimiento y creatividad aplicables del Analista. Esto también es a menudo llamado un ejercicio del equipo rojo.

Fuente: (Herzog, 2010, pág. 37) (Elaboración Propia)

3.1.2. Fases de las pruebas de Seguridad

El testing o prueba de software de cualquier aplicación debe de ser un proceso guiado por una metodología que tenga un carácter científico, es decir que pueda ser reproducible y evaluado en sus fases, de ahí que, entendidas como un proceso las pruebas de seguridad deban pasar por fases específicas para una correcta evaluación de la misma, Pérez (2006) indica el siguiente proceso

Tabla 7 Fases del Testing

Fase	Descripción
Estudio preliminar y planificación	En esta fase se acuerda hasta donde se va a evaluar o qué puntos se van a evaluar, se realiza un proyecto general sobre el tema y se planea fechas específicas para su ejecución.

Ciclos de prueba	Se producen y realizan las pruebas planeadas anteriormente
Seguimiento del ciclo	Se realiza un rastreo permanente de cada etapa que conlleva el realizar las pruebas, para validar que se ejecute de forma correcta.
Configuración del entorno	Se diferencia entre el entorno que conlleva la realización de las pruebas y lo que éstas evalúan.
Diseño de pruebas	Se crean las pruebas en relación a las características del dispositivo y las necesidades del tema en sí.
Ejecución	Se compara el funcionamiento actual del producto creado frente a aquello que se esperaba del mismo
Evaluación	Se valora el proceso y los resultados y se almacena la información.

Fuente: (Pérez, 2006) (Elaboración propia)

Además de lo planteado por Pérez (2006), Herzog, (2010) define una serie de recomendaciones que el evaluador debería tener en cuenta para llevar a cabo correctamente una prueba de seguridad.

En principio es necesario definir exactamente qué es lo que se desea proteger, posteriormente necesario definir una zona de compromiso, es decir, es necesario identificar el área que se desea evaluar y los mecanismos de protección y los servicios con los cuales cuenta la aplicación, asimismo es necesario definir los alcances de la prueba de seguridad, también es necesario identificar el equipo y las herramientas que deben tenerse disponibles para realizar la prueba, debe definirse el tipo de prueba que se va realizar (ciega, caja gris, etc.), eso depende de la relación del evaluador con la infraestructura a evaluar, y finalmente es necesario asegurarse que las pruebas cumplen con reglas de compromiso, estas están diseñadas para garantizar que se cumpla las expectativas y que no se genere malentendidos tras la realización de las pruebas (pág. 33)

3.1.3. Herramientas para testing de seguridad

Entre los requerimientos para realizar pruebas de seguridad se encuentran las herramientas para el testing de seguridad, estas herramientas pueden ser de pago, o software libre, en todo caso debe permitir son pruebas autónomas y masivas que

evalúen efectivamente los criterios de seguridad de las aplicaciones. Fundamentalmente tres tipos de herramientas: herramientas estáticas, dinámicas y forenses (Ver Tabla 7)

Tabla 8 Tipos de Herramientas

Tipo	Descripción
Herramientas de pruebas estáticas	Como su nombre lo indica, analizan la aplicación cuando está no está en funcionamiento. Esta clase de herramientas puede beneficiar las alteraciones que puedan proceder en relación a los datos del usuario.
Herramientas de pruebas dinámicas	Permiten la evaluación de la aplicación cuando ésta se encuentra en movimiento.
Herramientas de pruebas forenses	Posibilitan evaluar aquellas unidades que por un momento se ha utilizado (como “datos sensibles”), pero que luego el usuario se ha enfocado en otros asuntos de la aplicación.

Fuente: (Cornell, 2014) (Elaboración Propia)

3.1.4. Vulnerabilidades en Android

Para la correcta realización de las pruebas de seguridad, se deben tener enfoques basados en los riesgos que tienen las aplicaciones pueden software que se quiere evaluar, esto porque la seguridad nunca puede garantizarse totalmente, en función de esto, es necesario que los desarrolladores realicen evaluaciones de los riesgos de seguridad y que se determinen las vulnerabilidades a las que está expuesto el software.

Como se afirmó anteriormente, Android es una de las plataformas más expuestas a vulnerabilidades por ser la que tiene mayor difusión y mayor cuota de mercado, lo que la hace el objetivo preferido por los ciberdelincuentes. Según la empresa de seguridad informática ESET Android cierra el año 2017 “con un global de 735 fallos conocidos, un 40.5% más que 2016” (ESET, 2018, párr. 2). Uno de los datos más relevantes que surgen a partir del informe de ESET es “el 27% de estas vulnerabilidades podrían implicar la ejecución de código malicioso, mientras que el 45% de las mismas se consideran altamente críticas” (ESET, 2018, párr. 2), es

decir, que estos fallos vulneran la seguridad de la información almacenada en el dispositivo o de la que se transfiere por él o hacia él.

Entre los reportes positivos de ESET, se cuenta con que el volumen de aplicaciones maliciosas detectadas en la tienda de Android ha disminuido “un 47.24% con respecto a 2016” (ESET, 2018, párr. 3), el tipo de riesgo más común por el cual se caracterizan estas aplicaciones maliciosas, son las de troyanos disfrazados de aplicaciones benignas, pero la sustancial disminución que se ha reportado se debe a mayores controles de seguridad aplicados por Google para las aplicaciones que se suben en su tienda.

3.1.5. Clasificación de la protección de datos para aplicaciones mHealth

Si bien el desarrollo de las aplicaciones mHealth ha impactado positivamente el campo de los servicios sanitarios, toda vez que las aplicaciones mHealth permiten, por ejemplo, monitorizar el tratamiento de los pacientes en su propio entorno, y toda una gama de beneficios en términos de acceso eficaz y eficiente a la información, estos desarrollos han implicado una serie de riesgos para el aseguramiento de la cadena de custodia de unos datos que se consideran críticos, esto se debe a que “el rápido crecimiento de este mercado hace que buena parte de las aplicaciones que se descargan no hayan sido acreditados por un organismo que garantice su calidad y seguridad” (Alonso-Arévalo & Mirón-Canelo, 2017, pág. 5).

Sumado a lo anterior, si bien existe sobreoferta en el mercado de aplicaciones mHealth, la incorporación en el campo sanitario y su uso por parte de profesionales, usuarios y proveedores ha resultado limitada, según Guillen-Pinto, Ramírez-López & Cifuentes-Sanabria, (2017), debido a “conectividad limitada y falta de integración con los sistemas de salud, bajos niveles de confidencialidad de datos, privacidad, seguridad e incertidumbres regulatorias y falta de evidencia científica que mide la eficacia de las aplicaciones” (pág. 281), según los mismos autores una solución que puede romper con las problemáticas presentadas en el campo de las aplicaciones

mHealth, pasa por definir una normativa internacional convergente que permita evaluar, supervisar y controlar la producción y distribución de las aplicaciones.

Fundamentalmente la protección de la información de carácter clínico, según Alonso-Arévalo & Mirón-Canelo (2017) debe ser asegurada en dos aspectos: protección y seguridad de los datos y funcionamiento seguro que no afecte negativamente la salud del paciente (pág. 5).

Según un estudio realizado por el State of Application Security en 2016, citado por Alonso-Arévalo & Mirón-Canelo (2017), la mayoría de los encuestados considera favorable el uso apropiado de dispositivos móviles en el campo de la salud, pero “el 85% de los encuestados cita la seguridad como el principal reto, mientras que el 77% cita la privacidad de los datos y aproximadamente la mitad el uso inapropiado” (pág. 7), según los autores, quienes citan un estudio realizado por OWASP, los riesgos más incidentes de las aplicaciones mHealth se caracterizan porque el “97% carecían de protección del código binario y podían ser modificadas por lógica inversa, y casi el 80% tenía una pobre protección de la capa de transporte y podría usarse para robar datos o identidad” (pág. 7). Consecuentemente, el cifrado de datos es un componente esencial del aseguramiento de la información y la preservación del anonimato, característica crucial que deberían tener las aplicaciones mHealth.

Otro de los aspectos importantes que señalan Alonso-Arévalo & Mirón-Canelo (2017), es el funcionamiento seguro de las aplicaciones, esto significa que las aplicaciones móviles mHealth deben disponer de garantías de seguridad para que su comportamiento sea el que se supone que debe tener, y de esta manera se minimiza el riesgo de afectación a la salud de los pacientes, en el sentido que la información proporcionada por un médico o paciente debe ser exacta. El riesgo sustancial aquí es que las aplicaciones presenten mal funcionamiento, lo cual puede conllevar “a un diagnóstico erróneo por parte del profesional sanitario por haber obtenido datos inadecuados, o por la inadecuada utilización del dispositivo por parte del paciente” (pág. 7).

Martínez et al. (2015) citado por Guillen-Pinto, Ramírez-López & Cifuentes-Sanabria, (2017), establece unas categorías para la seguridad de aplicaciones mHealth, así: cobertura de la información de los pacientes, requerimientos y métodos utilizados para recoger la información, requisitos de consentimiento, retención de datos, seguridad durante la adquisición, transmisión y almacenamiento de los datos y obligaciones de notificación (pág. 282). Fundados en estos aspectos los autores Guillén-Pinto et al. (2017) establecen categorías para los requerimientos de seguridad de las aplicaciones mHealth, de esta forma:

Tabla 9 Categorías para los requerimientos de seguridad de las aplicaciones mHealth

Categoría	Descripción
Autenticación	Permite conocer la identidad de cada uno de los usuarios dispositivos que pertenecen a una red de información
Autorización	Consiste en que los usuarios sólo pueden acceder a la información para la cual están autorizados
Confidencialidad	Consiste en garantizar que la información clínica del usuario sólo sea conocida por él y por los usuarios a los cuales él autorice (médicos o enfermeras)
Control de acceso	Consiste limitar el número de usuarios que acceden a la información, con el objetivo de no afectar la disponibilidad de los servicios prestados.
Integridad	Consiste en garantizar la no alteración de la información durante el proceso de transmisión
Disponibilidad	Consiste en asegurar que la información clínica del paciente en caso de una emergencia sea accesible
Interoperabilidad	Consiste en que todos los elementos del sistema deben estar regidos por parámetros que permitan la comunicación entre ellos y el uso de protocolos que faciliten la comprensión de la comunicación entre las aplicaciones del sistema información

Fuente: (Guillen-Pinto, Ramírez-López & Cifuentes-Sanabria, 2017, pp. 283-284) (Elaboración propia)

Detectar y clasificar las vulnerabilidades de las aplicaciones mHealth es un paso fundamental para garantizar la correcta incorporación de estas tecnologías en el campo de la salud, paso necesario pues dichas aplicaciones contribuyen de manera efectiva a apoyar e informar adecuadamente a los pacientes en procesos autogestionados de salud y bienestar, pero para que se cumpla este objetivo es necesario disponer de aplicaciones fiables y útiles. Y para esto es necesario

disponer de metodologías de seguridad que permitan anticipar los riesgos de exposición a vulnerabilidades y a través de ellas asegurar la calidad de las aplicaciones mHealth, que es el propósito siguiente capítulo.

3.2. Metodologías para la realización de pruebas de seguridad

Existen diferentes compañías y organizaciones que trabajan en el desarrollo de metodologías para el análisis seguridad. En este apartado se realiza una descripción de las metodologías a las que se tuvo acceso. Entre estas aplicaciones se referencian algunas que no hacen alusión específicamente a Android, por lo que no se las aborda a profundidad, pero se incluyen pues son importantes para el proceso de investigación, ya que aportan información de contexto sobre las características de las metodologías de pruebas o testing seguridad. Las metodologías sobre las que se hace análisis en profundidad son: OWASP y OASAM, y es a partir de ellas que se construye la propuesta de test de seguridad.

3.2.1. Metodología OSSTMM

El Manual de la Metodología Abierta de Testeo de Seguridad (OSSTMM)[2] (Ver Figura 10), es una metodología de pruebas de seguridad que tiene carácter gratuito y abierto diseñada por el Instituto ISECOM. Es una de las metodologías más completas y usadas en procesos de auditoría de seguridad en Internet. El trabajo que realizan lo fundan en un principio filosófico según el cual, los hechos no proviene de los grandes saltos del descubrimiento, sino de los pequeños y cuidadosos pasos de la verificación (Herzog, 2017).

[2] La información de referencia sobre la metodología OSSTMM se puede ampliar en http://www.isecom.org/research/

Figura 9 Manual de la Metodología Abierta de Testeo de Seguridad (OSSTMM)

Según la web del Instituto que desarrolla la metodología, la ventaja de usarla es que se logra una profunda comprensión de la interconexión de las cosas. Las personas, los procesos, los sistemas y del software.

La metodología estructura su contenido en función de los aspectos más importantes de los sistemas de información, tratando de abarcarlos en su totalidad:

- Seguridad de la información
- Seguridad de los procesos
- Seguridad en las tecnologías de Internet
- Seguridad en las comunicaciones
- Seguridad inalámbrica
- Seguridad física

A cada uno de estos aspectos le corresponde unas actividades de pruebas específicas en las que se realizan comprobaciones de las especificaciones de seguridad del entorno que se esté evaluando. Un aspecto especial de esta metodología es que no sólo se refiere a los aspectos puramente técnicos en el campo de la seguridad, sino que también evalúa las capacidades de los responsables de las pruebas de seguridad.

Otro de los aspectos especiales de esta metodología es que busca definir en categorías claras los alcances de cada una de sus actividades, lo que les permite

delimitar el contexto de las comprobaciones, OSSTMM establece las siguientes categorías:

- Búsqueda de Vulnerabilidades: Se orienta a realizar comprobaciones automáticas de un sistema o sistemas dentro de una red.
- Escaneo de la Seguridad: Está orientado al sondeo de las principales vulnerabilidades en el sistema a comprobar, e incluyen verificaciones manuales de falsos positivos, identificación de los puntos débiles en el sistemas y análisis individualizado.
- Test de Intrusión: Son test de pruebas que se plantean con el objetivo de romper la seguridad de un sistema.
- Evaluación de Riesgo: Se compone la serie de análisis de seguridad a través de entrevistas e investigación de nivel medio que incorporan justificaciones de negocios, legales y específicas de la industria.
- Auditoria de Seguridad: Se ha ejercicios continuados de inspección que obran sobre un sistema por parte de los administradores que garantizan el cumplimiento de las políticas de seguridad definidas.
- Hacking Ético: Está orientado a obtener por medio de test de intrusión, objetivos dentro de la red.

El trabajo minucioso que han hecho los creadores de esta metodología la convierten en un ejemplo valioso para los propósitos de este trabajo, ya que aporta elementos interesantes como por ejemplo: que las actividades de testing no sólo se restrinjan a los procesos técnicos, sino también a las actividades humanas en relación con el control y manejo de las aplicaciones, esto porque es la actividad humana, expresada diversas formas, la que genera los riesgos de seguridad, de ahí que este haya sido un aprendizaje interesante al haber abordado esta metodología.

3.2.2. Offensive Security

Figura 10 Ofenssive Security

Offensive Security[3] es una metodología internacional para realizar auditorías informáticas por medio de pruebas de intrusión y estudios de seguridad, su enfoque metodológico se orienta en el concepto *seguridad ofensiva* por medio de su implementación lo que se busca es explotar las vulnerabilidades que pudieran presentar los objetivos de las auditorías, este modelo es directamente intrusivo y los resultados que arroja no se basan en estadísticas compuestas por herramientas sino en los resultados de las pruebas de penetración.

La metodología de Offensive Security corresponde a un proceso de cinco pasos cada uno diferenciado y con objetivos claros, entre ellos se cuentan: recolección de información, análisis de vulnerabilidades, definición de objetivos secundarios, ataque, análisis de resultados.

La etapa de recolección de información corresponde al momento de identificación de los objetivos a atacar en los cuales existen dos escenarios posibles: realización de pruebas ciegas, es decir, no se cuenta con información del cliente, o realización de pruebas con información, el cliente proporciona cierta información al auditor.

La etapa de análisis de vulnerabilidades consiste en determinar los problemas de seguridad y esta etapa puede ser realizada de forma manual, o de forma automática

[3] La información de referencia sobre la metodología Offensive Security se puede ampliar en https://www.offensive-security.com/

cuando se utilizan herramientas para auditorías, esta etapa finaliza con la identificación de la estrategia de realización de la prueba de penetración.

Etapa de definición de objetivos, consiste en profundizar en las definiciones de los objetivos anteriormente señalados esto se hace con el objetivo de aumentar la probabilidad de que los ataques resulte exitosos, en esta etapa se realiza el escalamiento de los privilegios, es decir, se definen objetivos secundarios que pueden aportar a la consecución del objetivo primario.

La cuarta etapa es la etapa de ataques, en ella se cristalizan los hallazgos de las etapas anteriores, es decir, la definición de los objetivos y la detección de vulnerabilidades. Es en esta etapa es donde se evidencian las problemáticas del sistema de seguridad que se está evaluando.

La quinta etapa es el análisis de resultados, esta fase consiste en analizar los resultados de los ataques producidos en la etapa anterior, los autores aconsejan que si el ciclo pruebas anteriores no tuvo resultados positivos, las pruebas deben ser repetidas.

Análisis final y documentación es la última etapa en ella se realiza un informe detallado que condensa los hallazgos y las soluciones a los problemas encontrados.

La metodología Offensive Security, es una metodología intrusiva y altamente confiable, pues ella no presenta falsos positivos, dadas las características de la realización de las pruebas, la cual es una debilidad de muchas otras metodologías que usan herramientas de auditoría automatizadas para probar las características de los sistemas de seguridad que evalúan. Esto es relevante para este trabajo de investigación, puesto que una metodología asertiva y confiable debe privilegiar las pruebas de penetración en lugar del uso de herramientas automatizadas.

3.2.3. Metodología ISSAF

Figura 11 Information System Security Assessment Framework ISSAF

La metodología ISSAF o Marco de Evaluación de la Seguridad del Sistema de Información[4] es un marco estructurado revisado por pares que categoriza la evaluación de la seguridad del sistema de información en varios dominios, a su vez realizan evaluaciones específicas y establece los criterios de prueba para cada uno de los dominios evaluados.

Su objetivo es proporcionar aportes de campo sobre las evaluaciones de seguridad con el objetivo que reflejen escenarios de la vida real. ISSAF se usa fundamentalmente para dar cumplimiento a los requisitos de evaluación de seguridad de una organización. El objetivo final de ISSAF es el de obtener una imagen completa de las vulnerabilidades que puedan existir.

ISSAF estructuras son evaluaciones a partir de criterios los cuales son revisados por pares externos expertos en la materia. Entre estos criterios de evaluación se incluyen:

- Descripción de los criterios de evaluación.
- Clara definición de fines y objetivos.
- Establecimiento de requisitos previos para realizar las evaluaciones.
- Descripción del proceso para la evaluación.
- Formulación de informes sobre los resultados esperados

[4] La información de referencia sobre la metodología ISSAF se puede ampliar en http://www.oissg.org/files/issaf0.2.1.pdf

- Recomendaciones sobre contramedidas para tomar

Los objetivos trazados por la metodología ISSAF buscan establecerlo como un estándar de referencia a nivel global sobre los criterios de seguridad de los sistemas informáticos, entre estos objetivos se cuentan.

- Actuar como marco referencia para la evaluación de seguridad.
- Estandarizar el proceso de evaluación de seguridad.
- Proporcionar una línea de base para la realización de evaluaciones
- Ser una referencia para evaluar las protecciones contra el acceso no autorizado.
- Fortalecer los procesos y tecnologías de seguridad existentes.

Como se afirmó anteriormente el objetivo de la ISSAF es proporcionar un único punto de referencia para evaluar la seguridad y sus diseñadores afirman que sus protocolos están estrechamente alineados con los problemas del mundo real, lo que la constituye como una propuesta de valor para los grupos de interés. Para ello, la ISSAF traza una agenda que se describe a continuación:

- Realizar evaluaciones de las políticas y los procesos de seguridad de la información y asegurarse que cumplan con los requisitos de la industria y regulaciones aplicables.
- Identificar la infraestructura de los sistemas informáticos críticos necesarios en la cadena de valor de las organizaciones y evaluar su seguridad.
- Realizar evaluaciones de vulnerabilidad y pruebas de penetración para resaltar las vulnerabilidades del sistema y en consecuencia, identificar las debilidades en los sistemas, redes y aplicaciones.
- Evaluar los controles seguridad mediante:

Encontrar configuraciones erróneas y rectificarlas.

Identificar los riesgos de las tecnologías y abordarlos.

Identificar riesgos dentro de personas o procesos de negocio y abordarlos.

Fortalecerlos los procesos y tecnologías existentes.

• Priorizar las actividades de evaluación según la criticidad del sistema, los gastos de prueba y los beneficios potenciales.

• Educar a las personas en la realización de evaluaciones de seguridad

• Educar a las personas sobre sistemas de seguridad, redes y aplicaciones.

Este enfoque de ISSAF se basa en usar el camino más corto para lograr los objetivos trazados, intentando encontrar fallas que puedan explotarse, con el menor esfuerzo. El objetivo de este marco de evaluación es proveer integridad y precisión a los sistemas de información, además de eficiencia a las evaluaciones de seguridad.

Uno de los principales aportes que realiza la metodología ISSAF, es que parte integral de su desarrollo el entendimiento de los clientes sobre el por qué y cómo se deben realizar evaluaciones de seguridad, además de concientizarlos sobre cuáles son las acciones de la vida diaria que aumenta la vulnerabilidad de las organizaciones, de ahí que el haber entrado en relación con esta metodología fue relevante para este trabajo de investigación.

3.2.4. Metodología OWASP

OWASP o Proyecto Abierto de Seguridad de Aplicaciones Web, por sus siglas en inglés, es una organización sin ánimo de lucro que surge en diciembre de 2001, con el objetivo de aportar a mejorar las capacidades de seguridad del software, además de visibilizar la importancia de garantizar los criterios de seguridad en aplicaciones y brindar información adecuada para gestionar los riesgos relativos a la seguridad de las aplicaciones. El proyecto OWASP funda sus acciones a partir de un enfoque colaborativo en el que cualquier profesional de la seguridad informática puede aportar su conocimiento. OWASP es una organización que promueve el carácter abierto de los contenidos e ideas y conceptos innovadores con respecto a garantizar la seguridad.

El proyecto OWASP, como su nombre lo indica, nace con el propósito de garantizar la seguridad de aplicaciones web, pero partir de su experiencia en el mercado de la seguridad, se hizo evidente que dado el auge de las aplicaciones móviles era necesario generar una guía enfocada específicamente a este tipo de aplicaciones, con este propósito se crea Mobile Security Requirements and Testing Guide (Ver Figura 12)

Figura 12 Mobile Security Requirements and Testing Guide de OWASP

Una parte integral de la guía desarrollada por OWASP es el Mobile Application Security Verification Standard (MASVS)[5] o Estándar de Verificación de Seguridad en Aplicaciones Móviles, el MASVS tiene la intención de estandarizar los requerimientos de seguridad ajustados a los diferentes escenarios como el diseño, desarrollo y prueba de aplicaciones móviles, además de estar ajustado para los diferentes niveles de amenaza.

El MASVS define los niveles de verificación de la seguridad: L1 y L2, “Los niveles MASVS-L1 y MASVS-L2 contienen requerimientos genéricos de seguridad recomendados para todas las aplicaciones móviles (L1) y para aplicaciones que manejan datos altamente sensibles (L2)” (OWASP, 2018, pág. 8).

Para OWASP la verificación MASVS-L2 son un conjunto de medidas recomendadas para aplicaciones que manejen información de tipo sensible, entre las cuales ubica específicamente a la industria de la salud. Por este hecho en este trabajo nos

[5] La información de referencia sobre la metodología OWASP se puede ampliar en https://www.owasp.org/index.php/Main_Page

concentraremos en describir los elementos esenciales de la verificación MASVS-L2.

El estándar de Verificación de Seguridad en Aplicaciones Móviles (MASVS) contiene un listado de los requerimientos de seguridad para las aplicaciones móviles, mientras que la Guía de Pruebas de Seguridad Móvil de OWASP (MSTG) describe los procedimientos técnicos para verificar dichos requerimientos. Para efectos del cumplimiento de los objetivos de este trabajo haremos uso del estándar MASVS, dejando la información requerida para tener acceso a los procedimientos técnicos en el caso que el lector quiera profundizar en ellos[6].

3.2.4.1. Requerimientos en el Almacenamiento de datos y la Privacidad

En esta categoría se listan los requerimientos concernientes al almacenamiento de la información y su aseguramiento, un aspecto clave de la seguridad móvil. El tipo de información que se pretende asegurar a través de estas medidas son: las credenciales de usuario información privada, información que se define como datos sensibles, para el MASVS los datos sensibles son:

- ✓ Información de identificación personal que puede ser usada para el robo de identidad: números de seguro social, números de tarjetas de crédito, números de cuentas bancarias, información médica;
- ✓ Datos altamente confidenciales que, en caso de que se comprometieran, ocasionarían daños a la reputación y/o costes financieros: información contractual, información cubierta por acuerdos de confidencialidad, información de gestión;
- ✓ Cualquier dato que debe ser protegido por ley o por razones de conformidad (OWASP, 2018, pág. 16).

[6] La guía de pruebas de seguridad móvil de OWAST (MSTG) está disponible en el sitio web https://sushi2k.gitbooks.io/the-owasp-mobile-security-testing-guide/content/

El aseguramiento de los datos sensibles puede garantizarse a través de seguir unas reglas sencillas (Ver Tabla)

#	Descripción	L2
1	Las funcionalidades de almacenamiento de credenciales del sistema son utilizadas para almacenar información sensible, como credenciales de usuario y claves y criptográficas	✓
2	No se escribe información sensible en los registros de la aplicación	✓
3	No se comparte información sensible con servicios externos salvo que sea una necesidad de la arquitectura	✓
4	Se desactiva el caché del teclado en los campos de texto donde se maneja información sensible	✓
5	Se desactiva el portapapeles en los campos de texto donde se maneja información sensible	✓
6	No se expone información sensible mediante mecanismos entre procesos (IPC)	✓
7	No se expone información sensible como contraseñas y números de tarjetas de crédito a través de la interfaz o capturas de pantalla	✓
8	No se incluye información sensible en los backups generados por el sistema operativo	✓
9	La aplicación remueve la información sensible de la vista cuando la aplicación pasa a un segundo plano	✓
10	La aplicación no conserva la información sensible en memoria más de lo necesario y la memoria limpiada luego de su uso	✓
11	La aplicación obligada a que exista una política mínima seguridad en el dispositivo, como que el usuario deba configurar un código de acceso	✓
12	La aplicación educa al usuario acerca de los tipos de información personal que procesa y de las mejores prácticas de seguridad que el usuario debería seguir al utilizar la aplicación	✓

Fuente: (OWASP, 2018, pág. 16)

3.2.4.2. Requerimientos de criptografía

Un componente esencial para la protección de la información y de los datos almacenados en un dispositivo móvil es la criptografía, el propósito de los controles

es asegurar que la aplicación utiliza las mejores prácticas criptográficas de la industria, las cuales incluyen:

- ✓ Uso de librerías conocidas y probadas;
- ✓ Configuración y elección de primitivas criptográficas apropiado;
- ✓ Cuando se requiere de randomización se selecciona el generador debido.

#	Descripción	L2
1	La aplicación no depende únicamente de criptografía simétrica con "claves a fuego"	✓
2	La aplicación utiliza implementaciones de criptografía probadas	✓
3	La aplicación utiliza primitivas de seguridad que son apropiadas para el caso particular y su configuración y sus parámetros siguen las mejoras prácticas de la industria	✓
4	La aplicación no utiliza protocolos o algoritmos criptográficos que son considerados deprecados para aspectos de seguridad	✓
5	La aplicación no reutiliza la misma clave criptográfica para varios propósitos	✓
6	Los valores random son generados utilizando un generador de números suficientemente ramdómicos	✓

Fuente: (OWASP, 2018, pág. 18)

3.2.4.3. Requerimientos de Autenticación y Manejo de Sesiones

En las aplicaciones móviles, la mayoría de las veces los usuarios deben iniciar sesión en un servicio remoto, lo cual constituye una parte esencial de la arquitectura global de las aplicaciones, y aunque "la mayoría de la lógica ocurren el servidor, MASVS define algunos requerimientos básicos sobre cómo manejar las cuentas y sesiones del usuario" (OWASP, 2018, pág. 19)

#	Descripción	L2
1	Si la aplicación provee acceso un servidor remoto, un mecanismo aceptable de autenticación como usuario y contraseña es realizado en el servidor remoto	✓
2	Si se utiliza la gestión de sesiones por estado, el servidor remoto usa tokens de acceso ramdómicos para autenticar los pedidos del cliente sin requerir el envío de las credenciales del usuario en cada uno	✓
3	Si se utiliza la autenticación basada en tokens sin estado, el servidor proporciona un token que se ha firmado utilizando un algoritmo seguro	✓

4	Cuando el usuario se desloguea se termina la sesión también en el servidor	✓
5	Existe una política de contraseñas y es aplicada en el servidor	✓
6	El servidor implementa mecanismos, cuando credenciales de autenticación son ingresadas una cantidad excesiva de veces	✓
7	La autenticación biométrica, si hay, no está atada a un evento (usando una API que simplemente retorna “true” o “false”). Sino que está basado en el desbloqueo del Keystore de Android	✓
8	Las sesiones y los tokens de acceso expiran luego de un tiempo predefinido de inactividad	✓
9	Existe un mecanismo de segundo factor de autenticación (2FA) en el servidor y es aplicado consistentemente	✓
10	Para realizar acciones que manejen información sensible se requiere una re-autenticación	✓
11	La aplicación informa al usuario acerca de los accesos a su cuenta. El usuario es capaz de ver una lista de los dispositivos conectados y bloquear el acceso desde ciertos dispositivos	✓

Fuente: (OWASP, 2018, pág. 19)

3.2.4.4. Requerimientos de comunicación a través de la red

Esta categoría pretende “asegurar la confidencialidad e integridad de la información intercambiada entre la aplicación móvil y los servicios del servidor. Como mínimo se deben utilizar canales seguros y cifrados” (OWASP, 2018, pág. 21).

#	Descripción	L2
1	La información es enviada cifrada utilizando TLS. El canal seguro es usado consistentemente la aplicación	✓
2	Las configuraciones del protocolo TLS siguen las mejores prácticas o tan cerca posible mientras que el sistema operativo del dispositivo lo permite	✓
3	La aplicación vertical certificado X.509 del servidor al establecer el canal seguro y no sólo se aceptan certificados firmados por una CA válida	✓
4	La aplicación utiliza su propio almacén de certificados o realiza una fijación del certificado o la clave pública del servidor y no establece una conexión con servidores que ofrecen otros certificados o clave por más que estén firmados por una CA confiable	✓
5	La aplicación no depende de un único canal de comunicación inseguro (e-mail o SMS) para operaciones críticas como registros o recuperación de cuentas	✓

6	La aplicación sólo depende de bibliotecas de conectividad y seguridad actualizadas	✓

Fuente: (OWASP, 2018, pág. 21)

3.2.4.5. Requerimientos de interacción con la plataforma

Estos controles tienen el propósito de verificar que las APIs de la plataforma y componentes estándar se usen de forma segura.

#	Descripción	L2
1	La aplicación requiere la mínima cantidad de permisos	✓
2	Toda entrada del usuario y fuentes externas es válida y si es necesario ser sanitizada. Esto incluye información recibida por la UI, y mecanismo IPC como los intents, URLs y fuentes de la red	✓
3	La aplicación no exporta funcionalidades sensibles vía esquemas de URL, salvo que dichos mecanismos estén debidamente protegidos.	✓
4	La aplicación no exporta funcionalidades sensibles a través de mecanismos IPC salvo que los mecanismos estén debidamente protegidos.	✓
5	JavaScript se encuentra deshabilitado en los WebViews salvo que sea necesario.	✓
6	Los WebViews se encuentran configurados para permitir el mínimo de los manejadores (idealmente, solo https). Manejadores peligrosos como file, tel y app-id se encuentran deshabilitados	✓
7	Si objetos nativos son expuestos en WebViews, verificar que solo se cargan JavaScripts contenidos del paquete de la aplicación.	✓
8	Serialización de objetos, si se realiza, se implementa utilizando API seguras.	✓

Fuente: (OWASP, 2018, pág. 22)

3.2.4.6. Requerimientos de calidad de código y configuración del compilador

El propósito de estos controles es asegurar que fueron seguidas las prácticas básicas de seguridad en el desarrollo de la aplicación.

#	Descripción	L2
1	La aplicación es firmada y provista con un certificado válido.	✓
2	La aplicación fue liberada en modo release y con las configuraciones apropiadas para el mismo (ej. non-debuggable).	✓
3	Los símbolos de debug fueron removidos de los binarios nativos.	✓

4	La aplicación no contiene código de prueba y no realiza log de errores o mensajes de debug	✓
5	Todos los componentes de terceros se encuentran identificados y revisados por vulnerabilidades conocidas.	✓
6	La aplicación captura y maneja debidamente las posibles excepciones.	✓
7	Los controles de seguridad deniegan el acceso por defecto.	✓
8	En código no administrado, la memoria es pedida, usada y liberada de manera correcta.	✓
9	Funcionalidades de seguridad gratuitas se encuentran activadas	✓

Fuente: (OWASP, 2018, pág. 23)

3.2.5. Metodología OASAM

OASAM es el acrónimo de Open Android Security Assessment Methodology[7] (Ver Figura 13), que traducido al español corresponde a: Metodología Abierta de Evaluación de Seguridad de Android; el objetivo manifiesto de esta metodología es convertirse en un marco de referencia en las evaluaciones de vulnerabilidad de aplicaciones de Android.

Figura 13 Open Android Security Assessment Methodology (OASAM)

La consecuencia de que Android sea el sistema operativo más extendido, es que las aplicaciones para este SO sean las que mayor distribución tienen, sin embargo este auge no se corresponde con las condiciones de seguridad que se evidencian en las aplicaciones, y por otro lado, “a pesar de que hay trabajos particulares en este sentido, no existe una taxonomía completa y consistente de vulnerabilidades

[7] La información de referencia sobre la metodología OASAM se puede ampliar en https://github.com/b66l/OASAM

en aplicaciones específicas de Android que permita catalogar tales vulnerabilidades" (OASAM, 2016).

Por las anteriores razones es que se funda el proyecto OASAM, que pretende convertirse en una taxonomía completa y consistente de las vulnerabilidades del SO Android, configurándose como framework que respalde y dé soporte a los desarrolladores de aplicaciones, y a las personas encargadas de buscar vulnerabilidades en ellas.

La Metodología Abierta de Evaluación de Seguridad de Android (OASAM) contiene una serie de controles de seguridad para las aplicaciones Android. Para efectos del cumplimiento de los objetivos de este trabajo describiremos los elementos esenciales de la metodología OASAM, dejando la información requerida para tener acceso a los procedimientos en caso que el lector quiera profundizar en ellos[8].

Los controles de seguridad que se presentan a continuación son una traducción de la Metodología, la cual está estructurada por categorías y cada una se corresponde con un ámbito de evaluación.

3.2.5.1. OASAM-INFO: Recopilación de información

La recopilación de información referente a la aplicación es una de las partes más importantes en el proceso de evaluación de las aplicaciones Android, es en esta etapa en la que se definen los parámetros del ataque.

OASAM-INFO 001: Información general de la aplicación

La información general sobre la aplicación que se recopilan esta etapa la siguiente:

- ✓ Nombre de la versión de la aplicación;
- ✓ Número de versión de la aplicación;
- ✓ Fecha de instalación de la aplicación;

[8] La Metodología Abierta de Evaluación de Seguridad de Android (OASAM) está disponible en el sitio web https://github.com/b66l/OASAM

- ✓ Fecha de actualización más reciente;
- ✓ Ruta de la aplicación en el dispositivo;
- ✓ Bibliotecas compartidas;
- ✓ Ruta de acceso de APK en el dispositivo;
- ✓ SDK de destino de la aplicación;
- ✓ ¿Está habilitado el modo Isdebugging?
- ✓ Permisos requeridos;
- ✓ Intención de lanzamiento de la aplicación: Intento necesario para iniciar la aplicación.

Riesgo	Recomendación
Un atacante extraerá información para adquirir conocimiento sobre la aplicación mientras que se requiere información global para el funcionamiento de la aplicación, por lo que su exposición en sí misma no constituye un riesgo para la seguridad. El valor de cualquiera de los parámetros puede convertirse en un riesgo menor; por ejemplo, si el modo de depuración está habilitado.	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el acceso a dicha información. Algunos parámetros pueden comprometer la seguridad, por ejemplo, si la aplicación tiene habilitado el modo de depuración, que se recomienda evitar.

OASAM-INFO 002: Lista de componentes de la aplicación

En esta etapa se recopila la información básica sobre los componentes de una aplicación, tal como:

- ✓ Ocupaciones
- ✓ Servicios
- ✓ Proveedores de contenido
- ✓ Receptores de difusión

Riesgo	Recomendación
Un atacante obtendrá la información para adquirir conocimiento sobre la aplicación, mientras que la lista de componentes de una aplicación es necesaria para ejecutar dicha aplicación, por lo que su exposición en sí misma no constituye un riesgo para la seguridad.	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el acceso a dicha información.

OASAM-INFO 003: Permisos de componentes de la aplicación

Los permisos de acceso necesario se pueden definir para cada componente de la aplicación, un posible atacante mostrará una lista de permisos para acceder a componentes funcionales que no tienen las restricciones correspondientes.

Riesgo	Recomendación
Se puede acceder a un componente sin permisos definidos desde una aplicación sin restricciones, incluso se puede acceder desde cualquier otra aplicación del dispositivo si se exporta.	Se recomienda establecer permisos para todos los componentes que ejecutan funcionalidades confidenciales.

OASAM-INFO 004: Componentes exportados

Los componentes de la aplicación pueden ser exportados si la bandera “exportadas” en AndroidManifest está habilitada. Cuando un componente es exportado puede iniciarse desde cualquiera aplicación del dispositivo.

Riesgo	Recomendación
Si la funcionalidad del componente es sensible o necesita permisos habilitados en la aplicación, un componente exportado incorrectamente podría permitir la ejecución de dicha funcionalidad sensible desde otra aplicación hasta su lanzamiento.	Se recomienda exportar los componentes solo cuando sea estrictamente necesario. Al exportar un componente, se recomienda establecer el permiso, por lo que el lanzamiento necesitará un grado de seguridad adicional.

OASAM-INFO 005: Intenciones de lanzamiento de componente

Un atacante buscará la intención necesaria para ejecutar el componente. Por lo tanto, si el componente es accesible (exportado en el caso de otras aplicaciones del dispositivo), podrá iniciarlo desde otra aplicación o componente.

Riesgo	Recomendación
Un atacante obtendrá la información para adquirir conocimiento sobre la aplicación, mientras que la lista de Intenciones de lanzamiento de un componente de la aplicación es necesaria para ejecutar dicha aplicación, por lo que su exposición en sí misma no constituye un riesgo para la seguridad.	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el acceso a dicha información.

3.2.5.2. OASAM-CONF: Configuración e implementación de evaluación

Los errores en la configuración de las aplicaciones o los componentes son un compromiso serio para la seguridad de la aplicación. En esta etapa de la metodología se definen varios errores que pueden cometerse ya sea en la configuración o en las opciones de implementación de la aplicación.

Con el propósito de evaluar la incidencia de tales errores es necesario aplicar los controles que se listan en esta sección.

OASAM-CONF-001: Depuración sin restricciones

Es habitual que el modo de depuración esté habilitado al desarrollar una aplicación esto con el propósito de extraer información sobre su funcionamiento, pero es recomendable que éste no esté desactivado el desarrollar aplicaciones

Riesgo	Recomendación
La aplicación en el modo de depuración proporciona información que un atacante podría usar para realizar ataques en dicha aplicación.	Se recomienda establecer la opción de depurar en falso o eliminarla en el archivo AndroidManifest.xml, debido a que una aplicación desactiva el modo de depuración de manera predeterminada

OASAM-CONF-002: Uso de bibliotecas no actualizadas

El uso de bibliotecas desactualizadas u obsoletas aumenta la exposición a vulnerabilidades que están en capacidad de poner en riesgo la aplicación y los datos procesados por ésta.

Riesgo	Recomendación
Es más probable que las bibliotecas desactualizadas contengan vulnerabilidades que pueden poner en riesgo la aplicación y los datos procesados por dicha aplicación.	Se recomienda utilizar las últimas versiones estables proporcionadas por los proveedores y / o desarrolladores de cada biblioteca utilizada en las aplicaciones.

OASAM-CONF-003: Archivos predeterminados y de respaldo

En esta etapa, el atacante busca versiones antiguas de los archivos modificados, incluye archivos que se han cargado en el lenguaje de programación en uso y que

se pueden descargar como código fuente o incluso archivos automáticos o manuales de respaldo como archivos comprimidos. En el caso de Android, estos archivos se pueden dejar por error en la estructura del archivo APK.

Riesgo	Recomendación
Todos estos archivos podrían permitir que un atacante tenga acceso a operaciones internas, puertas traseras, interfaces administrativas o incluso credenciales para conectarse a la interfaz administrativa o al servidor de la base de datos.	Antes de empaquetar una aplicación de Android, se recomienda asegurar que los archivos que pertenecen a la aplicación sean estrictamente necesarios y que no se incluya ningún archivo que contenga información innecesaria o confidencial.

OASAM-CONF-004: Metadatos sobre los archivos

En esta etapa, un atacante examinará los metadatos de los archivos incluidos en la aplicación para buscar información útil durante el ataque de penetración. En el caso de Android, estos archivos se encuentran en la estructura del archivo APK.

Riesgo	Recomendación
Dependiendo de la información almacenada en los metadatos, el riesgo puede variar mucho. Los metadatos típicos que se pueden encontrar en los archivos de las aplicaciones de Android son los relacionados con las imágenes contenidas en las aplicaciones.	Se recomienda eliminar los metadatos de los archivos incluidos en la aplicación para no proporcionar información innecesaria. Existen herramientas de código abierto que permiten eliminar metadatos en muchos formatos diferentes; por ejemplo, Exiftool

OASAM-CONF-005: Endurecimiento de WebView insuficiente

WebView es una extensión de Android que permite mostrar contenido en línea dentro de las Actividades de Android. Sin embargo, al instalar WebView en aplicaciones de Android, se debe tener en cuenta el hecho de que una configuración deficiente de esta extensión podría exponer al usuario de la aplicación a una multitud de riesgos.

Riesgo	Recomendación
Un endurecimiento	hay un conjunto de prácticas que deben tenerse en cuenta para reforzar WebView:

insuficiente puede implicar riesgos para el usuario, convirtiéndose en posibles objetivos para ataques a través de la web	✓ Inhabilitar la compatibilidad con los complementos del navegador; por ejemplo, con la siguiente instrucción: webview.getSettings (). setPluginsEnabled (false); ✓ Deshabilitar el acceso a archivos locales; por ejemplo, con la siguiente instrucción: webview.getSettings (). setAllowFileAccess (false); ✓ Evite cargar contenido de sitios de terceros, verificando que WebView solo pueda acceder a esos sitios web que la aplicación necesita, utilizando listas blancas para dicho efecto cuando sea posible. ✓ Deshabilitando JavaScript, con la siguiente instrucción: webview.getWebSettings (). SetJavaScriptEnabled (false); ✓ Desactiva el acceso a la URL de contenido dentro de WebView. El acceso a la URL de contenido permite que WebView cargue contenido de un proveedor de contenido instalado en el sistema., Con la siguiente declaración: webview.getWebSettings (). SetsetAllowContentAccess (false); ✓ Establece si JavaScript que se ejecuta en el contexto de una URL de esquema de archivos debe tener acceso a contenido de otras URL de esquema de archivos, con la siguiente instrucción: webview.getWebSettings (). SetAllowFileAccessFromFileURLs (false); ✓ Establece si JavaScript que se ejecuta en el contexto de un esquema de archivo URL debe tener acceso a contenido de cualquier origen, con la siguiente instrucción: webview.getWebSettings (). SetAllowUniversalAccessFromFileURLs (false);

OASAM-CONF-006: Permisos de archivos incorrectos

La generación de archivos con el permiso "MODE_WORLD_READABLE" permite la lectura de un archivo global, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda configurar archivos con el permiso "MODE_WORLD_WRITABLE". Un ejemplo de declaración de archivo peligroso se puede ver a continuación:

file = openFileOutput ("File_Name", Context.MODE_WORLD_READABLE);

Riesgo	Recomendación
Establecer permisos de lectura globales	Se recomienda generar archivos con permisos globales de lectura o escritura solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos:

revela la información contenida en un archivo. Si el permiso de escritura está habilitado, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	✓ Preferencias compartidas. Para almacenar opciones como pares de valores. ✓ Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. Por defecto, este tipo de almacenamiento no es accesible por aplicaciones de terceros. ✓ Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. ✓ Base de datos SQLite Para almacenar datos en BBDD SQLite. Por defecto, estos tipos de bases de datos no son accesibles por aplicaciones de terceros. ✓ Conexión de red. Para almacenar datos sobre servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de encriptación.

OASAM-CONF-007: Permisos de proveedor de contenido inapropiado

En Android, los proveedores de contactos son una forma de compartir información entre las aplicaciones a través de una API estructurada que permite mantener la integridad de los datos. También es posible establecer permisos de acceso y escritura para los proveedores de contenido. Si estos permisos no están configurados, cualquier aplicación podría tener acceso a los datos almacenados.

Riesgo	Recomendación
Una aplicación maliciosa podría acceder y / o modificar los datos almacenados en un proveedor de contenido sin la conciencia del usuario, lo que afecta la integridad y confidencialidad de dichos datos.	Se recomienda establecer permisos para los proveedores de contenido. Para tal fin, se debe usar la guía de "permiso de uso"; por ejemplo: <uses-permission android: name = "android.permission.READ_USER_DICTIONARY">

OASAM-CONF-008: Permisos de actividades incorrectas

En esta etapa, el atacante enumerará los permisos necesarios para iniciar cada actividad. Con esta información, es posible extraer la superficie de ataque tanto

desde dentro como fuera del entorno de la aplicación, por lo que el atacante podría aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, una actividad que realiza funcionalidades que requieren permisos especiales y que se puede iniciar fuera de la aplicación (exportada) y no requiere permisos, compromete la seguridad debido al hecho de que permite que cualquier aplicación del dispositivo inicie la actividad.

Riesgo	Recomendación
Los riesgos varían según la sensibilidad de la funcionalidad de cada actividad.	Como regla general, se recomienda solicitar permisos sobre las actividades que realizan funciones confidenciales, especialmente si se exportan actividades

OASAM-CONF-009: Permisos de servicios inadecuados

En esta etapa, el atacante enumerará los permisos necesarios para iniciar cada servicio. Con esta información, es posible obtener la superficie de ataque tanto dentro como fuera del entorno de la aplicación, por lo que el atacante podría aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, un servicio que realiza funcionalidades que requieren permisos especiales y que puede iniciarse fuera de la aplicación (exportada) y no requiere permisos, compromete la seguridad debido al hecho de que permite que cualquier aplicación de dispositivo inicie el servicio

Riesgo	Recomendación
Los riesgos varían según la sensibilidad de la funcionalidad de cada servicio.	Como regla general, se recomienda solicitar permisos en los servicios que realizan funciones confidenciales, especialmente si se exportan servicios

OASAM-CONF-010: Permisos de receptores de difusión inapropiados

En esta etapa, el atacante enumerará los permisos necesarios para iniciar cada receptor de difusión. Con esta información, es posible obtener la superficie de ataque tanto dentro como fuera del entorno de la aplicación, por lo que el atacante podría aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, un receptor de difusión que realiza funcionalidades que requieren permisos

especiales y que puede lanzarse fuera de la aplicación (exportada) y no requiere permisos, compromete la seguridad debido al hecho de que permite que el receptor de difusión sea lanzado por cualquier aplicación de dispositivo.

Riesgo	Recomendación
Los riesgos varían según la sensibilidad de la funcionalidad de cada receptor de emisión.	Como regla general, se recomienda solicitar permisos a los receptores de difusión que realizan funcionalidades confidenciales, particularmente si son receptores de difusión exportados.

OASAM-CONF-011: Permisos de base de datos incorrectos.

La generación de bases de datos con el permiso "MODE_WORLD_READABLE" habilitado permite la lectura de la base de datos global, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer bases de datos con el permiso "MODE_WORLD_WRITABLE". Un ejemplo de declaración de archivo peligroso se puede ver a continuación:

SQLiteDatabasemyWorldReadDB = openOrCreateDatabase ("Contactos", MODE_WORLD_READABLE, nulo);

Riesgo	Recomendación
Establecer permisos de lectura globales revela la información contenida en un archivo. Si se proporciona el permiso de escritura, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la	Se recomienda generar archivos con permisos globales de lectura o escritura solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: ✓ Preferencias compartidas. Para almacenar opciones como pares de valores. ✓ Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. Por defecto, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . ✓ Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. ✓ Base de datos SQLite Para almacenar datos en BBDD SQLite. Por defecto, estos tipos de bases de datos no son accesibles por aplicaciones de terceros.

superficie de ataque.	✓ Conexión de red. Para almacenar datos sobre servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de encriptación.

OASAM-CONF-012: Permisos de preferencias compartidas incorrectos

La generación de preferencias compartidas con el permiso "MODE_WORLD_READABLE" permite una lectura global de Preferencias compartidas, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer Preferencias compartidas con el permiso "MODE_WORLD_WRITABLE". Un ejemplo de declaración de archivo peligroso se puede ver a continuación:

SharedPreferencesprefsAllWrite = getSharedPreferences ("MisPreferenciasWrite", MODE_WORLD_WRITEABLE);

Riesgo	Recomendación
Establecer permisos de lectura globales revela la información contenida en un archivo. Si se proporciona el permiso de escritura, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	Se recomienda generar archivos con permisos globales de lectura o escritura solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: ✓ Preferencias compartidas. Para almacenar opciones como pares de valores. ✓ Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. Por defecto, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . ✓ Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. ✓ Base de datos SQLite Para almacenar datos en BBDD SQLite. Por defecto, estos tipos de bases de datos no son accesibles por aplicaciones de terceros. ✓ Conexión de red. Para almacenar datos sobre servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y usar técnicas de encriptación

3.2.5.3. OASAM-AUTH: Evaluación de autenticación

En esta sección, se verificarán las funcionalidades relacionadas con el uso de inicios de sesión a través de la aplicación. Tenga en cuenta que los casos de vulnerabilidad se buscarán dentro de la aplicación de Android. Si la autenticación se lleva a cabo

contra un tercero (servicio web, servicio REST, etc.), no se evaluará la seguridad de terceros, solo las debilidades relacionadas con la aplicación de Android.

3.2.5.4. OASAM-CRYPT: Evaluación de uso de criptografía

En esta sección, se prueban las funcionalidades relacionadas con el uso de criptografías en la aplicación. Esto puede ocurrir al enviar o almacenar datos.

Los siguientes controles se aplican en esta sección:

OASAM-CRYPT-001: credenciales codificadas

El uso de contraseñas codificadas o vacías es fácil de extraer para un usuario malintencionado mediante el desmontaje de la aplicación.

Riesgo	Recomendación
Un usuario malintencionado podría tener acceso a credenciales codificadas, accediendo a los servicios permitidos por tales credenciales.	Se recomienda almacenar información confidencial en lugares a los que solo puede acceder la aplicación y, preferiblemente, encriptar dicha información

OASAM-CRYPT-002: Almacenamiento de datos inseguros

El cifrado no es suficiente para garantizar la confidencialidad de la información. Es necesario utilizar las funciones de cifrado de forma adecuada, utilizando algoritmos más robustos y la mayor longitud de clave posible. Por ejemplo, usar AES como algoritmo de encriptación es posible en Android, pero se recomienda usar una clave de 256 bits en lugar de 128 bits.

Riesgo	Recomendación
Un usuario malintencionado podría realizar ataques contra la criptografía en uso. Tenga en cuenta que un usuario con privilegios de administrador podría consultar la memoria de los procesos del dispositivo, pudiendo acceder a toda la información no encriptada.	Se recomienda utilizar AES con claves de 256 bits. Además, se recomienda generar IV vectores a través de SecureRandom y la clave debe derivar de una contraseña utilizando protocolos conocidos como PBKDF2 (Función de Derivación de Claves Basada en Contraseña).

OASAM-CRYPT-003: uso inseguro del protocolo de transporte

Las comunicaciones de información a través de canales HTTP se pueden interceptar fácilmente. En el caso de usar HTTPS, un atacante podría usar técnicas SSLStrip para acceder a dicha información, haciendo necesario el uso de mecanismos adicionales como forzar, desde el lado del cliente, para permitir el envío de información solo a través de un canal HTTPS seguro.

Riesgo	Recomendación
Un usuario malintencionado podría realizar ataques contra la criptografía en uso. En el caso de usar HTTPS, existen varios tipos de ataques; el más conocido es SSLStrip. Si el atacante tiene éxito, se puede acceder a la información que se supone que se envió cifrada.	Se recomienda utilizar HTTPS en lugar de HTTP siempre que sea posible. Además, se recomienda utilizar mecanismos del lado del cliente que fuercen el envío de información a través de un canal seguro; por ejemplo, redirigir a un canal seguro si se detecta un intento de enviar información a través de un canal no encriptado. Además, se recomienda evaluar la necesidad de utilizar encabezados de Seguridad de transporte HTTP estricto en servidores web; con tales encabezados, el navegador forzaría el uso de canales encriptados en las comunicaciones.

OASAM-CRYPT-004: fijación de certificados

Si la aplicación intercambia información a través de SSL y no confía en las autoridades de certificación, no hay razón para usar esta confiabilidad porque no brindan seguridad a las comunicaciones de la aplicación. Además, si alguna de estas autoridades de certificación se ve comprometida, los usuarios de la aplicación no serían vulnerables.

Riesgo	Recomendación
Si una autoridad de certificación está en peligro, el envío de certificados fraudulentos podría tener un impacto en la confidencialidad de la información transmitida, debido al hecho de que la aplicación confiaría en estos certificados porque los proporciona una autoridad certificadora conocida.	En caso de utilizar solo SSL para servicios definidos, se recomienda incluir técnicas de fijación de certificados. El navegador Google Chrome ya los usa para los servicios de Google y también para la aplicación oficial de Twitter.

3.2.5.5. OASAM-LEAK: Evaluación de fuga de información confidencial

En la sección de autenticación, se verificará la filtración de información en los medios. La información confidencial podría estar relacionada con el usuario o el teléfono. Los siguientes controles se aplican en esta sección:

OASAM-LEAK-001: fuga de información para registrar archivos.

Es posible almacenar información en los registros de Android a través de las siguientes funciones:

- ✓ Log.v (). Para almacenar información extendida.
- ✓ Log.d (). Para almacenar información de depuración.
- ✓ Log.i (). Para almacenar información básica.
- ✓ Log.w (). Para almacenar información sobre alertas.
- ✓ Log.e (). Para almacenar registros de errores.

No se recomienda almacenar información confidencial en los registros del dispositivo porque cualquier aplicación con el permiso "READ_LOGS" podría tener acceso a dichos registros.

Riesgo	Recomendación
Cualquier aplicación con acceso a los registros podría extraer información confidencial que podría almacenarse en esos registros	Se recomienda almacenar información confidencial en lugares a los que solo puede acceder la aplicación y, preferiblemente, encriptar dicha información. En términos generales, estas pautas se deben seguir para almacenar datos: ✓ Preferencias compartidas. Para almacenar opciones como pares de valores. ✓ Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. Por defecto, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . ✓ Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. ✓ Base de datos SQLite Para almacenar datos en BBDD SQLite. Por defecto, estos tipos de bases de datos no son accesibles por aplicaciones de terceros.

	✓ Conexión de red. Para almacenar datos sobre servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de encriptación.

OASAM-LEAK-002: Fuga de información a SDCard.

Es posible almacenar datos en tarjetas SD a través de las siguientes funciones:

- ✓ Con API 7 o inferior. A través de la función getExternalStorageDirectory ().
- ✓ Con API8 o superior. A través de la función getExternalFilesDir ().

No se recomienda almacenar datos confidenciales en la tarjeta SD del dispositivo, ya que cualquier aplicación podría extraerlos.

Riesgo	Recomendación
Cualquier aplicación podría tener acceso a los datos confidenciales almacenados en la tarjeta SD.	Se recomienda almacenar información confidencial en lugares a los que solo puede acceder la aplicación y, preferiblemente, encriptar dicha información. En términos generales, se deben seguir las siguientes pautas para almacenar datos: ✓ Preferencias compartidas. Para almacenar opciones como pares de valores. ✓ Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. Por defecto, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . ✓ Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. ✓ Base de datos SQLite Para almacenar datos en BBDD SQLite. Por defecto, estos tipos de bases de datos no son accesibles por aplicaciones de terceros. ✓ Conexión de red. Para almacenar datos sobre servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de encriptación.

OASAM-LEAK-003: Fuga de información a la red

Para enviar datos a la red, es necesario tener activado el permiso "android.permission.INTERNET". Una aplicación con tal permiso puede enviar información a la red a través de muchas bibliotecas; los más comunes son java.net. *y android.net.*. No se recomienda enviar información sensible no encriptada ya que un atacante podría extraerla olfateando el tráfico de la red.

Riesgo	Recomendación
Un atacante podría obtener acceso a la información delicada sin encriptar enviada a la red olfateando.	Se recomienda almacenar información confidencial en lugares a los que solo puede acceder la aplicación y, preferiblemente, encriptar dicha información. En términos generales, estas pautas se deben seguir para almacenar datos: ✓ Preferencias compartidas. Para almacenar opciones como pares de valores. ✓ Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. Por defecto, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . ✓ Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. ✓ Base de datos SQLite Para almacenar datos en BBDD SQLite. Por defecto, estos tipos de bases de datos no son accesibles por aplicaciones de terceros. ✓ Conexión de red. Para almacenar datos sobre servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de encriptación.

OASAM-LEAK-004: Fuga de información al IPC de Android

Es posible almacenar información en Intents a través de las funciones putExtra () y putExtras () de la clase Intent, así como a través del URI en los parámetros mediante el método GET a través del análisis sintáctico () de la clase URI. No se recomienda enviar datos confidenciales sin encriptar a través de Intenciones implícitas ya que la aplicación que procesará dicha información se desconoce a priori.

Riesgo	Recomendación
Una aplicación ilegítima podría tener acceso a datos confidenciales al interceptar el Intento.	Se recomienda almacenar información confidencial en lugares a los que solo puede acceder la aplicación y, preferiblemente, encriptar dicha información. En términos generales, estas pautas se deben seguir para almacenar datos: ✓ Preferencias compartidas. Para almacenar opciones como pares de valores. ✓ Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. Por defecto, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . ✓ Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD.

	✓ Base de datos SQLite Para almacenar datos en BBDD SQLite. Por defecto, estos tipos de bases de datos no son accesibles por aplicaciones de terceros. ✓ Conexión de red. Para almacenar datos sobre servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de encriptación

3.2.5.6. OASAM-DV: Evaluación de gestión de entrada de usuario

En esta categoría, se incluirán las vulnerabilidades relacionadas con la entrada recibida del usuario que administra la aplicación. La validación incorrecta de los datos del usuario es uno de los principales vectores de ataque, debido a que podría permitir que un atacante altere el flujo de datos de la aplicación, inyectando código y afectando de manera grave la aplicación y los datos almacenados en él. Esta categoría ocupa el 4 ° lugar en los 10 principales riesgos de la aplicación móvil titulada "Inyección del lado del cliente". A pesar de que el origen de este tipo de vulnerabilidad es el mismo, dependiendo de la ubicación de la interacción de los datos recibidos del usuario, pueden tener lugar diferentes tipos de vulnerabilidades con múltiples impactos.

Los siguientes controles se aplican en esta categoría:

OASAM-DV-001: Inyección de código HTML.

Las vulnerabilidades de cross site scripting (XSS) son el resultado de una codificación incorrecta de los datos recibidos de fuentes inseguras antes de usarlos en la salida HTML. En el caso de Android, la vulnerabilidad se deriva básicamente del uso de la función "setJavaScriptEnabled ()" de la clase WebView. Esta función permite incluir objetos Java en el contexto JavaScript de WebView, de modo que si los datos proporcionados a dicha función no se desinfectan, se podría ejecutar código JavaScript arbitrario.

Riesgo	Recomendación
Las vulnerabilidades de cross site scripting (XSS) permiten a un atacante ejecutar código de script (HTML, JavaScript, VBScript) en el	Como regla general, la información recibida del usuario no debe ser confiable. Es peligroso utilizar los datos recibidos directamente del usuario sin una codificación adecuada en la

navegador del usuario (en el dominio de la aplicación vulnerable). Por ejemplo, un atacante podría explotar una vulnerabilidad XSS para extraer cookies de usuario (para suplantar a la víctima en la aplicación vulnerable después) o solicitar información a los usuarios (como credenciales). Esta vulnerabilidad se produce como resultado del uso de información insegura en un resultado HTML sin una codificación adecuada y previa.	generación de la página de respuesta. De forma predeterminada, WebView no ejecuta JavaScript, por lo que debe habilitarse solo cuando sea estrictamente necesario, controlando el origen de la carga dinámica del código a ejecutar. Sin embargo, para deshabilitar expresamente la ejecución de JavaScript, se recomienda ingresar la siguiente declaración: webview.getSettings (). SetJavaScriptEnabled (false); En el caso de utilizar la función "addJavascriptInterface ()", se recomienda filtrar los objetos Java proporcionados a WebView a través de listas blancas cuando sea posible.

OASAM-DV-002: Desbordamiento de búfer.

Las vulnerabilidades de desbordamiento del búfer son el resultado de no verificar correctamente el tamaño de ciertos datos antes de copiarlos a un área de memoria en particular. En el caso de Android, este tipo de error es fácil de confirmar al usar código nativo si el tamaño del búfer de datos no está validado. Aunque Android incorpora protecciones como ASLR y DEP que reducen la capacidad de explotación, no resuelven el problema fundamental.

Riesgo	Recomendación
Los riesgos asociados a este tipo de error se consideran muy graves para copiar información en un área de memoria que estaba reservada para otras tareas y para almacenar otros datos del programa. Cuando se produce un desbordamiento, los resultados son impredecibles y provoca el final de la ejecución del programa como regla general, y también es posible causar una denegación de servicio. Al explotar tales errores, un atacante podría controlar la ejecución del programa, pudiendo acceder al sistema de forma remota por un atacante experimentado.	Como regla general, cualquier operación que implique la copia de datos en un área de memoria se debe realizar cuidadosamente para garantizar que el área de memoria objetivo tenga suficiente memoria para contener los datos para copiar. Existe un conjunto de funciones que tienen más probabilidades de causar errores y no son recomendables.

OASAM-DV-003: Inyección de comandos en bases de datos.

Las vulnerabilidades de inyección SQL pueden ocurrir cuando la información proveniente de contenedores inseguros se usa para construir consultas SQL dinámicas sin implementar controles y / o filtros relevantes para dicha información.

Los contenedores inseguros son cualquier fuente de datos del código fuente, como datos proporcionados por el usuario, bases de datos no controladas, servicios web, etc. Android usa SQLite para almacenar datos. Un ejemplo de código vulnerable se muestra a continuación:

String con = "DELETE FROM case_values WHERE _id =" + paramString;

localSQLiteDatabase.execSQL (con);

Riesgo	Recomendación
Las vulnerabilidades de inyección de SQL podrían permitir que un atacante use caracteres especiales en el lenguaje SQL para alterar una declaración que está construida con información externa de una manera dinámica. Como resultado, podría ser posible obtener información arbitraria de la base de datos, alterar esta información y alterar el flujo de trabajo de la aplicación afectada	Se recomienda validar cada campo a través de listas blancas. Por lo tanto, solo se aceptarán personajes no peligrosos. Es importante normalizar la entrada para evitar evasiones a través de codificaciones de parámetros. Si es posible, se recomienda utilizar declaraciones preparadas, ya que es la opción más segura para construir sentencias de SQL con parámetros externos

OASAM-DV-004: Inyección de ruta en el acceso a archivos.

Si la generación de archivos depende de la entrada del usuario, es necesario filtrar estos datos porque, de lo contrario, un usuario malintencionado podría tener acceso a información arbitraria mediante el uso de los permisos de la aplicación vulnerable. A continuación, se muestra un ejemplo de generación de archivos peligrosos:

file = openFileOutput (unfiltered_variable, Context.MODE_WORLD_READABLE);

Si la "variable no filtrada" proviene del usuario sin filtro, el acceso a los archivos por parte de la aplicación podría controlarse, permitiendo que un usuario malintencionado sobrescriba los archivos de la aplicación.

Riesgo	Recomendación
Esta vulnerabilidad pone en riesgo la integridad y la disponibilidad de los archivos administrados por la aplicación.	Se recomienda no utilizar datos provenientes del usuario para crear nombres de archivo. Si es necesario, se recomienda filtrar los datos de entrada, preferiblemente de acuerdo con el uso de listas blancas.

OASAM-DV-005: Verificación de parámetros nulos

Android finaliza las aplicaciones cuando se encuentra una referencia nula. Si las referencias nulas no se verifican al recuperar datos de un Intento, un atacante podría enviar intenciones con campos con valores nulos para denegar el servicio a las aplicaciones, especialmente aquellas que tienen servicios en ejecución. En particular, la existencia de referencias nulas debe verificarse al recuperar datos de un Intent, especialmente los siguientes métodos: getAction () y getExtra ().

Riesgo	Recomendación
Esta vulnerabilidad pone en riesgo la disponibilidad de las aplicaciones que no realizan comprobaciones para no intentar acceder a los valores nulos. En caso de que los servicios ejecuten aplicaciones, un atacante podría denegar dichos servicios sin que el usuario lo sepa, por lo que es especialmente dañino.	Se recomienda verificar los valores nulos al recuperar datos de Intents, especialmente los siguientes métodos: getAction () y getExtra ().

OASAM-DV-006: Inyección de registro

Un atacante podría incluir ciertos caracteres especiales en una cadena de texto especialmente manipulada, por lo que cuando se utiliza en la creación de una nueva entrada en los archivos de registro, dará como resultado una manipulación gratuita de la salida por parte del atacante. En el caso de Android, las funciones estándar de registro de información son las siguientes:

- ✓ Log.v (). Para almacenar información extendida.
- ✓ Log.d (). Para almacenar información de depuración.
- ✓ Log.i (). Para almacenar información básica.
- ✓ Log.w (). Para almacenar información sobre alertas.
- ✓ Log.e (). Para almacenar registros de errores.

Riesgo	Recomendación
Los archivos de registro podrían estar dañados debido a una entrada falsa o no válida. Esto podría dar como resultado registros poco confiables y la necesidad de descartar todos los registros sobre las acciones realizadas en el sistema por los	Se deben tener en cuenta todos los metacaracteres utilizados para la aplicación al generar registros. Por ejemplo, si la aplicación usa el carácter de la tubería "|" (sin las comillas) para formatear la entrada, este será un meta-carácter. Es decir, un

usuarios. Los atacantes podrían usar esta situación para ocultar acciones delictivas e impedir que se detecten sus acciones y así evitar su persecución por los crímenes cometidos.	atacante podría incluir un extra- | y sería imposible reconstruir después si eso | fue ingresado por la aplicación o el atacante. Esto se aplica a cualquier carácter especial en los registros.

OASAM-DV-007: Control del proceso de inyección a través de datos de intención.

Si la entrada del usuario se carga de forma dinámica en los datos de Intento, un usuario malintencionado podría manipular dichos datos para ejecutar código a través de él. En particular, la existencia de datos dinámicos debe verificarse al incluir dichos datos en un Intento, especialmente a través de los siguientes métodos de Intención:

- ✓ addCategory (),
- ✓ setAction (),
- ✓ setClass (),
- ✓ setClassName (),
- ✓ setComponent (),
- ✓ setData ()
- ✓ andsetDataAndType ().

Riesgo	Recomendación
Esta vulnerabilidad pone en riesgo la ejecución normal de la aplicación, ya que si un usuario es capaz de modificar los datos almacenados dentro de las Intenciones, también se podría modificar el rendimiento normal de la aplicación, así como la interacción entre las aplicaciones enviando Intenciones	Se recomienda verificar los valores de entrada del usuario, especialmente aquellos que van a los datos de Intento creando una instancia de la aplicación. Para tal fin, se recomienda inspeccionar los datos de usuario que terminan como parámetro en cualquiera de las siguientes funciones de intención: addCategory (), setAction (), setClass (), setClassName (), setComponent (), setData () y setDataAndType ().

3.2.5.7. OASAM-IS: Evaluación de la gestión de recepción intencional

Esta categoría cubre todas las vulnerabilidades involucradas en la entrega de Intenciones arbitrarias a un componente que espera recibir otro tipo de intención. De esta forma, el atacante utilizará los filtros de la víctima para enviar datos

inesperados y así aprovechar estas funcionalidades. Para tener éxito con este tipo de vulnerabilidad, los componentes del objetivo deben ser públicos o exportables como regla general; de lo contrario, no estarán disponibles para otras aplicaciones. Para tal fin, los componentes deben tener el indicador "EXPORTED" habilitado en el archivo AndroidManifest.xml. Tenga en cuenta que Broadcast Receivers podría declararse en AndroidManifest o en tiempo de ejecución.

Los siguientes controles se aplican en esta categoría:

OASAM-IS-001: Intento de suplantación en los componentes de difusión.

Los receptores de difusión se utilizan para registrar intenciones y responderlas. Si uno de estos componentes es público, cualquier aplicación podría enviar Intents. El registro de Intentos con acciones del sistema por parte de Broadcast Receivers es peligroso, ya que el hecho de registrar tales Intentos hace que el elemento sea público y accesible, de modo que una aplicación maliciosa podría enviar Intenciones explícitas sin las acciones correspondientes porque son acciones del sistema.

Riesgo	Recomendación
En muchos casos, los Receptores de difusión obtienen Intentos y activan Actividades o Servicios con sus datos. Las vulnerabilidades de inyección de difusión podrían permitir que un atacante manipule estos componentes para interactuar con ellos mediante la inyección arbitraria de datos.	Se recomienda utilizar componentes públicos solo cuando sea necesario y, en tales casos, validar completamente cada parámetro recuperado del Intento. Si los Receptores de difusión registran acciones del sistema, se recomienda validar la acción del Intento, ya que un Intento malicioso no puede registrar las acciones del sistema, por lo que esta validación se puede usar para saber si el Intento proviene del sistema Android o no.

OASAM-IS-002: Lanzamiento arbitrario de actividades

Las actividades se pueden iniciar sobre la base de los datos que provienen del usuario o la recepción de Intents arbitrarios. Esto permite varias superficies de ataque:

- ✓ Si una Actividad utiliza datos del Intento sin verificar su origen, la integridad de los datos de la aplicación se verá afectada.

- ✓ Si una actividad entrega datos considerados como confidenciales y se pueden lanzar públicamente, un intento especialmente manipulado podría lanzarlo, lo que llevaría a una divulgación de información confidencial.

Riesgo	Recomendación
La presencia de esta vulnerabilidad podría inducir a error a un usuario, debido a que los usuarios pensarían que están utilizando una Actividad legítima, cuando en realidad están realizando acciones que se informan al atacante. Además, es posible la filtración de la información utilizada por la Actividad.	Se recomienda utilizar componentes públicos solo cuando sea necesario y, en esos casos, validar minuciosamente cada parámetro recuperado del Intento o proveniente del usuario.

OASAM-IS-003: Lanzamiento arbitrario de Servicios.

Los servicios se pueden iniciar sobre la base de los datos que provienen del usuario o la recepción de Intents arbitrarios. Esto permite varias superficies de ataque:

- ✓ Si un Servicio utiliza datos del Intento sin verificar su origen, la integridad de los datos de la aplicación se verá afectada.
- ✓ Si un Servicio entrega datos considerados sensibles y se pueden lanzar públicamente, un Intento especialmente manipulado podría lanzarlo, lo que llevaría a la divulgación de información confidencial.

Riesgo	Recomendación
La presencia de esta vulnerabilidad podría permitir la realización de acciones arbitrarias en la aplicación al ejecutar los Servicios y también conducir a la divulgación de la información utilizada por el Servicio. Como circunstancia agravante, los Servicios generalmente confían más en los datos proporcionados por los usuarios y los Intents; además, proporcionan en muchos casos interfaces API para que las aplicaciones puedan comunicarse entre sí.	Se recomienda utilizar componentes públicos solo cuando sea necesario y, en esos casos, validar minuciosamente cada parámetro recuperado del Intento o proveniente del usuario. Además, los servicios se pueden proteger estableciendo permisos.

OASAM-IS-004: Debilidades relacionadas con el uso inseguro de Intentos Pendientes.

Una aplicación puede delegar tareas a otra aplicación a través de Intentos Pendientes. La aplicación que recibe la intención pendiente no puede modificar sus

datos, pero completa los datos vacíos. Si una aplicación genera un Intento Pendiente para tal fin pero el objetivo no está establecido, una aplicación que reciba el Intento Pendiente podría agregarlo, por lo que dichos datos se enviarían a un componente arbitrario.

Riesgo	Recomendación
Esta vulnerabilidad podría permitir la filtración de la información que está siendo procesada por la aplicación que generó el Intento Pendiente. Además, una aplicación maliciosa podría recibir, procesar y enviar dicha información, realizando un ataque Man-in-the-Middle.	Cuando las tareas se delegan en otra aplicación a través de una intención pendiente, se recomienda establecer siempre el destino al que se delegan las tareas para que las ejecute una aplicación de terceros.

3.2.5.8. OASAM-UIR: Evaluación de resolución de intención

Esta categoría cubre todas las vulnerabilidades relacionadas con la resolución de la entrega de intención implícita. Cuando una aplicación envía un Intento implícito, no hay garantía de que una aplicación maliciosa no recopile dicho Intento, ya que una aplicación maliciosa podría registrar un Filtro de Intento capaz de pasar la resolución (acción, datos y categoría), a menos que dicho Intento tenga un conjunto de los permisos necesarios que el usuario malicioso no tiene. Si una aplicación maliciosa puede interceptar Intents implícitos, podría tener acceso a la ejecución del flujo de datos, pudiendo realizar ataques de denegación de servicio o phishing. Esta categoría considera cómo se puede exponer este tipo de vulnerabilidad en componentes particulares: Difusión, Actividades y Servicios.

Los siguientes controles se aplican en esta categoría:

OASAM-UIR-001: Intercepción intencionada en los componentes de difusión.

Un Intento implícito es público cuando no está protegido con permisos de tipo "Firma" o "SignatureOrSystem". Cuando una aplicación envía un Intento implícito público, dicho Intento puede ser leído por cualquier componente del sistema. Una aplicación maliciosa podría tener acceso a ella, obteniendo datos confidenciales si el Intención simplemente lo contiene definiendo Filtros de Intención para todas las acciones, datos y categorías. Esto es especialmente peligroso en Sticky Broadcasts,

ya que el Intent persiste en ellos y puede ser redirigido repetidamente a varios destinatarios. Esto proporciona una mayor ventana de tiempo en la que el componente malicioso puede operar. Además, Sticky Broadcasts no puede protegerse a través de permisos.

Riesgo	Recomendación
Un atacante podría realizar un ataque de denegación de servicio en las transmisiones ordenadas, ya que un Intento solo se puede propagar en ellas si el primer componente que recibe el Intento lo usa para la salida. Además, podría usarse para realizar ataques Man-in-the-Middle con su posterior inyección de datos en Intents dispersos.	Se recomienda considerar la necesidad de enviar Intents implícitos. Para comunicar Intents en una Aplicación, siempre se recomienda el uso de Intents explícitos. Cuando es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos estableciendo permisos y encriptando dichos datos.

OASAM-UIR-002: Intercepción intencionada de los componentes de la actividad.

Al aprovechar esta vulnerabilidad, se lanza una actividad maliciosa en lugar de la esperada, por lo que el usuario estará en una aplicación incorrecta sin darse cuenta. Esto sucede cuando el cargo de una Actividad depende de un Intento implícito. El atacante registra un filtro de intención más preciso y lo controla. Esto no siempre es posible, ya que si varias Actividades pueden procesar un Intento implícito, aparecerá un mensaje para que el usuario elija la aplicación. Sin embargo, esto es muy peligroso porque si esto sucede y el usuario establece como acción predeterminada la carga de la aplicación maliciosa, la Actividad maliciosa siempre se abrirá sin que se le pregunte.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite ejecutar ataques de phishing, así como filtraciones de la información manejada por el usuario en la Actividad involucrada. Además, esta vulnerabilidad permite al atacante modificar los datos, poniendo en riesgo su integridad.	Se recomienda considerar la necesidad de enviar Intents implícitos. Para comunicar Intents en una Aplicación, siempre se recomienda el uso de Intents explícitos. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos estableciendo permisos y encriptando dichos datos.

OASAM-UIR-003: Intercepción intencionada en los componentes del servicio.

Al aprovechar esta vulnerabilidad, se lanza una actividad maliciosa en lugar de la esperada, por lo que el usuario estará en una aplicación incorrecta sin darse cuenta. Esto sucede cuando el cargo de una Actividad depende de un Intento implícito. El atacante registra un filtro de intención más preciso y lo controla. Esta vulnerabilidad es más persistente debido a que es transparente para el usuario porque los servicios no incluyen una interfaz gráfica para ella.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite la filtración de la información procesada por el Servicio involucrado. Además, esta vulnerabilidad permite a un atacante modificar datos, poniendo en riesgo su integridad.	Se recomienda considerar la necesidad de enviar Intents implícitos. Para comunicar Intents en una Aplicación, siempre se recomienda el uso de Intents explícitos. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos estableciendo permisos y encriptando dichos datos

OASAM-UIR-004: Intercepción de intención pendiente.

Si se envía un intento implícito para tratar con un proveedor de contenido, puede contener información confidencial en el URI, pudiendo leerse si un componente malicioso intercepta el intento. Por otro lado, una intención pendiente conserva los permisos de su creador, por lo que si un componente malicioso intercepta una intención pendiente, podría usar los permisos de la creación de intención pendiente para suplantarla.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite la filtración de información incluida en el URI. Además, esta vulnerabilidad permite al atacante suplantar los permisos del creador de Intención pendiente.	Se recomienda evaluar la necesidad de enviar Intents implícitos. Para comunicar Intents en una aplicación, se recomienda encarecidamente utilizar Intents explícitos. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos estableciendo permisos y encriptando dichos datos.

3.2.5.9. OASAM-BL Evaluación de la lógica de los negocios de la aplicación.

En esta categoría, se incluyen las vulnerabilidades con componentes más centrados en el diseño en lugar de la codificación. Se incluyen tanto la casuística del

rendimiento como la capacidad de la aplicación para funcionar de una manera inesperada que afecte su flujo de trabajo.

Los ataques dirigidos a las lógicas de negocios de las aplicaciones son peligrosos, difíciles de detectar y especialmente dirigidos a la aplicación.

Partiendo de los elementos anteriores y de los hallazgos encontrados a través de la revisión bibliográfica es necesario plantear la propuesta metodológica de seguridad para aplicaciones mHealth, el cual es el objetivo del siguiente subcapítulo

3.3. Propuesta de metodología de seguridad para aplicaciones mHealth para el Sistema Operativo Android

Para definir la metodología que es el propósito de este trabajo investigación se partió de la *Metodología Abierta de Evaluación de Seguridad de Android* (OASAM, por sus siglas en inglés), esta determinación se tomó basados en dos criterios, el primero, que es una de las pocas metodologías diseñadas exclusivamente para las aplicaciones Android, y el segundo, porque su objetivo manifiesto es convertirse en un marco de referencia en las evaluaciones de vulnerabilidad de aplicaciones de Android. Por lo cual resultaba muy apropiada para los intereses de este trabajo de investigación.

Posteriormente lo que se realizó fue incluir componentes de la metodología *Proyecto Abierto de Seguridad de Aplicaciones Web* (OWASP, por sus siglas en inglés), que es otra de las metodologías a las que tuvimos acceso y que tiene indicaciones exclusivas para la plataforma Android, en la metodología OASAM.

La manera como se construyó la metodología propuesta en este trabajo de investigación, fue la de realizar un análisis comparativo entre estas dos metodologías, e incluir componentes de la metodología OWASP, que fueran útiles para los requerimientos de seguridad de las aplicaciones mHealth en el compendio de pasos descritos en la metodología OASAM. Con lo cual, lo que se aportó a la metodología OASAM fue potenciar los pasos que, a criterio de los autores de este

trabajo de investigación, eran necesarios para adaptarla a los requerimientos de las aplicaciones mHealth.

Información de la aplicación

Recopilación de la información.

La recopilación de información de la aplicación es una de las partes más importantes en las evaluaciones de aplicaciones de Android. En esta etapa, se define la superficie de ataque, lo que motiva la prueba de vulnerabilidad que serán definidos por los controles.

Obtener la Definición de la arquitectura de la aplicación y los servicios.

En esta etapa, Se puede obtener la información para saber si se definió una arquitectura de alto nivel para la aplicación y los servicios y si incluyeron controles de seguridad en la misma.

Identificación de los campos de información sensible que maneja la aplicación.

En esta etapa se evalúan los tipos de datos que maneja la aplicación y se define cuál de ellos es sensible, con el fin de saber que controles de seguridad debo manejar para su tratamiento.

Riesgo	Recomendación
El no definir el carácter de los datos que maneja el app, podría desproteger de controles de seguridad, lo que podría traer problemas de confidencialidad futuros	Evaluar los datos ciñéndose a las definiciones legales

Información general

En esta etapa, se recopila información general sobre la aplicación. Algunos parámetros de interés se describen a continuación:

Nombre del paquete a analizar.

GID asociado al paquete en el dispositivo: Identificadores de los grupos asociados a la aplicación.

ID de usuario compartidas: Identificadores de usuario de Linux con los que se ejecuta la aplicación.

Etiquetas de usuario compartidas: Cadenas asociadas a los identificadores de usuario de Linux con los que se ejecuta la aplicación.

- Nombre de la versión de la aplicación:
- Número de versión de la aplicación:
- Fecha de instalación de la aplicación:
- Fecha de actualización más reciente:
- Ruta de la aplicación en el dispositivo:
- Bibliotecas compartidas:
- Ruta de APK en el dispositivo:
- Aplicación Target SDK:
- ¿Está habilitado el modo de depuración?
- Permisos requeridos:
- Intención de lanzamiento de la aplicación: Intención necesaria para iniciar la aplicación.

Riesgo	Recomendación
Un atacante extraerá información para adquirir conocimiento sobre la aplicación, mientras que la información global es necesaria para el funcionamiento de la aplicación, por lo que su exposición en sí no constituye un riesgo para la seguridad. El valor de cualquiera de los parámetros puede convertirse en un riesgo menor; por ejemplo, si el modo de depuración está habilitado.	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el acceso a dicha información. Algunos parámetros podrían comprometer la seguridad, por ejemplo, si la aplicación tiene el modo de depuración habilitado, lo que se recomienda evitar.

Lista de componentes

En esta etapa, se recopila información sobre los componentes de una aplicación. Estos son los componentes básicos de una aplicación de Android: Ocupaciones, Servicios, Proveedores de contenido, Receptores de radiodifusión.

Riesgo	Recomendación
Un atacante obtendrá la información para adquirir conocimiento sobre la aplicación, mientras que la lista de componentes de una aplicación es necesaria para ejecutar dicha aplicación, por lo	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el

que su exposición en sí, constituye un riesgo para la seguridad.	acceso a dicha información.

Necesidad de los componentes

Todos los componentes de la aplicación están definidos en términos de la lógica de negocio o las funciones de seguridad que proveen.

Riesgo	Recomendación
El atacante puede encontrar puertas traseras en los componentes que no son necesarios.	Se recomienda quitar todos los componentes que no requiera la app

Permisos de acceso a los componentes

Los permisos de acceso necesarios se pueden definir para cada componente de la aplicación de Android. Un atacante enumerará dichos permisos para acceder a componentes con funcionalidad sensible que no tienen las restricciones correspondientes.

Riesgo	Recomendación
Se puede acceder a un componente sin permisos definidos desde una aplicación sin restricciones, incluso se puede acceder desde cualquier otra aplicación del dispositivo si se exporta.	Se recomienda establecer permisos para todos los componentes que ejecutan funciones sensibles.

Validar si los componentes tienen habilitados la posibilidad de exportarse

Los componentes de la aplicación se pueden exportar si el indicador "exportado" en AndroidManifest está habilitado. Cuando se exporta un componente, se puede iniciar desde cualquier aplicación en el dispositivo.

Riesgo	Recomendación
Si la funcionalidad del componente es confidencial o necesita permisos que se hayan habilitado en la aplicación, un componente exportado incorrectamente podría permitir ejecutar dicha funcionalidad sensible desde otra aplicación hasta su lanzamiento.	Se recomienda exportar los componentes solo cuando sea estrictamente necesario. Al exportar un componente, se recomienda establecer un permiso, por lo que el lanzamiento necesitará un grado de seguridad adicional.

Validar si los componentes están en launchmode

Un atacante buscará la intención necesaria para lanzar el componente. Por lo tanto, si el componente es accesible (exportado en el caso de otras aplicaciones de dispositivos), podrá iniciarlo desde otra aplicación o componente.

Riesgo	Recomendación
Un atacante obtendrá la información para adquirir conocimiento sobre la aplicación, mientras que la lista de Intenciones de Lanzamiento de un componente de la aplicación es necesaria para ejecutar dicha aplicación, por lo que su exposición en sí no constituye un riesgo para la seguridad.	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el acceso a dicha información.

Diseño de modelo de amenazas

Realizar el levantamiento de un modelado de amenazas para la aplicación móvil y los servicios en el que se definieron las mismas y sus contramedidas.

Configuración y gestión de la implementación

Los errores en la configuración de la aplicación o los componentes comprometen la seguridad de la aplicación. En esta etapa, se definen varios errores en la configuración o en las opciones de implementación de la aplicación.

Validar si está habilitada la Depuración sin restricciones

Es habitual habilitar el modo de depuración mientras se desarrolla una aplicación para extraer información sobre su funcionamiento. Sin embargo, este modo debe estar deshabilitado al desarrollar aplicaciones.

Riesgo	Recomendación
La aplicación en el modo de depuración proporciona información que un atacante podría usar para realizar ataques en dicha aplicación.	Se recomienda establecer la opción de depuración en falso o eliminarla en el archivo AndroidManifest.xml, ya que una aplicación deshabilita el modo de depuración de forma predeterminada.

Validar el Uso de bibliotecas no actualizadas

El uso de bibliotecas desactualizadas podría causar vulnerabilidades que podrían poner en riesgo la aplicación y los datos procesados por dicha aplicación.

Riesgo	Recomendación

Es más probable que las bibliotecas desactualizadas contengan vulnerabilidades que pueden poner en riesgo la aplicación y los datos procesados por dicha aplicación.	Se recomienda usar las últimas versiones estables proporcionadas por los proveedores y / o desarrolladores de cada biblioteca utilizada en las aplicaciones.

Validar los Archivos predeterminados y de copia de seguridad

En esta etapa, el atacante busca versiones anteriores modificadas de archivos modificados, incluye archivos que se han cargado en el lenguaje de programación en uso y se pueden descargar como código fuente o incluso archivos de copia de seguridad automática o manual como archivos comprimidos. En el caso de Android, estos archivos pueden dejarse por error en la estructura de archivos APK.

Riesgo	Recomendación
Todos estos archivos podrían permitir que un atacante acceda a operaciones internas, puertas traseras, interfaces administrativas o incluso credenciales para conectarse a la interfaz administrativa o al servidor de bases de datos.	Antes de empaquetar una aplicación de Android, se recomienda asegurarse de que los archivos que pertenecen a la aplicación sean estrictamente necesarios y no se incluya ningún archivo que contenga información innecesaria o confidencial.

Metadatos sobre los archivos

En esta etapa, un atacante examinará los metadatos de los archivos incluidos en la aplicación para buscar información útil durante el ataque de penetración.

Riesgo	Recomendación
Dependiendo de la información almacenada en los metadatos, el riesgo puede variar mucho. Los metadatos típicos que se pueden encontrar en los archivos de aplicaciones de Android son los relacionados con las imágenes contenidas en las aplicaciones.	Se recomienda eliminar los metadatos de los archivos incluidos en la aplicación para no proporcionar información innecesaria. Existen herramientas de código abierto que permiten eliminar metadatos en muchos formatos diferentes; por ejemplo, Exiftool.

Insuficiente endurecimiento de WebView

WebView es una clase de Android que permite mostrar contenido en línea dentro de las Actividades de Android. Sin embargo, al instalar WebView en aplicaciones

Android, se debe tener en cuenta el hecho de que una configuración deficiente de esta extensión podría exponer al usuario de la aplicación a una multitud de riesgos.

Riesgo	Recomendación
Un endurecimiento insuficiente puede implicar riesgos para el usuario, convirtiéndose en posibles objetivos para ataques a través de la web.	Deshabilitar el soporte de los complementos del navegador; por ejemplo, con la siguiente declaración: webview.getSettings (). setPluginsEnabled (false); Deshabilitar el acceso a archivos locales; por ejemplo, con la siguiente declaración: webview.getSettings (). setAllowFileAccess (false); Evite cargar contenidos de sitios de terceros, verificando que WebView solo pueda acceder a aquellos sitios web que la aplicación necesita, usando listas blancas para tal efecto cuando sea posible. Deshabilitando JavaScript, con la siguiente declaración: webview.getWebSettings (). SetJavaScriptEnabled (false); Deshabilita el acceso a la URL de contenido dentro de WebView. El acceso a la URL de contenido permite que WebView cargue contenido de un proveedor de contenido instalado en el sistema, con la siguiente declaración: webview.getWebSettings (). SetsetAllowContentAccess (false); Establece si se debe permitir que JavaScript que se ejecuta en el contexto de una URL de esquema de archivos acceda al contenido de otras URL de esquemas de archivos, con la siguiente declaración: webview.getWebSettings (). SetAllowFileAccessFromFileURLs (false); Establece si se debe permitir que JavaScript que se ejecuta en el contexto de una URL de esquema de archivos acceda al contenido desde cualquier origen, con la siguiente declaración: webview.getWebSettings (). SetAllowUniversalAccessFromFileURLs (false);

Permisos de archivo mproper

La generación de archivos con el permiso “MODE_WORLD_READABLE” permite una lectura global de archivos, por lo que no se recomienda, excepto si se trata de

un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer archivos con el permiso "MODE_WORLD_WRITABLE".

Riesgo	Recomendación
La configuración de permisos de lectura global revela la información contenida en un archivo. Si el permiso de escritura está habilitado, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	Se recomienda generar archivos con permisos de lectura o escritura globales solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado. Un ejemplo de declaración de archivo peligroso puede verse a continuación: file = openFileOutput ("File_Name", Context.MODE_WORLD_READABLE)

Permisos inapropiados del proveedor de contenido

En Android, los Proveedores de contacto son una forma de compartir información entre aplicaciones a través de una API estructurada que permite mantener la integridad de los datos. También es posible establecer permisos de acceso y escritura para los proveedores de contenido. Si estos permisos no están configurados, cualquier aplicación podría tener acceso a los datos almacenados.

Riesgo	Recomendación

Una aplicación maliciosa podría acceder y / o modificar los datos almacenados en un proveedor de contenido sin que el usuario lo sepa, afectando la integridad y confidencialidad de dichos datos.	Se recomienda establecer permisos para los proveedores de contenido. Para tal efecto, se debe utilizar la guía de "permisos de uso"; por ejemplo: <uses-permission android: name = "android.permission.READ_USER_DICTIONARY">

Permisos de actividades mproper

En esta etapa, el atacante listará los permisos necesarios para iniciar cada actividad. Con esta información, es posible extraer la superficie de ataque tanto desde dentro como desde fuera del entorno de la aplicación, para que el atacante pueda aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, una actividad que realiza funciones que requieren permisos especiales y que se puede iniciar fuera de la aplicación (exportar) y no requiere permisos, compromete la seguridad debido a que permite que la actividad sea iniciada por cualquier aplicación de dispositivo.

Riesgo	Recomendación
Los riesgos varían según la sensibilidad de la funcionalidad de cada actividad.	Como regla general, se recomienda solicitar permisos sobre las actividades que realizan funciones confidenciales, especialmente si se exportan.

Permisos de servicios inadecuados

En esta etapa, el atacante listará los permisos necesarios para iniciar cada servicio. Con esta información, es posible obtener la superficie de ataque tanto dentro como fuera del entorno de la aplicación, por lo que el atacante podría aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, un servicio que realiza funciones que requieren permisos especiales y que se pueden iniciar fuera de la aplicación (exportar) y no requiere permisos, compromete la seguridad debido a que permite que cualquier aplicación de dispositivo inicie el servicio.

Riesgo	Recomendación
Los riesgos varían según la sensibilidad de la	Como regla general, se recomienda solicitar permisos en los servicios que realizan funciones confidenciales, especialmente si se exportan

funcionalidad de cada servicio.	

Permisos inadecuados de los receptores de difusión

En esta etapa, el atacante listará los permisos necesarios para iniciar cada receptor de difusión. Con esta información, es posible obtener la superficie de ataque tanto dentro como fuera del entorno de la aplicación, por lo que el atacante podría aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, un receptor de difusión que realiza funciones que requieren permisos especiales y que se pueden iniciar fuera de la aplicación (exportar) y no requiere permisos, compromete la seguridad debido a que permite que cualquier aplicación de dispositivo inicie el receptor de difusión.

Riesgo	Recomendación
Os riesgos varían según la sensibilidad de la funcionalidad de cada receptor de difusión.	Como regla general, se recomienda que se exijan permisos a los receptores de difusión que realizan funciones sensibles, especialmente si se exportan receptores de difusión.

Permisos de base de datos inadecuados

La generación de bases de datos con el permiso "MODE_WORLD_READABLE" habilitado permite la lectura global de la base de datos, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer bases de datos con el permiso "MODE_WORLD_WRITABLE". Un ejemplo de declaración de archivo peligroso puede verse a continuación: SQLiteDatabasemyWorldReadDB = openOrCreateDatabase ("Contacts", MODE_WORLD_READABLE, null);

Riesgo	Recomendación
La configuración de permisos de lectura global revela la información contenida en	Se recomienda generar archivos con permisos de lectura o escritura globales solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores.

un archivo. Si se proporciona el permiso de escritura, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Permisos de preferencias compartidas impropias

La generación de Preferencias Compartidas con el permiso “MODE_WORLD_READABLE” permite una lectura global de Preferencias Compartidas, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer Preferencias Compartidas con el permiso "MODE_WORLD_WRITABLE". Un ejemplo de declaración de archivo peligroso puede verse a continuación: SharedPreferencesprefsAllWrite = getSharedPreferences ("MisPreferenciasWrite", MODE_WORLD_WRITEABLE);

Riesgo	Recomendación
La configuración de permisos de lectura global revela la información contenida en un archivo. Si se proporciona el permiso de escritura, cualquier	Se recomienda generar archivos con permisos de lectura o escritura globales solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las

aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Criptografía

En esta sección, se prueban las funcionalidades relacionadas con el uso de criptografías en la aplicación. Esto puede ocurrir al enviar o almacenar datos.

Credenciales codificadas

El uso de contraseñas codificadas o vacías es fácil de extraer para un usuario malintencionado al desensamblar la aplicación.

Riesgo	Recomendación
Un usuario malintencionado podría tener acceso a credenciales codificadas, accediendo a los servicios permitidos por dichas credenciales.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información

Almacenamiento de datos inseguros

El cifrado no es suficiente para garantizar la confidencialidad de la información. Es necesario usar las funciones de encriptación correctamente, usando algoritmos más robustos y la mayor longitud de clave posible. Por ejemplo, es posible usar AES como algoritmo de cifrado en Android, pero se recomienda usar una clave de 256 bits en lugar de 128 bits.

Riesgo	Recomendación
Un usuario malintencionado podría realizar ataques en la criptografía en uso. Tenga en cuenta que un usuario con privilegios de administrador puede consultar la memoria de los procesos del dispositivo, pudiendo tener acceso a toda la información no encriptada.	Se recomienda usar AES con claves de 256 bits. Además, se recomienda generar vectores IV a través de SecureRandom y la clave debe derivar de una contraseña que utiliza protocolos conocidos como PBKDF2 (función de derivación de clave basada en contraseña).

Uso inseguro del protocolo de transporte

Las comunicaciones de información a través de los canales HTTP pueden ser fácilmente interceptadas. En el caso de usar HTTPS, un atacante podría usar técnicas de SSLStrip para acceder a dicha información, haciendo necesario el uso de mecanismos adicionales como forzar, desde el lado del cliente, para permitir el envío de información solo a través de un canal seguro de HTTPS.

Riesgo	Recomendación
Un usuario malintencionado podría realizar ataques en la criptografía en uso. En el caso de usar HTTPS, existen varios tipos de ataques; El más conocido es SSLStrip. Si el atacante tiene éxito, se podría acceder a la información que debía enviarse cifrada.	Se recomienda utilizar HTTPS en lugar de HTTP cuando sea posible. Además, se recomienda utilizar mecanismos del lado del cliente que obligan al envío de información a través de un canal seguro; por ejemplo, redirigir a un canal seguro si se detecta un intento de enviar información a través de un canal no cifrado. Además, se recomienda evaluar la necesidad de usar encabezados de seguridad de transporte estricto de HTTP en los servidores web; con dichos encabezados, el navegador forzaría el uso de canales encriptados en las comunicaciones

Fijación de certificados

Si la aplicación intercambia información a través de SSL y no confía en las autoridades de certificación, no hay razón para usar esta confiabilidad porque no brindan seguridad a las comunicaciones de la aplicación. Además, si alguna de estas autoridades de certificación está comprometida, los usuarios de la aplicación no serían vulnerables.

Riesgo	Recomendación
Si una autoridad de certificación está comprometida, el envío de certificados fraudulentos podría tener un impacto en la confidencialidad de la información transmitida, debido al hecho de que la aplicación confiaría en estos certificados porque los proporciona una autoridad de certificación conocida.	En caso de usar solo SSL para servicios definidos, se recomienda incluir técnicas de identificación de certificados. El navegador Google Chrome ya los usa para los servicios de Google y también para la aplicación oficial de Twitter.

Política de cifrado

Existe una política explicita para el manejo de las claves criptográficas (si se usan) y se refuerza su ciclo de vida. Idealmente siguiendo un estándar del manejo de claves como el NIST SP 800-57.

Fuga de información

En la sección de autenticación, se verifican las fugas de información en los medios. La información sensible podría estar relacionada con el usuario o con el propio teléfono.

Fuga de información a los archivos de registro

Es posible almacenar información en los registros de Android a través de las siguientes funciones:

Log.v (). Para almacenar información ampliada.
Log.d (). Con el fin de almacenar información de depuración.
Log.i (). Para almacenar información básica.
Log.w (). Con el fin de almacenar información sobre las alertas.
Log.e (). Con el fin de almacenar los registros de errores.

No se recomienda almacenar información confidencial en los registros del dispositivo porque cualquier aplicación con el permiso "READ_LOGS" podría tener acceso a dichos registros.

Riesgo	Recomendación
Cualquier aplicación con acceso a los registros podría extraer información confidencial que podría almacenarse en esos registros.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, estas pautas deben seguirse para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las

	tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Fuga de información a la tarjeta SD

Es posible almacenar datos en tarjetas SD a través de las siguientes funciones: Con API 7 o inferior. A través de la función getExternalStorageDirectory (). Con API8 o superior. A través de la función getExternalFilesDir (). No se recomienda almacenar datos confidenciales en la tarjeta SD del dispositivo, ya que cualquier aplicación podría extraerlos.

Riesgo	Recomendación
Cualquier aplicación podría tener acceso a los datos confidenciales almacenados en la tarjeta SD.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, se deben seguir las siguientes pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Fuga de información a la red

Para enviar datos a la red, es necesario tener el permiso "android.permission.INTERNET" habilitado. Una aplicación con dicho permiso puede enviar información a la red a través de muchas bibliotecas; los más comunes son java.net. * y android.net. *. No se recomienda enviar información confidencial sin cifrar, ya que un atacante podría extraerla detectando el tráfico de la red.

Riesgo	Recomendación
Un atacante podría obtener acceso a información confidencial sin cifrar enviada a la red mediante el rastreo.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, estas pautas deben seguirse para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Fuga de información al IPC

Es posible almacenar información en Intentos a través de las funciones theputExtra () y putExtras () de la clase Intent, así como a través del URI en los parámetros por el método GET a través del parse () de la clase URI. No se recomienda enviar datos confidenciales sin cifrar a través de Intenciones Implícitas, ya que la aplicación que procesará dicha información es desconocida a priori.

Riesgo	Recomendación
Una aplicación	SSe recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia,

ilegítima podría obtener acceso a datos confidenciales al interceptar la Intención.	cifrar dicha información. En términos generales, estas pautas deben seguirse para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado

Validación de datos:

Evaluación de gestión de entrada de usuario En esta categoría, se incluirán las vulnerabilidades relacionadas con la entrada recibida del usuario que es administrada por la aplicación. La validación incorrecta de los datos del usuario es uno de los principales vectores de ataque, ya que podría permitir que un atacante altere el flujo de datos de la aplicación, inyectando código y afectando seriamente la aplicación y los datos almacenados en él. Esta categoría ocupa el 4º lugar en los 10 principales riesgos de la aplicación móvil titulado "Inyección del lado del cliente". A pesar de que el origen de este tipo de vulnerabilidad es el mismo, dependiendo de la ubicación de la interacción de los datos recibidos del usuario, pueden tener lugar diferentes tipos de vulnerabilidades con múltiples impactos.

Cross Site Scripting

Las vulnerabilidades de Cross Site Scripting (XSS) son el resultado de una codificación incorrecta de los datos recibidos de fuentes inseguras antes de usarlos en la salida HTML. En el caso de Android, la vulnerabilidad se deriva básicamente del uso de la función "setJavaScriptEnabled ()" de la clase WebView. Esta función

permite incluir objetos Java en el contexto de JavaScript de WebView, por lo que si los datos proporcionados a dicha función no están limpios, se podría ejecutar un código JavaScript arbitrario

Riesgo	Recomendación
Las vulnerabilidades de secuencias de comandos en sitios cruzados (XSS) permiten que un atacante ejecute un código de secuencia de comandos (HTML, JavaScript, VBScript) en el navegador del usuario (en el dominio de la aplicación vulnerable). Por ejemplo, un atacante podría explotar una vulnerabilidad de XSS para extraer cookies de usuario (para luego suplantar a la víctima en la aplicación vulnerable) o solicitar información a los usuarios (como las credenciales). Esta vulnerabilidad se produce como resultado del uso de información insegura en una salida HTML sin la codificación correcta y previa.	Como regla general, la información recibida del usuario no debe ser confiable. Es peligroso usar los datos recibidos directamente del usuario sin codificarlos correctamente en la generación de la página de respuesta. De forma predeterminada, WebView no ejecuta JavaScript, por lo que debe estar habilitado solo cuando sea estrictamente necesario, controlando el origen de la carga dinámica del código a ejecutar. Sin embargo, para deshabilitar expresamente la ejecución de JavaScript, se recomienda ingresar la siguiente declaración: webview.getSettings (). SetJavaScriptEnabled (false); En el caso de utilizar la función "addJavascriptInterface ()", se recomienda filtrar los objetos Java proporcionados a WebView a través de listas blancas siempre que sea posible.

Desbordamiento de búfer

Las vulnerabilidades de desbordamiento de búfer son el resultado de no verificar correctamente el tamaño de ciertos datos antes de copiarlos en un área de memoria en particular. En el caso de Android, este tipo de error es fácil de cometer mientras se usa el código nativo si no se valida el tamaño del búfer de datos. Aunque Android incorpora protecciones como ASLR y DEP que reducen la explotabilidad, no resuelven el problema fundamental.

Riesgo	Recomendación
Los riesgos asociados con este tipo de error se consideran muy serios para copiar información en un área de memoria que estaba reservada para otras tareas y almacenar otros datos del programa. Cuando se produce un desbordamiento, los resultados son impredecibles y provocan el final de la ejecución	Como regla general, cualquier operación que implique copiar datos en un área de memoria debe realizarse con cuidado para garantizar que el área de memoria de destino tenga

del programa como una regla general, y también es posible provocar una denegación de servicio. Al explotar tales errores, un atacante podría controlar la ejecución del programa, pudiendo acceder al sistema de forma remota por un atacante experimentado.	suficiente memoria para contener los datos a copiar. Existe un conjunto de funciones que es más probable que causen errores y no se recomiendan.

Inyección SQL

Las vulnerabilidades de inyección de SQL pueden ocurrir cuando la información proveniente de contenedores inseguros se usa para crear consultas dinámicas de SQL sin implementar controles y / o filtros relevantes para dicha información. Los contenedores inseguros son cualquier fuente de datos del código fuente, como los datos proporcionados por el usuario, las bases de datos no controladas, los servicios web, etc. Android utiliza SQLite para almacenar datos. A continuación se muestra un ejemplo de código vulnerable:
String con = "DELETE FROM case_values WHERE _id =" + paramString; localSQLiteDatabase.execSQL (con);:

Riesgo	Recomendación
Las vulnerabilidades de inyección de SQL podrían permitir que un atacante use caracteres especiales en el lenguaje SQL para alterar una declaración que se construye con información externa de una manera dinámica. Como resultado, podría ser posible obtener información arbitraria de la base de datos, alterar esta información y alterar el flujo de trabajo de la aplicación afectada.	Se recomienda validar cada campo a través de listas blancas. Por lo tanto, solo se aceptarán caracteres no peligrosos. Es importante normalizar la entrada para evitar evasiones mediante codificaciones de parámetros. Si es posible, se recomienda usar sentencias preparadas, ya que es la opción más segura actual para construir sentencias SQL con parámetros externos.

Manipulación del camino

Si la generación de archivos depende de la entrada del usuario, es necesario filtrar estos datos porque, de lo contrario, un usuario malintencionado podría tener acceso a información arbitraria utilizando los permisos de la aplicación vulnerable. A continuación se muestra un ejemplo de generación de archivos peligrosos:

file = openFileOutput (unfiltered_variable, Context.MODE_WORLD_READABLE); Si "unffiltered_variable" proviene del usuario sin filtrar, el acceso a los archivos por parte de la aplicación podría controlarse, permitiendo que un usuario malintencionado sobrescriba los archivos de la aplicación.

Riesgo	Recomendación
Esta vulnerabilidad pone en riesgo la integridad y la disponibilidad de los archivos administrados por la aplicación.	Se recomienda no utilizar datos provenientes del usuario para crear nombres de archivos. Si es necesario, se recomienda filtrar los datos de entrada, preferiblemente de acuerdo con el uso de listas blancas

Dos Null Check en IPC.

Android finaliza las aplicaciones cuando se encuentra una referencia nula. Si no se verifican las referencias nulas al recuperar datos de un Intent, un atacante podría enviar intentos con campos con valores nulos para denegar el servicio a las aplicaciones, especialmente a aquellas que tienen servicios en ejecución. En particular, se debe verificar la existencia de referencias nulas al recuperar datos de un Intent, especialmente los siguientes métodos: getAction () y getExtra ().:

Riesgo	Recomendación
Esta vulnerabilidad pone en riesgo la disponibilidad de las aplicaciones que no realizan comprobaciones para no intentar acceder a valores nulos. En caso de que los servicios ejecuten los servicios, un atacante podría denegar dichos servicios sin que el usuario lo sepa, por lo que es especialmente dañino.	Se recomienda verificar los valores nulos al recuperar datos de Intents, especialmente los siguientes métodos: getAction () y getExtra ().

Inyección de troncos

Un atacante podría incluir ciertos caracteres especiales en una cadena de texto especialmente manipulada, por lo que cuando se usa en la creación de una nueva entrada en los archivos de registro, el atacante resultará en una manipulación libre de la salida. En el caso de Android, las funciones estándar de registro de información son las siguientes:

Log.v (). Para almacenar información ampliada.
Log.d (). Con el fin de almacenar información de depuración.
Log.i (). Para almacenar información básica.

Log.w (). Con el fin de almacenar información sobre las alertas.
Log.e (). Con el fin de almacenar los registros de errores.

Riesgo	Recomendación
Los archivos de registro podrían estar dañados debido a una entrada falsa o no válida. Esto podría dar como resultado registros no confiables y la necesidad de descartar todos los registros sobre las acciones realizadas en el sistema por los usuarios. Los atacantes podrían usar esta situación para ocultar acciones criminales y evitar que se detecten sus acciones y, por lo tanto, evitar su persecución por los delitos cometidos.	Se deben tener en cuenta todos los metacaracteres utilizados para la aplicación al generar registros. Por ejemplo, si la aplicación utiliza el carácter de canalización "|" (sin las comillas) para formatear la entrada, este será un meta-carácter. Es decir, un atacante podría incluir un extra | y sería imposible reconstruir después si eso | Fue ingresado por la aplicación o por el atacante. Esto se aplica para cualquier carácter especial en los registros.

Intent inyectable

Si la entrada del usuario se carga de forma dinámica en los datos de Intención, un usuario malintencionado podría manipular dichos datos para ejecutar el código a través de ellos. En particular, se debe verificar la existencia de datos dinámicos al incluir dichos datos en un Intent, especialmente a través de los siguientes métodos de Intent: addCategory (), setAction (), setClass (), setClassName (), setComponent (), setData () andsetDataAndType ().:

Riesgo	Recomendación
esta vulnerabilidad pone en riesgo la ejecución normal de la aplicación, ya que si un usuario puede modificar los datos almacenados en los Intentos, el rendimiento normal de la aplicación también podría modificarse, así como la interacción entre las aplicaciones mediante el envío de Intenciones	Se recomienda verificar los valores de entrada del usuario, especialmente aquellos que van a Intent data instantiating la aplicación. Para tal fin, se recomienda inspeccionar los datos de usuario que terminan como parámetros en cualquiera de las siguientes funciones de Intención: addCategory (), setAction (), setClass (), setClassName (), setComponent (), setData () y setDataAndType (

Intención Spoofing

Esta categoría cubre todas las vulnerabilidades involucradas en la entrega de intenciones arbitrarias a un componente que espera recibir otro tipo de intención. De esta manera, el atacante utilizará los filtros de la víctima para enviar datos

inesperados y, por lo tanto, aprovechar estas funcionalidades. Para tener éxito con este tipo de vulnerabilidad, los componentes de destino deben ser públicos o exportables como regla general; De lo contrario, no estarán disponibles para otras aplicaciones. Para tal fin, los componentes deben tener el indicador "EXPORTADO" habilitado en el archivo AndroidManifest.xml. Tenga en cuenta que los receptores de difusión pueden declararse en AndroidManifest o en tiempo de ejecución.

Inyección de difusión

Los receptores de difusión se utilizan para registrar intenciones y responderlas. Si uno de estos componentes es público, cualquier aplicación podría enviar intenciones. El registro de Intenciones con acciones del sistema por parte de Broadcast Receivers es peligroso, ya que el hecho de registrar tales Intenciones hace que el elemento sea público y accesible, por lo que una aplicación maliciosa podría enviar Intenciones explícitas sin las acciones correspondientes porque son acciones del sistema.

Riesgo	Recomendación
En muchos casos, los receptores de difusión obtienen Intenciones y activan Actividades o Servicios con sus datos. Las vulnerabilidades de la inyección de difusión podrían permitir a un atacante manipular estos componentes para interactuar con ellos mediante la inyección de datos arbitrarios.	Se recomienda usar componentes públicos solo cuando sea necesario y, en tales casos, validar a fondo cada parámetro recuperado de la Intención. Si Broadcast Receivers registra acciones del sistema, se recomienda validar la acción del intento, ya que un intento malicioso no puede registrar las acciones del sistema, por lo que esta validación se puede utilizar para saber si el intento proviene del sistema Android o no.

Lanzamiento de actividad maliciosa

Actividades pueden iniciarse sobre la base de datos provenientes del usuario o la recepción de intenciones arbitrarias. Esto permite varias superficies de ataque: Si una Actividad utiliza datos del Intención sin verificar su origen, la integridad de los datos de la aplicación se verá afectada.

Si una Actividad entrega datos considerados como confidenciales y pueden ser lanzados públicamente, un Intento especialmente manipulado podría lanzarlos, lo que lleva a una divulgación de información confidencial.ndaciones:

Riesgo	Recomendación
La presencia de esta vulnerabilidad podría confundir a un usuario, ya que los usuarios pensarían que están usando una Actividad legítima, cuando en realidad están realizando acciones que se informan al atacante. Adicionalmente, es posible la filtración de la información utilizada por la Actividad.	Se recomienda usar componentes públicos solo cuando sea necesario y, en esos casos, validar a fondo cada parámetro recuperado de la Intención o del usuario.

Lanzamiento de servicio malicioso.

Se recomienda usar componentes públicos solo cuando sea necesario y, en esos casos, validar a fondo cada parámetro recuperado de la Intención o del usuario. Además, los servicios pueden ser protegidos estableciendo permisos.

Riesgo	Recomendación
La presencia de esta vulnerabilidad podría permitir realizar acciones arbitrarias en la aplicación mediante la ejecución de Servicios y también dar lugar a la divulgación de información utilizada por el Servicio. Como circunstancia agravante, los Servicios generalmente confían más en los datos proporcionados por los usuarios y las intenciones; además, en muchos casos proporcionan interfaces API para que las aplicaciones puedan comunicarse entre sí.	Se recomienda usar componentes públicos solo cuando sea necesario y, en esos casos, validar a fondo cada parámetro recuperado de la Intención o del usuario. Además, los servicios pueden ser protegidos estableciendo permisos.

Inseguro Control de Intención Pendiente

Una aplicación puede delegar tareas a otra aplicación a través de intenciones pendientes. La aplicación que recibe la Intención pendiente no puede modificar sus datos, pero completa los datos vacíos. Si una aplicación genera una Intención pendiente con tal propósito pero el objetivo no está definido, una aplicación que reciba la Intención pendiente podría agregarla, por lo que dichos datos se enviarían a un componente arbitrario.

Riesgo	Recomendación
Esta vulnerabilidad podría permitir la filtración de la información que está	Cuando las tareas se delegan a otra aplicación a través de una Intención

siendo procesada por la aplicación que generó la Intención pendiente. Además, una aplicación maliciosa podría recibir, procesar y enviar dicha información, realizando un ataque Man-in-the-Middle.	pendiente, se recomienda establecer siempre el destino en el que se delegan las tareas para que sean ejecutadas por una aplicación de terceros.

Intención no autorizada

Esta categoría cubre todas las vulnerabilidades relacionadas con la resolución de la entrega de Intención implícita. Cuando una aplicación envía un Intent implícito, no hay garantía de que una aplicación malintencionada no recopile dicho Intent, ya que una aplicación maliciosa podría registrar un Intent Filter capaz de pasar la resolución (acción, datos y categoría), a menos que dicho Intent tenga un conjunto de permisos requeridos que el usuario malicioso no tiene. Si una aplicación malintencionada puede interceptar Intenciones implícitas, podría tener acceso a la ejecución del flujo de datos, pudiendo realizar ataques de denegación de servicio o phishing. Esta categoría considera cómo este tipo de vulnerabilidad puede ser expuesta en componentes particulares: Broadcast, Activities y Services.

Robo de transmisiones

Una intención implícita es pública cuando no está protegida con los permisos de tipo "Signature" o "SignatureOrSystem". Cuando una aplicación envía un intento público implícito, dicho componente puede leer dicho intento. Una aplicación malintencionada podría tener acceso a él, obteniendo datos confidenciales si el intento simplemente contiene filtros de intento para todas las acciones, datos y categorías. Esto es especialmente peligroso en Sticky Broadcasts, ya que el Intento persiste en ellos y puede redirigirse repetidamente a varios destinatarios. Esto le da una mayor ventana de tiempo en la que el componente malicioso puede operar. Además, Sticky Broadcasts no puede protegerse mediante permisos.

Riesgo	Recomendación
Un atacante podría realizar un ataque de denegación de servicio en las transmisiones ordenadas, ya que un Intent solo se puede propagar sobre	Se recomienda considerar la necesidad de enviar Intentos implícitos. Para comunicar intenciones en una aplicación, siempre se

ellos si el primer componente que recibe el Intent para usarlo como salida. Además, podría usarse para realizar ataques Man-in-the-Middle con su posterior inyección de datos en los Intentos de propagación.	recomienda el uso de intenciones explícitas. Cuando es necesario enviar datos confidenciales en un Intento, se recomienda protegerlos configurando permisos y cifrando dichos datos.

Secuestro de actividad

Al aprovechar esta vulnerabilidad, se lanza una Actividad maliciosa en lugar de la esperada, por lo que el usuario estará en una aplicación incorrecta sin ser consciente. Esto sucede cuando el cargo de una Actividad depende de una Intención implícita. El atacante registra un Intent Filter más preciso y lo controla. Esto no siempre es posible, ya que si varias Actividades son capaces de procesar una Intención implícita, se mostrará un mensaje al usuario para que elija la aplicación. Sin embargo, esto es muy peligroso porque si esto sucede y el usuario configura como acción predeterminada la carga de la aplicación maliciosa, la Actividad maliciosa siempre se abrirá sin que se le solicite.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite ejecutar ataques de phishing, así como también fugas de la información que maneja el usuario en la Actividad involucrada. Además, esta vulnerabilidad permite al atacante modificar los datos, poniendo en riesgo su integridad.	Se recomienda considerar la necesidad de enviar Intentos implícitos. Para comunicar intenciones en una aplicación, siempre se recomienda el uso de intenciones explícitas. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos configurando permisos y cifrando dichos datos.

Servicio de secuestro

Al aprovechar esta vulnerabilidad, se lanza una Actividad maliciosa en lugar de la esperada, por lo que el usuario estará en una aplicación incorrecta sin ser consciente. Esto sucede cuando el cargo de una Actividad depende de una Intención implícita. El atacante registra un Intent Filter más preciso y lo controla. Esta vulnerabilidad es más persistente debido a que es transparente para el usuario porque los servicios no incluyen una interfaz gráfica para ella.

Riesgo	Recomendación

La presencia de esta vulnerabilidad permite la filtración de la información procesada por el Servicio involucrado. Además, esta vulnerabilidad permite que un atacante modifique los datos, poniendo en riesgo su integridad	Se recomienda considerar la necesidad de enviar Intentos implícitos. Para comunicar intenciones en una aplicación, siempre se recomienda el uso de intenciones explícitas. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos configurando permisos y cifrando dichos datos.

Debilidad intenciones especiales.

Si se envía un intento implícito para tratar con un proveedor de contenido, puede contener información confidencial en el URI, pudiendo leerse si un componente malicioso intercepta el intento. Por otro lado, un intento pendiente conserva los permisos de su creador, por lo que si un componente malintencionado intercepta un intento pendiente, podría usar los permisos de la creación del intento pendiente para suplantarlo.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite la filtración de información incluida en el URI. Además, esta vulnerabilidad permite al atacante suplantar los permisos del creador de la intención pendiente.	Se recomienda evaluar la necesidad de enviar intenciones implícitas. Para comunicar intenciones en una aplicación, se recomienda utilizar intenciones explícitas. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos estableciendo permisos y cifrando dichos datos.

Lógica de negocio.

En esta categoría, se incluyen las vulnerabilidades con componentes más centrados en el diseño en lugar de la codificación. Se incluyen tanto la casuística de rendimiento como la capacidad de la aplicación para trabajar de una manera inesperada que afecta su flujo de trabajo.

Los ataques dirigidos a lógicas de negocios de aplicaciones son peligrosos, difíciles de detectar y especialmente dirigidos a la aplicación.

4. METODOLOGÍA

El tipo de investigación que se aplicó a lo largo del trabajo *Metodología de testing de seguridad para aplicaciones móviles Android, en el campo de la salud*, correspondió al tipo descriptivo, ya que a través de él se describen las categorías de seguridad que debe cumplir una aplicación mHealth para garantizar el manejo seguro de la información de los pacientes. El trabajo de investigación asimismo tiene un carácter documental, por tanto, se hizo una revisión documental que implicó "hacer una revisión previa de los estudios anteriores y de literatura relacionada que permitió establecer que se ha dicho sobre el tema propuesto, desde que punto de vista y con qué resultados" (Galeano, 2004, pág. 116). A su vez tiene un carácter propositivo, pues formuló una metodología de testing de seguridad para aplicaciones mHealth en el Sistema Operativo Android.

Para el cumplimiento cabal de los objetivos del trabajo de investigación se organizó por fases. En las tres primeras fases del trabajo de investigación, las cuales dieron cumplimiento a los tres primeros objetivos específicos, la principal técnica de investigación fue la revisión documental, la cual tuvo como objetivo realizar una revisión de la literatura asociada con el tema, y tenía como propósito ubicar, clasificar y analizar la información detectada. El compendio de documentos con los cuales se realizó esta revisión documental, se conformó a partir de búsquedas sistemáticas en bibliotecas, además de rastreos en los principales motores de búsqueda y bases de datos disponibles en red, tales como SciELO, Dialnet, World Wide Science y Google Scholar. Los documentos encontrados fueron categorizados según su relevancia para los propósitos de este trabajo, y se usaron fichas de lectura para consignar la información más significativa, en todo el trabajo de rastreo y ubicación de la información se priorizó la búsqueda de fuentes autorizadas, tales como libros, artículos de revista indexadas, investigaciones, tesis, y en general, documentación validada que aportaron elementos para esta investigación.

La primera fase del trabajo de investigación realizó una revisión bibliográfica tendiente a identificar las características relevantes que deben cumplir las aplicaciones móviles mHealth, dicha revisión bibliográfica hace parte del capítulo dos. En la segunda fase se realizó una revisión bibliográfica que identificó los criterios de seguridad de las aplicaciones móviles con énfasis en las aplicaciones mHealth, esta revisión hace parte del capítulo tercero. Y la tercera fase permitió describir las diferentes metodologías dedicadas al testing de seguridad de aplicaciones móviles, los cuales también son componentes del capítulo tercero. Para dar cumplimiento a la cuarta fase, y por consiguiente al cuarto objetivo específico, se definió una metodología de seguridad contando con la información recabada en las fases anteriores el cual también hace parte del capítulo tercero. Y la quinta fase se realizó a partir de la definición de la metodología de testing propuesta con la cual construyó el test de seguridad para aplicaciones mHealth propuesto.

4.1. Propuesta de metodología de seguridad para aplicaciones mHealth para el Sistema Operativo Android

Para definir la metodología que es el propósito de este trabajo investigación se partió de la *Metodología Abierta de Evaluación de Seguridad de Android* (OASAM, por sus siglas en inglés), esta determinación se tomó basados en dos criterios, el primero, que es una de las pocas metodologías diseñadas exclusivamente para las aplicaciones Android, y el segundo, porque su objetivo manifiesto es convertirse en un marco de referencia en las evaluaciones de vulnerabilidad de aplicaciones de Android. Por lo cual resultaba muy apropiada para los intereses de este trabajo de investigación.

Posteriormente lo que se realizó fue incluir componentes de la metodología *Proyecto Abierto de Seguridad de Aplicaciones Web* (OWASP, por sus siglas en inglés), que es otra de las metodologías a las que tuvimos acceso y que tiene indicaciones exclusivas para la plataforma Android, en la metodología OASAM.

La manera como se construyó la metodología propuesta en este trabajo de investigación, fue la de realizar un análisis comparativo entre estas dos metodologías, e incluir componentes de la metodología OWASP, que fueran útiles para los requerimientos de seguridad de las aplicaciones mHealth en el compendio de pasos descritos en la metodología OASAM. Con lo cual, lo que se aportó a la metodología OASAM fue potenciar los pasos que, a criterio de los autores de este trabajo de investigación, eran necesarios para adaptarla a los requerimientos de las aplicaciones mHealth.

Información de la aplicación

Recopilación de la información.

La recopilación de información de la aplicación es una de las partes más importantes en las evaluaciones de aplicaciones de Android. En esta etapa, se define la superficie de ataque, lo que motiva la prueba de vulnerabilidad que serán definidos por los controles.

Obtener la Definición de la arquitectura de la aplicación y los servicios.

En esta etapa, Se puede obtener la información para saber si se definió una arquitectura de alto nivel para la aplicación y los servicios y si incluyeron controles de seguridad en la misma.

Identificación de los campos de información sensible que maneja la aplicación.

En esta etapa se evalúan los tipos de datos que maneja la aplicación y se define cuál de ellos es sensible, con el fin de saber que controles de seguridad debo manejar para su tratamiento.

Riesgo	Recomendación
El no definir el carácter de los datos que maneja el app, podría desproteger de controles de seguridad, lo que podría traer problemas de confidencialidad futuros	Evaluar los datos ciñéndose a las definiciones legales

Información general

En esta etapa, se recopila información general sobre la aplicación. Algunos parámetros de interés se describen a continuación:
Nombre del paquete a analizar.

GID asociado al paquete en el dispositivo: Identificadores de los grupos asociados a la aplicación.

ID de usuario compartidas: Identificadores de usuario de Linux con los que se ejecuta la aplicación.

Etiquetas de usuario compartidas: Cadenas asociadas a los identificadores de usuario de Linux con los que se ejecuta la aplicación.

- Nombre de la versión de la aplicación:
- Número de versión de la aplicación:
- Fecha de instalación de la aplicación:
- Fecha de actualización más reciente:
- Ruta de la aplicación en el dispositivo:
- Bibliotecas compartidas:
- Ruta de APK en el dispositivo:
- Aplicación Target SDK:
- ¿Está habilitado el modo de depuración?
- Permisos requeridos:
- Intención de lanzamiento de la aplicación: Intención necesaria para iniciar la aplicación.

Riesgo	Recomendación
Un atacante extraerá información para adquirir conocimiento sobre la aplicación, mientras que la información global es necesaria para el funcionamiento de la aplicación, por lo que su exposición en sí no constituye un riesgo para la seguridad. El valor de cualquiera de los parámetros puede convertirse en un riesgo menor; por ejemplo, si el modo de depuración está habilitado.	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el acceso a dicha información. Algunos parámetros podrían comprometer la seguridad, por ejemplo, si la aplicación tiene el modo de depuración habilitado, lo que se recomienda evitar.

Lista de componentes

En esta etapa, se recopila información sobre los componentes de una aplicación. Estos son los componentes básicos de una aplicación de Android: Ocupaciones, Servicios, Proveedores de contenido, Receptores de radiodifusión.

Riesgo	Recomendación
Un atacante obtendrá la información para adquirir conocimiento sobre la aplicación, mientras que la lista de componentes de una aplicación es necesaria para ejecutar dicha aplicación, por lo que su exposición en	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el acceso a dicha información.

sí, constituye un riesgo para la seguridad.	

Necesidad de los componentes

Todos los componentes de la aplicación están definidos en términos de la lógica de negocio o las funciones de seguridad que proveen.

Riesgo	Recomendación
El atacante puede encontrar puertas traseras en los componentes que no son necesarios.	Se recomienda quitar todos los componentes que no requiera la app

Permisos de acceso a los componentes

Los permisos de acceso necesarios se pueden definir para cada componente de la aplicación de Android. Un atacante enumerará dichos permisos para acceder a componentes con funcionalidad sensible que no tienen las restricciones correspondientes.

Riesgo	Recomendación
Se puede acceder a un componente sin permisos definidos desde una aplicación sin restricciones, incluso se puede acceder desde cualquier otra aplicación del dispositivo si se exporta.	Se recomienda establecer permisos para todos los componentes que ejecutan funciones sensibles.

Validar si los componentes tienen habilitados la posibilidad de exportarse

Los componentes de la aplicación se pueden exportar si el indicador "exportado" en AndroidManifest está habilitado. Cuando se exporta un componente, se puede iniciar desde cualquier aplicación en el dispositivo.

Riesgo	Recomendación
Si la funcionalidad del componente es confidencial o necesita permisos que se hayan habilitado en la aplicación, un componente exportado incorrectamente podría permitir ejecutar dicha funcionalidad sensible desde otra aplicación hasta su lanzamiento.	Se recomienda exportar los componentes solo cuando sea estrictamente necesario. Al exportar un componente, se recomienda establecer un permiso, por lo que el lanzamiento necesitará un grado de seguridad adicional.

Validar si los componentes están en launchmode

Un atacante buscará la intención necesaria para lanzar el componente. Por lo tanto, si el componente es accesible (exportado en el caso de otras aplicaciones de dispositivos), podrá iniciarlo desde otra aplicación o componente.

Riesgo	Recomendación
Un atacante obtendrá la información para adquirir conocimiento sobre la aplicación, mientras que la lista de Intenciones de Lanzamiento de un componente de la aplicación es necesaria para ejecutar dicha aplicación, por lo que su exposición en sí no constituye un riesgo para la seguridad.	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el acceso a dicha información.

Diseño de modelo de amenazas
Realizar el levantamiento de un modelado de amenazas para la aplicación móvil y los servicios en el que se definieron las mismas y sus contramedidas.

Configuración y gestión de la implementación
Los errores en la configuración de la aplicación o los componentes comprometen la seguridad de la aplicación. En esta etapa, se definen varios errores en la configuración o en las opciones de implementación de la aplicación.

Validar si está habilitada la Depuración sin restricciones
Es habitual habilitar el modo de depuración mientras se desarrolla una aplicación para extraer información sobre su funcionamiento. Sin embargo, este modo debe estar deshabilitado al desarrollar aplicaciones.

Riesgo	Recomendación
La aplicación en el modo de depuración proporciona información que un atacante podría usar para realizar ataques en dicha aplicación.	Se recomienda establecer la opción de depuración en falso o eliminarla en el archivo AndroidManifest.xml, ya que una aplicación deshabilita el modo de depuración de forma predeterminada.

Validar el Uso de bibliotecas no actualizadas
El uso de bibliotecas desactualizadas podría causar vulnerabilidades que podrían poner en riesgo la aplicación y los datos procesados por dicha aplicación.

Riesgo	Recomendación
Es más probable que las bibliotecas desactualizadas contengan vulnerabilidades que pueden poner en riesgo la aplicación y los datos procesados por dicha aplicación.	Se recomienda usar las últimas versiones estables proporcionadas por los proveedores y / o desarrolladores de cada biblioteca utilizada en las aplicaciones.

Validar los Archivos predeterminados y de copia de seguridad

En esta etapa, el atacante busca versiones anteriores modificadas de archivos modificados, incluye archivos que se han cargado en el lenguaje de programación en uso y se pueden descargar como código fuente o incluso archivos de copia de seguridad automática o manual como archivos comprimidos. En el caso de Android, estos archivos pueden dejarse por error en la estructura de archivos APK.

Riesgo	Recomendación
Todos estos archivos podrían permitir que un atacante acceda a operaciones internas, puertas traseras, interfaces administrativas o incluso credenciales para conectarse a la interfaz administrativa o al servidor de bases de datos.	Antes de empaquetar una aplicación de Android, se recomienda asegurarse de que los archivos que pertenecen a la aplicación sean estrictamente necesarios y no se incluya ningún archivo que contenga información innecesaria o confidencial.

Metadatos sobre los archivos

En esta etapa, un atacante examinará los metadatos de los archivos incluidos en la aplicación para buscar información útil durante el ataque de penetración.

Riesgo	Recomendación
Dependiendo de la información almacenada en los metadatos, el riesgo puede variar mucho. Los metadatos típicos que se pueden encontrar en los archivos de aplicaciones de Android son los relacionados con las imágenes contenidas en las aplicaciones.	Se recomienda eliminar los metadatos de los archivos incluidos en la aplicación para no proporcionar información innecesaria. Existen herramientas de código abierto que permiten eliminar metadatos en muchos formatos diferentes; por ejemplo, Exiftool.

Insuficiente endurecimiento de WebView

WebView es una clase de Android que permite mostrar contenido en línea dentro de las Actividades de Android. Sin embargo, al instalar WebView en aplicaciones Android, se debe tener en cuenta el hecho de que una configuración deficiente de esta extensión podría exponer al usuario de la aplicación a una multitud de riesgos.

Riesgo	Recomendación
Un endurecimiento insuficiente puede implicar riesgos para el usuario, convirtiéndose en posibles objetivos para ataques a través de la web.	Deshabilitar el soporte de los complementos del navegador; por ejemplo, con la siguiente declaración: webview.getSettings (). setPluginsEnabled (false); Deshabilitar el acceso a archivos locales; por ejemplo, con la siguiente declaración: webview.getSettings (). setAllowFileAccess (false);

	Evite cargar contenidos de sitios de terceros, verificando que WebView solo pueda acceder a aquellos sitios web que la aplicación necesita, usando listas blancas para tal efecto cuando sea posible. Deshabilitando JavaScript, con la siguiente declaración: webview.getWebSettings (). SetJavaScriptEnabled (false); Deshabilita el acceso a la URL de contenido dentro de WebView. El acceso a la URL de contenido permite que WebView cargue contenido de un proveedor de contenido instalado en el sistema, con la siguiente declaración: webview.getWebSettings (). SetsetAllowContentAccess (false); Establece si se debe permitir que JavaScript que se ejecuta en el contexto de una URL de esquema de archivos acceda al contenido de otras URL de esquemas de archivos, con la siguiente declaración: webview.getWebSettings (). SetAllowFileAccessFromFileURLs (false); Establece si se debe permitir que JavaScript que se ejecuta en el contexto de una URL de esquema de archivos acceda al contenido desde cualquier origen, con la siguiente declaración: webview.getWebSettings (). SetAllowUniversalAccessFromFileURLs (false);

Permisos de archivo mproper

La generación de archivos con el permiso “MODE_WORLD_READABLE” permite una lectura global de archivos, por lo que no se recomienda, excepto si se trata de

un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer archivos con el permiso "MODE_WORLD_WRITABLE".

Riesgo	Recomendación
La configuración de permisos de lectura global revela la información contenida en un archivo. Si el permiso de escritura está habilitado, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	Se recomienda generar archivos con permisos de lectura o escritura globales solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado. Un ejemplo de declaración de archivo peligroso puede verse a continuación: file = openFileOutput ("File_Name", Context.MODE_WORLD_READABLE)

Permisos inapropiados del proveedor de contenido

En Android, los Proveedores de contacto son una forma de compartir información entre aplicaciones a través de una API estructurada que permite mantener la integridad de los datos. También es posible establecer permisos de acceso y escritura para los proveedores de contenido. Si estos permisos no están configurados, cualquier aplicación podría tener acceso a los datos almacenados.

Riesgo	Recomendación
Una aplicación maliciosa podría acceder y / o modificar los datos almacenados en un proveedor de contenido sin que el usuario lo sepa, afectando la integridad y confidencialidad de dichos datos.	Se recomienda establecer permisos para los proveedores de contenido. Para tal efecto, se debe utilizar la guía de "permisos de uso"; por ejemplo: <uses-permission android: name = "android.permission.READ_USER_DICTIONARY">

Permisos de actividades mproper

En esta etapa, el atacante listará los permisos necesarios para iniciar cada actividad. Con esta información, es posible extraer la superficie de ataque tanto desde dentro como desde fuera del entorno de la aplicación, para que el atacante pueda aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, una actividad que realiza funciones que requieren permisos especiales y que se puede iniciar fuera de la aplicación (exportar) y no requiere permisos, compromete la seguridad debido a que permite que la actividad sea iniciada por cualquier aplicación de dispositivo.

Riesgo	Recomendación
Los riesgos varían según la sensibilidad de la funcionalidad de cada actividad.	Como regla general, se recomienda solicitar permisos sobre las actividades que realizan funciones confidenciales, especialmente si se exportan.

Permisos de servicios inadecuados

En esta etapa, el atacante listará los permisos necesarios para iniciar cada servicio. Con esta información, es posible obtener la superficie de ataque tanto dentro como fuera del entorno de la aplicación, por lo que el atacante podría aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, un servicio que realiza funciones que requieren permisos especiales y que se pueden iniciar fuera de la aplicación (exportar) y no requiere permisos, compromete la seguridad debido a que permite que cualquier aplicación de dispositivo inicie el servicio.

Riesgo	Recomendación

Los riesgos varían según la sensibilidad de la funcionalidad de cada servicio.	Como regla general, se recomienda solicitar permisos en los servicios que realizan funciones confidenciales, especialmente si se exportan

Permisos inadecuados de los receptores de difusión

En esta etapa, el atacante listará los permisos necesarios para iniciar cada receptor de difusión. Con esta información, es posible obtener la superficie de ataque tanto dentro como fuera del entorno de la aplicación, por lo que el atacante podría aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, un receptor de difusión que realiza funciones que requieren permisos especiales y que se pueden iniciar fuera de la aplicación (exportar) y no requiere permisos, compromete la seguridad debido a que permite que cualquier aplicación de dispositivo inicie el receptor de difusión.

Riesgo	Recomendación
Os riesgos varían según la sensibilidad de la funcionalidad de cada receptor de difusión.	Como regla general, se recomienda que se exijan permisos a los receptores de difusión que realizan funciones sensibles, especialmente si se exportan receptores de difusión.

Permisos de base de datos inadecuados

La generación de bases de datos con el permiso "MODE_WORLD_READABLE" habilitado permite la lectura global de la base de datos, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer bases de datos con el permiso "MODE_WORLD_WRITABLE". Un ejemplo de declaración de archivo peligroso puede verse a continuación: SQLiteDatabasemyWorldReadDB = openOrCreateDatabase ("Contacts", MODE_WORLD_READABLE, null);

Riesgo	Recomendación
La configuración de permisos de lectura global revela la información contenida en un archivo. Si se proporciona el permiso de escritura, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	Se recomienda generar archivos con permisos de lectura o escritura globales solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada,

	este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Permisos de preferencias compartidas impropias

La generación de Preferencias Compartidas con el permiso “MODE_WORLD_READABLE” permite una lectura global de Preferencias Compartidas, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer Preferencias Compartidas con el permiso "MODE_WORLD_WRITABLE". Un ejemplo de declaración de archivo peligroso puede verse a continuación: SharedPreferencesprefsAllWrite = getSharedPreferences ("MisPreferenciasWrite", MODE_WORLD_WRITEABLE);

Riesgo	Recomendación
La configuración de permisos de lectura global revela la información contenida en un archivo. Si se proporciona el permiso de escritura, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	Se recomienda generar archivos con permisos de lectura o escritura globales solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada,

	este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Criptografía

En esta sección, se prueban las funcionalidades relacionadas con el uso de criptografías en la aplicación. Esto puede ocurrir al enviar o almacenar datos.

Credenciales codificadas

El uso de contraseñas codificadas o vacías es fácil de extraer para un usuario malintencionado al desensamblar la aplicación.

Riesgo	Recomendación
Un usuario malintencionado podría tener acceso a credenciales codificadas, accediendo a los servicios permitidos por dichas credenciales.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información

Almacenamiento de datos inseguros

El cifrado no es suficiente para garantizar la confidencialidad de la información. Es necesario usar las funciones de encriptación correctamente, usando algoritmos más robustos y la mayor longitud de clave posible. Por ejemplo, es posible usar AES como algoritmo de cifrado en Android, pero se recomienda usar una clave de 256 bits en lugar de 128 bits.

Riesgo	Recomendación
Un usuario malintencionado podría realizar ataques en la criptografía en uso. Tenga en cuenta que un usuario con privilegios de administrador puede	Se recomienda usar AES con claves de 256 bits. Además, se recomienda generar vectores IV a través de SecureRandom y la clave debe derivar

consultar la memoria de los procesos del dispositivo, pudiendo tener acceso a toda la información no encriptada.	de una contraseña que utiliza protocolos conocidos como PBKDF2 (función de derivación de clave basada en contraseña).

Uso inseguro del protocolo de transporte

Las comunicaciones de información a través de los canales HTTP pueden ser fácilmente interceptadas. En el caso de usar HTTPS, un atacante podría usar técnicas de SSLStrip para acceder a dicha información, haciendo necesario el uso de mecanismos adicionales como forzar, desde el lado del cliente, para permitir el envío de información solo a través de un canal seguro de HTTPS.

Riesgo	Recomendación
Un usuario malintencionado podría realizar ataques en la criptografía en uso. En el caso de usar HTTPS, existen varios tipos de ataques; El más conocido es SSLStrip. Si el atacante tiene éxito, se podría acceder a la información que debía enviarse cifrada.	Se recomienda utilizar HTTPS en lugar de HTTP cuando sea posible. Además, se recomienda utilizar mecanismos del lado del cliente que obligan al envío de información a través de un canal seguro; por ejemplo, redirigir a un canal seguro si se detecta un intento de enviar información a través de un canal no cifrado. Además, se recomienda evaluar la necesidad de usar encabezados de seguridad de transporte estricto de HTTP en los servidores web; con dichos encabezados, el navegador forzaría el uso de canales encriptados en las comunicaciones

Fijación de certificados

Si la aplicación intercambia información a través de SSL y no confía en las autoridades de certificación, no hay razón para usar esta confiabilidad porque no brindan seguridad a las comunicaciones de la aplicación. Además, si alguna de estas autoridades de certificación está comprometida, los usuarios de la aplicación no serían vulnerables.

Riesgo	Recomendación
Si una autoridad de certificación está comprometida, el envío de certificados fraudulentos podría tener un impacto en la confidencialidad de la información transmitida, debido al hecho de que la aplicación confiaría en estos certificados porque los	En caso de usar solo SSL para servicios definidos, se recomienda incluir técnicas de identificación de certificados. El navegador Google Chrome ya los usa para los servicios de Google y también para la aplicación oficial de Twitter.

proporciona una autoridad de certificación conocida.	

Política de cifrado
Existe una política explicita para el manejo de las claves criptográficas (si se usan) y se refuerza su ciclo de vida. Idealmente siguiendo un estándar del manejo de claves como el NIST SP 800-57.

Fuga de información
En la sección de autenticación, se verifican las fugas de información en los medios. La información sensible podría estar relacionada con el usuario o con el propio teléfono.

Fuga de información a los archivos de registro
Es posible almacenar información en los registros de Android a través de las siguientes funciones:

Log.v (). Para almacenar información ampliada.
Log.d (). Con el fin de almacenar información de depuración.
Log.i (). Para almacenar información básica.
Log.w (). Con el fin de almacenar información sobre las alertas.
Log.e (). Con el fin de almacenar los registros de errores.

No se recomienda almacenar información confidencial en los registros del dispositivo porque cualquier aplicación con el permiso "READ_LOGS" podría tener acceso a dichos registros.

Riesgo	Recomendación
Cualquier aplicación con acceso a los registros podría extraer información confidencial que podría almacenarse en esos registros.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, estas pautas deben seguirse para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. .

	Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Fuga de información a la tarjeta SD

Es posible almacenar datos en tarjetas SD a través de las siguientes funciones: Con API 7 o inferior. A través de la función getExternalStorageDirectory (). Con API8 o superior. A través de la función getExternalFilesDir (). No se recomienda almacenar datos confidenciales en la tarjeta SD del dispositivo, ya que cualquier aplicación podría extraerlos.

Riesgo	Recomendación
Cualquier aplicación podría tener acceso a los datos confidenciales almacenados en la tarjeta SD.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, se deben seguir las siguientes pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones

	compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Fuga de información a la red

Para enviar datos a la red, es necesario tener el permiso "android.permission.INTERNET" habilitado. Una aplicación con dicho permiso puede enviar información a la red a través de muchas bibliotecas; los más comunes son java.net. * y android.net. *. No se recomienda enviar información confidencial sin cifrar, ya que un atacante podría extraerla detectando el tráfico de la red.

Riesgo	Recomendación
Un atacante podría obtener acceso a información confidencial sin cifrar enviada a la red mediante el rastreo.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, estas pautas deben seguirse para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD.

	Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Fuga de información al IPC

Es posible almacenar información en Intentos a través de las funciones theputExtra () y putExtras () de la clase Intent, así como a través del URI en los parámetros por el método GET a través del parse () de la clase URI. No se recomienda enviar datos confidenciales sin cifrar a través de Intenciones Implícitas, ya que la aplicación que procesará dicha información es desconocida a priori.

Riesgo	Recomendación
Una aplicación ilegítima podría obtener acceso a datos confidenciales al interceptar la Intención.	SSe recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, estas pautas deben seguirse para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros.

	Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado

Validación de datos:
Evaluación de gestión de entrada de usuario En esta categoría, se incluirán las vulnerabilidades relacionadas con la entrada recibida del usuario que es administrada por la aplicación. La validación incorrecta de los datos del usuario es uno de los principales vectores de ataque, ya que podría permitir que un atacante altere el flujo de datos de la aplicación, inyectando código y afectando seriamente la aplicación y los datos almacenados en él. Esta categoría ocupa el 4º lugar en los 10 principales riesgos de la aplicación móvil titulado "Inyección del lado del cliente". A pesar de que el origen de este tipo de vulnerabilidad es el mismo, dependiendo de la ubicación de la interacción de los datos recibidos del usuario, pueden tener lugar diferentes tipos de vulnerabilidades con múltiples impactos.

Cross Site Scripting
Las vulnerabilidades de Cross Site Scripting (XSS) son el resultado de una codificación incorrecta de los datos recibidos de fuentes inseguras antes de usarlos en la salida HTML. En el caso de Android, la vulnerabilidad se deriva básicamente del uso de la función "setJavaScriptEnabled ()" de la clase WebView. Esta función permite incluir objetos Java en el contexto de JavaScript de WebView, por lo que si los datos proporcionados a dicha función no están limpios, se podría ejecutar un código JavaScript arbitrario

Riesgo	Recomendación
Las vulnerabilidades de secuencias de comandos en sitios cruzados (XSS) permiten que un atacante ejecute un código de secuencia de comandos (HTML, JavaScript, VBScript) en el navegador del usuario (en el dominio de la aplicación vulnerable). Por ejemplo, un atacante podría explotar una vulnerabilidad de XSS para extraer cookies de usuario (para luego suplantar a la víctima en la aplicación vulnerable) o solicitar información a los usuarios (como las credenciales). Esta vulnerabilidad se produce como resultado del uso de información insegura en una salida HTML sin la codificación correcta y previa.	Como regla general, la información recibida del usuario no debe ser confiable. Es peligroso usar los datos recibidos directamente del usuario sin codificarlos correctamente en la generación de la página de respuesta. De forma predeterminada, WebView no ejecuta JavaScript, por lo que debe estar habilitado solo cuando sea estrictamente necesario, controlando el origen de la carga dinámica del código a ejecutar. Sin embargo, para deshabilitar expresamente la ejecución de JavaScript, se recomienda ingresar la siguiente declaración: webview.getSettings (). SetJavaScriptEnabled (false); En el caso de utilizar la función

	"addJavascriptInterface ()", se recomienda filtrar los objetos Java proporcionados a WebView a través de listas blancas siempre que sea posible.

Desbordamiento de búfer

Las vulnerabilidades de desbordamiento de búfer son el resultado de no verificar correctamente el tamaño de ciertos datos antes de copiarlos en un área de memoria en particular. En el caso de Android, este tipo de error es fácil de cometer mientras se usa el código nativo si no se valida el tamaño del búfer de datos. Aunque Android incorpora protecciones como ASLR y DEP que reducen la explotabilidad, no resuelven el problema fundamental.

Riesgo	Recomendación
Los riesgos asociados con este tipo de error se consideran muy serios para copiar información en un área de memoria que estaba reservada para otras tareas y almacenar otros datos del programa. Cuando se produce un desbordamiento, los resultados son impredecibles y provocan el final de la ejecución del programa como una regla general, y también es posible provocar una denegación de servicio. Al explotar tales errores, un atacante podría controlar la ejecución del programa, pudiendo acceder al sistema de forma remota por un atacante experimentado.	Como regla general, cualquier operación que implique copiar datos en un área de memoria debe realizarse con cuidado para garantizar que el área de memoria de destino tenga suficiente memoria para contener los datos a copiar. Existe un conjunto de funciones que es más probable que causen errores y no se recomiendan.

Inyección SQL

Las vulnerabilidades de inyección de SQL pueden ocurrir cuando la información proveniente de contenedores inseguros se usa para crear consultas dinámicas de SQL sin implementar controles y / o filtros relevantes para dicha información. Los contenedores inseguros son cualquier fuente de datos del código fuente, como los datos proporcionados por el usuario, las bases de datos no controladas, los servicios web, etc. Android utiliza SQLite para almacenar datos. A continuación se muestra un ejemplo de código vulnerable:
String con = "DELETE FROM case_values WHERE _id =" + paramString; localSQLiteDatabase.execSQL (con);:

Riesgo	Recomendación
Las vulnerabilidades de inyección de SQL podrían permitir que un atacante use caracteres especiales en el	Se recomienda validar cada campo a través de listas blancas. Por lo tanto, solo se aceptarán caracteres no

lenguaje SQL para alterar una declaración que se construye con información externa de una manera dinámica. Como resultado, podría ser posible obtener información arbitraria de la base de datos, alterar esta información y alterar el flujo de trabajo de la aplicación afectada.	peligrosos. Es importante normalizar la entrada para evitar evasiones mediante codificaciones de parámetros. Si es posible, se recomienda usar sentencias preparadas, ya que es la opción más segura actual para construir sentencias SQL con parámetros externos.

Manipulación del camino

Si la generación de archivos depende de la entrada del usuario, es necesario filtrar estos datos porque, de lo contrario, un usuario malintencionado podría tener acceso a información arbitraria utilizando los permisos de la aplicación vulnerable. A continuación se muestra un ejemplo de generación de archivos peligrosos: file = openFileOutput (unfiltered_variable, Context.MODE_WORLD_READABLE); Si "unffiltered_variable" proviene del usuario sin filtrar, el acceso a los archivos por parte de la aplicación podría controlarse, permitiendo que un usuario malintencionado sobrescriba los archivos de la aplicación.

Riesgo	Recomendación
Esta vulnerabilidad pone en riesgo la integridad y la disponibilidad de los archivos administrados por la aplicación.	Se recomienda no utilizar datos provenientes del usuario para crear nombres de archivos. Si es necesario, se recomienda filtrar los datos de entrada, preferiblemente de acuerdo con el uso de listas blancas

Dos Null Check en IPC.

Android finaliza las aplicaciones cuando se encuentra una referencia nula. Si no se verifican las referencias nulas al recuperar datos de un Intent, un atacante podría enviar intentos con campos con valores nulos para denegar el servicio a las aplicaciones, especialmente a aquellas que tienen servicios en ejecución. En particular, se debe verificar la existencia de referencias nulas al recuperar datos de un Intent, especialmente los siguientes métodos: getAction () y getExtra ().:

Riesgo	Recomendación
Esta vulnerabilidad pone en riesgo la disponibilidad de las aplicaciones que no realizan comprobaciones para no intentar acceder a valores nulos. En caso de que los servicios ejecuten los servicios, un atacante podría denegar dichos servicios sin que el usuario lo sepa, por lo que es especialmente dañino.	Se recomienda verificar los valores nulos al recuperar datos de Intents, especialmente los siguientes métodos: getAction () y getExtra ().

Inyección de troncos

Un atacante podría incluir ciertos caracteres especiales en una cadena de texto especialmente manipulada, por lo que cuando se usa en la creación de una nueva entrada en los archivos de registro, el atacante resultará en una manipulación libre de la salida. En el caso de Android, las funciones estándar de registro de información son las siguientes:

Log.v (). Para almacenar información ampliada.
Log.d (). Con el fin de almacenar información de depuración.
Log.i (). Para almacenar información básica.
Log.w (). Con el fin de almacenar información sobre las alertas.
Log.e (). Con el fin de almacenar los registros de errores.

Riesgo	Recomendación
Los archivos de registro podrían estar dañados debido a una entrada falsa o no válida. Esto podría dar como resultado registros no confiables y la necesidad de descartar todos los registros sobre las acciones realizadas en el sistema por los usuarios. Los atacantes podrían usar esta situación para ocultar acciones criminales y evitar que se detecten sus acciones y, por lo tanto, evitar su persecución por los delitos cometidos.	Se deben tener en cuenta todos los metacaracteres utilizados para la aplicación al generar registros. Por ejemplo, si la aplicación utiliza el carácter de canalización “|” (sin las comillas) para formatear la entrada, este será un meta-carácter. Es decir, un atacante podría incluir un extra | y sería imposible reconstruir después si eso | Fue ingresado por la aplicación o por el atacante. Esto se aplica para cualquier carácter especial en los registros.

Intent inyectable

Si la entrada del usuario se carga de forma dinámica en los datos de Intención, un usuario malintencionado podría manipular dichos datos para ejecutar el código a través de ellos. En particular, se debe verificar la existencia de datos dinámicos al incluir dichos datos en un Intent, especialmente a través de los siguientes métodos de Intent: addCategory (), setAction (), setClass (), setClassName (), setComponent (), setData () andsetDataAndType ().:

Riesgo	Recomendación
esta vulnerabilidad pone en riesgo la ejecución normal de la aplicación, ya que si un usuario puede modificar los datos almacenados en los Intentos, el rendimiento normal de la aplicación también podría modificarse, así como la interacción entre las aplicaciones mediante el envío de Intenciones	Se recomienda verificar los valores de entrada del usuario, especialmente aquellos que van a Intent data instantiating la aplicación. Para tal fin, se recomienda inspeccionar los datos de usuario que terminan como parámetros en cualquiera de las siguientes funciones de Intención: addCategory (), setAction (), setClass

	(), setClassName (), setComponent (), setData () y setDataAndType (

Intención Spoofing

Esta categoría cubre todas las vulnerabilidades involucradas en la entrega de intenciones arbitrarias a un componente que espera recibir otro tipo de intención. De esta manera, el atacante utilizará los filtros de la víctima para enviar datos inesperados y, por lo tanto, aprovechar estas funcionalidades. Para tener éxito con este tipo de vulnerabilidad, los componentes de destino deben ser públicos o exportables como regla general; De lo contrario, no estarán disponibles para otras aplicaciones. Para tal fin, los componentes deben tener el indicador "EXPORTADO" habilitado en el archivo AndroidManifest.xml. Tenga en cuenta que los receptores de difusión pueden declararse en AndroidManifest o en tiempo de ejecución.

Inyección de difusión

Los receptores de difusión se utilizan para registrar intenciones y responderlas. Si uno de estos componentes es público, cualquier aplicación podría enviar intenciones. El registro de Intenciones con acciones del sistema por parte de Broadcast Receivers es peligroso, ya que el hecho de registrar tales Intenciones hace que el elemento sea público y accesible, por lo que una aplicación maliciosa podría enviar Intenciones explícitas sin las acciones correspondientes porque son acciones del sistema.

Riesgo	Recomendación
En muchos casos, los receptores de difusión obtienen Intenciones y activan Actividades o Servicios con sus datos. Las vulnerabilidades de la inyección de difusión podrían permitir a un atacante manipular estos componentes para interactuar con ellos mediante la inyección de datos arbitrarios.	Se recomienda usar componentes públicos solo cuando sea necesario y, en tales casos, validar a fondo cada parámetro recuperado de la Intención. Si Broadcast Receivers registra acciones del sistema, se recomienda validar la acción del intento, ya que un intento malicioso no puede registrar las acciones del sistema, por lo que esta validación se puede utilizar para saber si el intento proviene del sistema Android o no.

Lanzamiento de actividad maliciosa

Actividades pueden iniciarse sobre la base de datos provenientes del usuario o la recepción de intenciones arbitrarias. Esto permite varias superficies de ataque: Si una Actividad utiliza datos del Intención sin verificar su origen, la integridad de los datos de la aplicación se verá afectada.

Si una Actividad entrega datos considerados como confidenciales y pueden ser lanzados públicamente, un Intento especialmente manipulado podría lanzarlos, lo que lleva a una divulgación de información confidencial.ndaciones:

Riesgo	Recomendación

La presencia de esta vulnerabilidad podría confundir a un usuario, ya que los usuarios pensarían que están usando una Actividad legítima, cuando en realidad están realizando acciones que se informan al atacante. Adicionalmente, es posible la filtración de la información utilizada por la Actividad.	Se recomienda usar componentes públicos solo cuando sea necesario y, en esos casos, validar a fondo cada parámetro recuperado de la Intención o del usuario.

Lanzamiento de servicio malicioso.

Se recomienda usar componentes públicos solo cuando sea necesario y, en esos casos, validar a fondo cada parámetro recuperado de la Intención o del usuario. Además, los servicios pueden ser protegidos estableciendo permisos.

Riesgo	Recomendación
La presencia de esta vulnerabilidad podría permitir realizar acciones arbitrarias en la aplicación mediante la ejecución de Servicios y también dar lugar a la divulgación de información utilizada por el Servicio. Como circunstancia agravante, los Servicios generalmente confían más en los datos proporcionados por los usuarios y las intenciones; además, en muchos casos proporcionan interfaces API para que las aplicaciones puedan comunicarse entre sí.	Se recomienda usar componentes públicos solo cuando sea necesario y, en esos casos, validar a fondo cada parámetro recuperado de la Intención o del usuario. Además, los servicios pueden ser protegidos estableciendo permisos.

Inseguro Control de Intención Pendiente

Una aplicación puede delegar tareas a otra aplicación a través de intenciones pendientes. La aplicación que recibe la Intención pendiente no puede modificar sus datos, pero completa los datos vacíos. Si una aplicación genera una Intención pendiente con tal propósito pero el objetivo no está definido, una aplicación que reciba la Intención pendiente podría agregarla, por lo que dichos datos se enviarían a un componente arbitrario.

Riesgo	Recomendación
Esta vulnerabilidad podría permitir la filtración de la información que está siendo procesada por la aplicación que generó la Intención pendiente. Además, una aplicación maliciosa podría recibir, procesar y enviar dicha información, realizando un ataque Man-in-the-Middle.	Cuando las tareas se delegan a otra aplicación a través de una Intención pendiente, se recomienda establecer siempre el destino en el que se delegan las tareas para que sean ejecutadas por una aplicación de terceros.

Intención no autorizada

Esta categoría cubre todas las vulnerabilidades relacionadas con la resolución de la entrega de Intención implícita. Cuando una aplicación envía un Intent implícito, no hay garantía de que una aplicación malintencionada no recopile dicho Intent, ya que una aplicación maliciosa podría registrar un Intent Filter capaz de pasar la resolución (acción, datos y categoría), a menos que dicho Intent tenga un conjunto de permisos requeridos que el usuario malicioso no tiene. Si una aplicación malintencionada puede interceptar Intenciones implícitas, podría tener acceso a la ejecución del flujo de datos, pudiendo realizar ataques de denegación de servicio o phishing. Esta categoría considera cómo este tipo de vulnerabilidad puede ser expuesta en componentes particulares: Broadcast, Activities y Services.

Robo de transmisiones

Una intención implícita es pública cuando no está protegida con los permisos de tipo "Signature" o "SignatureOrSystem". Cuando una aplicación envía un intento público implícito, dicho componente puede leer dicho intento. Una aplicación malintencionada podría tener acceso a él, obteniendo datos confidenciales si el intento simplemente contiene filtros de intento para todas las acciones, datos y categorías. Esto es especialmente peligroso en Sticky Broadcasts, ya que el Intento persiste en ellos y puede redirigirse repetidamente a varios destinatarios. Esto le da una mayor ventana de tiempo en la que el componente malicioso puede operar. Además, Sticky Broadcasts no puede protegerse mediante permisos.

Riesgo	Recomendación
Un atacante podría realizar un ataque de denegación de servicio en las transmisiones ordenadas, ya que un Intent solo se puede propagar sobre ellos si el primer componente que recibe el Intent para usarlo como salida. Además, podría usarse para realizar ataques Man-in-the-Middle con su posterior inyección de datos en los Intentos de propagación.	Se recomienda considerar la necesidad de enviar Intentos implícitos. Para comunicar intenciones en una aplicación, siempre se recomienda el uso de intenciones explícitas. Cuando es necesario enviar datos confidenciales en un Intento, se recomienda protegerlos configurando permisos y cifrando dichos datos.

Secuestro de actividad

Al aprovechar esta vulnerabilidad, se lanza una Actividad maliciosa en lugar de la esperada, por lo que el usuario estará en una aplicación incorrecta sin ser consciente. Esto sucede cuando el cargo de una Actividad depende de una Intención implícita. El atacante registra un Intent Filter más preciso y lo controla. Esto no siempre es posible, ya que si varias Actividades son capaces de procesar una Intención implícita, se mostrará un mensaje al usuario para que elija la aplicación. Sin embargo, esto es muy peligroso porque si esto sucede y el usuario configura como acción predeterminada la carga de la aplicación maliciosa, la Actividad maliciosa siempre se abrirá sin que se le solicite.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite ejecutar ataques de phishing, así como también fugas de la información que maneja el usuario en la Actividad involucrada. Además, esta vulnerabilidad permite al atacante modificar los datos, poniendo en riesgo su integridad.	Se recomienda considerar la necesidad de enviar Intentos implícitos. Para comunicar intenciones en una aplicación, siempre se recomienda el uso de intenciones explícitas. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos configurando permisos y cifrando dichos datos.

Servicio de secuestro

Al aprovechar esta vulnerabilidad, se lanza una Actividad maliciosa en lugar de la esperada, por lo que el usuario estará en una aplicación incorrecta sin ser consciente. Esto sucede cuando el cargo de una Actividad depende de una Intención implícita. El atacante registra un Intent Filter más preciso y lo controla. Esta vulnerabilidad es más persistente debido a que es transparente para el usuario porque los servicios no incluyen una interfaz gráfica para ella.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite la filtración de la información procesada por el Servicio involucrado. Además, esta vulnerabilidad permite que un atacante modifique los datos, poniendo en riesgo su integridad	Se recomienda considerar la necesidad de enviar Intentos implícitos. Para comunicar intenciones en una aplicación, siempre se recomienda el uso de intenciones explícitas. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos configurando permisos y cifrando dichos datos.

Debilidad intenciones especiales.

Si se envía un intento implícito para tratar con un proveedor de contenido, puede contener información confidencial en el URI, pudiendo leerse si un componente malicioso intercepta el intento. Por otro lado, un intento pendiente conserva los permisos de su creador, por lo que si un componente malintencionado intercepta un intento pendiente, podría usar los permisos de la creación del intento pendiente para suplantarlo.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite la filtración de información incluida en el URI. Además, esta vulnerabilidad permite al atacante suplantar los permisos del creador de la intención pendiente.	Se recomienda evaluar la necesidad de enviar intenciones implícitas. Para comunicar intenciones en una aplicación, se recomienda utilizar intenciones explícitas. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos

	estableciendo permisos y cifrando dichos datos.

Lógica de negocio.

En esta categoría, se incluyen las vulnerabilidades con componentes más centrados en el diseño en lugar de la codificación. Se incluyen tanto la casuística de rendimiento como la capacidad de la aplicación para trabajar de una manera inesperada que afecta su flujo de trabajo.

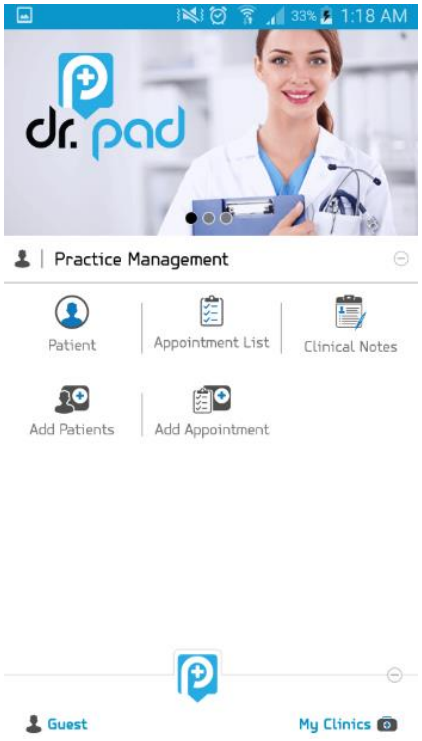
Los ataques dirigidos a lógicas de negocios de aplicaciones son peligrosos, difíciles de detectar y especialmente dirigidos a la aplicación.

5. RESULTADOS Y DISCUSIÓN

En este apartado se consignan los resultados de la prueba de seguridad en una aplicación de mHealth derivado de la propuesta de metodología de seguridad en aplicaciones mHealth.

Test de seguridad para aplicaciones mHealth a partir de la metodología propuesta

Información de la aplicación



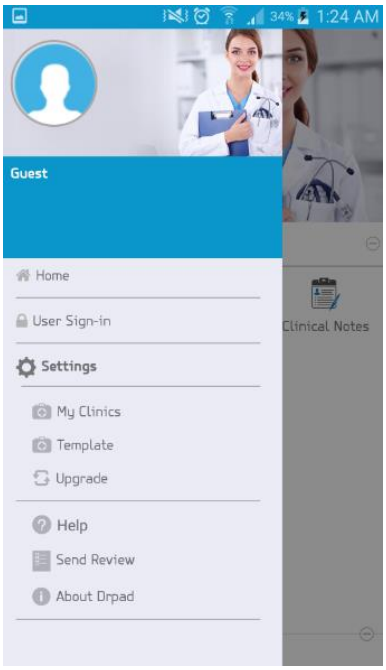
Recopilación de la información

Información app	Descripción
MHealth	Ultima actualizacion: 26 de mayo 2017 tamaño: 9.7m

	version: 1.30 android: 4.0.3 o posteriores permisos: identidad, contactos, ubicación, sms, telefono, fotos, almacenamiento, conexión, id dispositivo

Obtener la Definición de la arquitectura de la aplicación y los servicios.

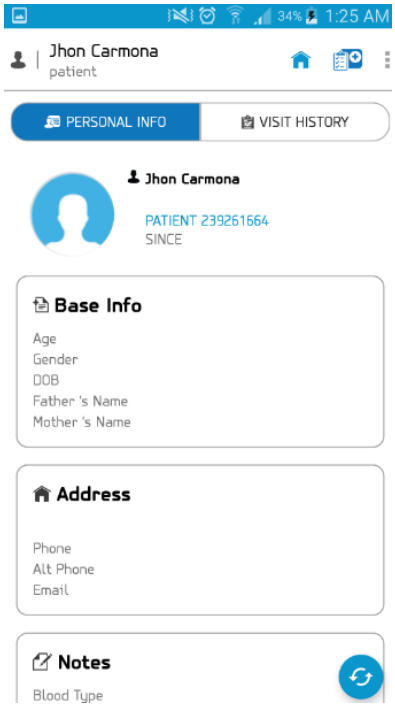
En esta etapa, Se puede obtener la información para saber si se definió una arquitectura de alto nivel para la aplicación y los servicios y si incluyeron controles de seguridad en la misma.



Requerimiento de Información	Descripción
Validar arquitectura de la app	De acuerdo a las pruebas realizas y los niveles de seguridad de la aplicación no es permitido ver la arquitectura de la aplicación

Identificación de los campos de información sensible que maneja la aplicación.

En esta etapa se evalúan los tipos de datos que maneja la aplicación y se define cuál de ellos es sensible, con el fin de saber que controles de seguridad debo manejar para su tratamiento.



Requerimientos de Informacion	Descripcion
Datos sensibles que pueden ser catalogados como críticos utilizados por la aplicación	Registros de pacientes, solicitud de datos personales, dirección, edad, nombres de padres, teléfono, etc.

Información general

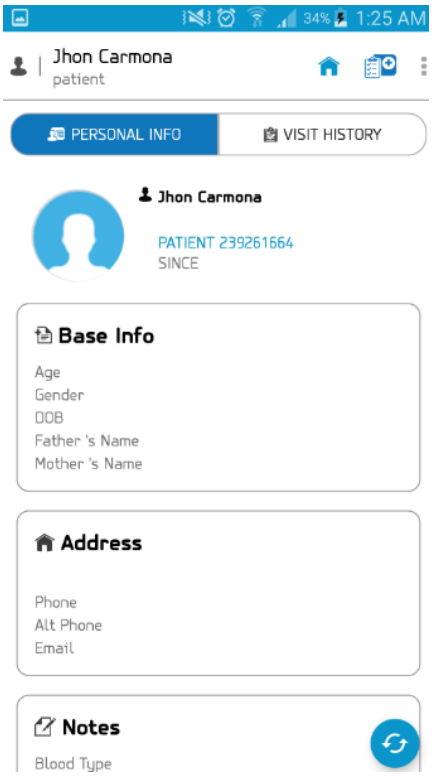
En esta etapa, se recopila información general sobre la aplicación. Algunos parámetros de interés se describen a continuación:

Nombre del paquete a analizar: Dr. Pad

GID asociado al paquete en el dispositivo: Identificadores de los grupos asociados a la aplicación: Pacientes, Doctores, clínicas.

Etiquetas de usuario compartidas: Cadenas asociadas a los identificadores de usuario de Linux con los que se ejecuta la aplicación.

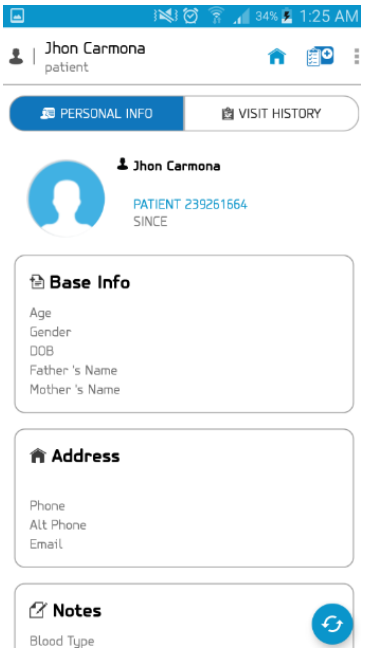
- Nombre de la versión de la aplicación:
- Número de versión de la aplicación:
- Fecha de instalación de la aplicación:
- Fecha de actualización más reciente:
- Ruta de la aplicación en el dispositivo:
- Bibliotecas compartidas:
- Ruta de APK en el dispositivo:
- Aplicación Target SDK:
- ¿Está habilitado el modo de depuración?
- Permisos requeridos:
- Intención de lanzamiento de la aplicación: Intención necesaria para iniciar la aplicación.



Requerimiento de Información	Descripcion
Un atacante extraerá información para adquirir conocimiento sobre la aplicación, mientras que la información global es necesaria para el funcionamiento de la aplicación, por lo que su exposición en sí no constituye un riesgo para la seguridad. El valor de cualquiera de los parámetros puede convertirse en un riesgo menor; por ejemplo, si el modo de depuración está habilitado.	El atacante podrá extraer información dado que la aplicación no solicita ningún tipo de validación de datos al momento de ingresar a ella.

Lista de componentes

En esta etapa, se recopila información sobre los componentes de una aplicación. Estos son los componentes básicos de una aplicación de Android: Ocupaciones, Servicios, Proveedores de contenido, Receptores de radiodifusión.



Requerimiento de Información	Descripcion
Un atacante obtendrá la información para adquirir conocimiento sobre la aplicación, mientras que la lista de componentes de una aplicación es necesaria para ejecutar dicha aplicación, por lo que su exposición en sí, constituye un riesgo para la seguridad.	Al no tener ninguna restricción de acceso cualquier atacante podrá obtener la información almacenada en la aplicación.

Necesidad de los componentes

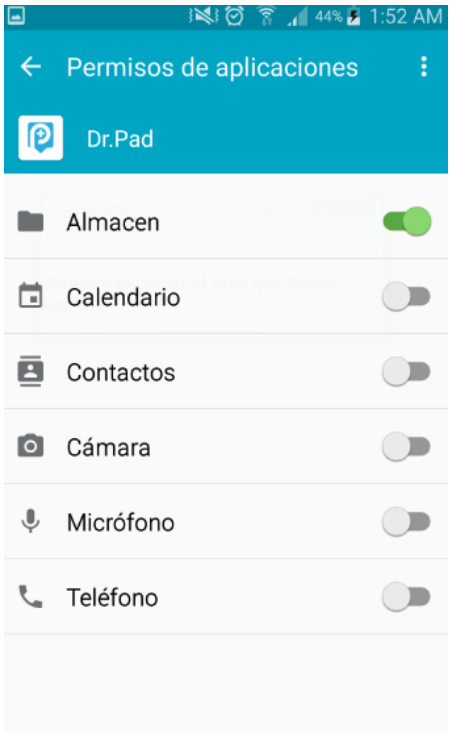
Todos los componentes de la aplicación están definidos en términos de la lógica de negocio o las funciones de seguridad que proveen.

Requerimiento de información	Descripción
El atacante puede encontrar puertas traseras en los componentes que no son necesarios.	No aplica, no permite ver los componentes de la aplicación.

Permisos de acceso a los componentes

Los permisos de acceso necesarios se pueden definir para cada componente de la aplicación de Android. Un atacante enumerará dichos permisos para acceder a

componentes con funcionalidad sensible que no tienen las restricciones correspondientes.



Riesgo	Recomendación
Se puede acceder a un componente sin permisos definidos desde una aplicación sin restricciones, incluso se puede acceder desde cualquier otra aplicación del dispositivo si se exporta.	Las restricciones a accesos en el dispositivo serán asignado por el usuario a la aplicación restringiendo que cualquier atacante acceda a información del equipo.

Validar si los componentes tienen habilitados la posibilidad de exportarse

Los componentes de la aplicación se pueden exportar si el indicador "exportado" en AndroidManifest está habilitado. Cuando se exporta un componente, se puede iniciar desde cualquier aplicación en el dispositivo.

Riesgo	Recomendación
Si la funcionalidad del componente es confidencial o necesita permisos que se hayan habilitado en la aplicación, un componente exportado incorrectamente podría permitir ejecutar dicha funcionalidad sensible	Se recomienda exportar los componentes solo cuando sea estrictamente necesario. Al exportar un componente, se recomienda establecer un permiso, por lo que el lanzamiento necesitará un grado de seguridad adicional.

desde otra aplicación hasta su lanzamiento.	

Validar si los componentes están en launchmode

Un atacante buscará la intención necesaria para lanzar el componente. Por lo tanto, si el componente es accesible (exportado en el caso de otras aplicaciones de dispositivos), podrá iniciarlo desde otra aplicación o componente.

Riesgo	Recomendación
Un atacante obtendrá la información para adquirir conocimiento sobre la aplicación, mientras que la lista de Intenciones de Lanzamiento de un componente de la aplicación es necesaria para ejecutar dicha aplicación, por lo que su exposición en sí no constituye un riesgo para la seguridad.	Esta información es necesaria para desarrollar aplicaciones, por lo que no es posible restringir el acceso a dicha información.

Diseño de modelo de amenazas

Realizar el levantamiento de un modelado de amenazas para la aplicación móvil y los servicios en el que se definieron las mismas y sus contramedidas.

Configuración y gestión de la implementación

Los errores en la configuración de la aplicación o los componentes comprometen la seguridad de la aplicación. En esta etapa, se definen varios errores en la configuración o en las opciones de implementación de la aplicación.

Validar si está habilitada la Depuración sin restricciones

Es habitual habilitar el modo de depuración mientras se desarrolla una aplicación para extraer información sobre su funcionamiento. Sin embargo, este modo debe estar deshabilitado al desarrollar aplicaciones.

Riesgo	Recomendación
La aplicación en el modo de depuración proporciona información	No aplica porque es una aplicación que ya esta desarrollada

que un atacante podría usar para realizar ataques en dicha aplicación.	

Validar el Uso de bibliotecas no actualizadas

El uso de bibliotecas desactualizadas podría causar vulnerabilidades que podrían poner en riesgo la aplicación y los datos procesados por dicha aplicación.

Riesgo	Recomendación
Es más probable que las bibliotecas desactualizadas contengan vulnerabilidades que pueden poner en riesgo la aplicación y los datos procesados por dicha aplicación.	Depende netamente de las actualizaciones que lance el proveedor de la aplicación.

Validar los Archivos predeterminados y de copia de seguridad

En esta etapa, el atacante busca versiones anteriores modificadas de archivos modificados, incluye archivos que se han cargado en el lenguaje de programación en uso y se pueden descargar como código fuente o incluso archivos de copia de seguridad automática o manual como archivos comprimidos. En el caso de Android, estos archivos pueden dejarse por error en la estructura de archivos APK.

Riesgo	Recomendación
Todos estos archivos podrían permitir que un atacante acceda a operaciones internas, puertas traseras, interfaces administrativas o incluso credenciales para conectarse a la interfaz administrativa o al servidor de bases de datos.	Antes de empaquetar una aplicación de Android, se recomienda asegurarse de que los archivos que pertenecen a la aplicación sean estrictamente necesarios y no se incluya ningún archivo que contenga información innecesaria o confidencial.

Metadatos sobre los archivos

En esta etapa, un atacante examinará los metadatos de los archivos incluidos en la aplicación para buscar información útil durante el ataque de penetración.

Riesgo	Recomendación
Dependiendo de la información almacenada en los metadatos, el riesgo puede variar mucho. Los metadatos típicos que se pueden encontrar en los archivos de	Se recomienda eliminar los metadatos de los archivos incluidos en la aplicación para no proporcionar información innecesaria. Existen herramientas de código abierto que

aplicaciones de Android son los relacionados con las imágenes contenidas en las aplicaciones.	permiten eliminar metadatos en muchos formatos diferentes; por ejemplo, Exiftool.

Insuficiente endurecimiento de WebView

WebView es una clase de Android que permite mostrar contenido en línea dentro de las Actividades de Android. Sin embargo, al instalar WebView en aplicaciones Android, se debe tener en cuenta el hecho de que una configuración deficiente de esta extensión podría exponer al usuario de la aplicación a una multitud de riesgos.

Riesgo	Recomendación
Un endurecimiento insuficiente puede implicar riesgos para el usuario, convirtiéndose en posibles objetivos para ataques a través de la web.	Deshabilitar el soporte de los complementos del navegador; por ejemplo, con la siguiente declaración: webview.getSettings (). setPluginsEnabled (false); Deshabilitar el acceso a archivos locales; por ejemplo, con la siguiente declaración: webview.getSettings (). setAllowFileAccess (false); Evite cargar contenidos de sitios de terceros, verificando que WebView solo pueda acceder a aquellos sitios web que la aplicación necesita, usando listas blancas para tal efecto cuando sea posible. Deshabilitando JavaScript, con la siguiente declaración: webview.getWebSettings (). SetJavaScriptEnabled (false); Deshabilita el acceso a la URL de contenido dentro de WebView. El acceso a la URL de contenido permite que WebView cargue contenido de un proveedor de contenido instalado en el sistema, con la siguiente declaración: webview.getWebSettings (). SetsetAllowContentAccess (false);

	Establece si se debe permitir que JavaScript que se ejecuta en el contexto de una URL de esquema de archivos acceda al contenido de otras URL de esquemas de archivos, con la siguiente declaración: webview.getWebSettings (). SetAllowFileAccessFromFileURLs (false); Establece si se debe permitir que JavaScript que se ejecuta en el contexto de una URL de esquema de archivos acceda al contenido desde cualquier origen, con la siguiente declaración: webview.getWebSettings (). SetAllowUniversalAccessFromFileURLs (false);

Permisos de archivo mproper

La generación de archivos con el permiso “MODE_WORLD_READABLE” permite una lectura global de archivos, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer archivos con el permiso "MODE_WORLD_WRITABLE".

Riesgo	Recomendación
La configuración de permisos de lectura global revela la información contenida en un archivo. Si el permiso de escritura está habilitado, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	Se recomienda generar archivos con permisos de lectura o escritura globales solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros.

	Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado. Un ejemplo de declaración de archivo peligroso puede verse a continuación: file = openFileOutput ("File_Name", Context.MODE_WORLD_READABLE)

Permisos inapropiados del proveedor de contenido

En Android, los Proveedores de contacto son una forma de compartir información entre aplicaciones a través de una API estructurada que permite mantener la integridad de los datos. También es posible establecer permisos de acceso y escritura para los proveedores de contenido. Si estos permisos no están configurados, cualquier aplicación podría tener acceso a los datos almacenados.

Riesgo	Recomendación
Una aplicación maliciosa podría acceder y / o modificar los datos almacenados en un proveedor de contenido sin que el usuario lo sepa, afectando la integridad y confidencialidad de dichos datos.	Se recomienda establecer permisos para los proveedores de contenido. Para tal efecto, se debe utilizar la guía de "permisos de uso"; por ejemplo: <uses-permission android: name = "android.permission.READ_USER_DICTIONARY">

Permisos de actividades mproper

En esta etapa, el atacante listará los permisos necesarios para iniciar cada actividad. Con esta información, es posible extraer la superficie de ataque tanto desde dentro como desde fuera del entorno de la aplicación, para que el atacante pueda aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, una actividad que realiza funciones que requieren permisos especiales y que se puede iniciar fuera de la aplicación (exportar) y no requiere permisos, compromete la seguridad debido a que permite que la actividad sea iniciada por cualquier aplicación de dispositivo.

Riesgo	Recomendación
Los riesgos varían según la sensibilidad de la funcionalidad de cada actividad.	Como regla general, se recomienda solicitar permisos sobre las actividades que realizan funciones confidenciales, especialmente si se exportan.

Permisos de servicios inadecuados

En esta etapa, el atacante listará los permisos necesarios para iniciar cada servicio. Con esta información, es posible obtener la superficie de ataque tanto dentro como fuera del entorno de la aplicación, por lo que el atacante podría aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, un servicio que realiza funciones que requieren permisos especiales y que se pueden iniciar fuera de la aplicación (exportar) y no requiere permisos, compromete la seguridad debido a que permite que cualquier aplicación de dispositivo inicie el servicio.

Riesgo	Recomendación
Los riesgos varían según la sensibilidad de la funcionalidad de cada servicio.	Como regla general, se recomienda solicitar permisos en los servicios que realizan funciones confidenciales, especialmente si se exportan

Permisos inadecuados de los receptores de difusión

En esta etapa, el atacante listará los permisos necesarios para iniciar cada receptor de difusión. Con esta información, es posible obtener la superficie de ataque tanto dentro como fuera del entorno de la aplicación, por lo que el atacante podría

aprender mejor cómo atacar la seguridad de la aplicación en el futuro. Por ejemplo, un receptor de difusión que realiza funciones que requieren permisos especiales y que se pueden iniciar fuera de la aplicación (exportar) y no requiere permisos, compromete la seguridad debido a que permite que cualquier aplicación de dispositivo inicie el receptor de difusión.

Riesgo	Recomendación
Os riesgos varían según la sensibilidad de la funcionalidad de cada receptor de difusión.	Como regla general, se recomienda que se exijan permisos a los receptores de difusión que realizan funciones sensibles, especialmente si se exportan receptores de difusión.

Permisos de base de datos inadecuados

La generación de bases de datos con el permiso "MODE_WORLD_READABLE" habilitado permite la lectura global de la base de datos, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer bases de datos con el permiso "MODE_WORLD_WRITABLE". Un ejemplo de declaración de archivo peligroso puede verse a continuación: SQLiteDatabasemyWorldReadDB = openOrCreateDatabase ("Contacts", MODE_WORLD_READABLE, null);

Riesgo	Recomendación
La configuración de permisos de lectura global revela la información contenida en un archivo. Si se proporciona el permiso de escritura, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	Se recomienda generar archivos con permisos de lectura o escritura globales solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es

	accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Permisos de preferencias compartidas impropias

La generación de Preferencias Compartidas con el permiso “MODE_WORLD_READABLE” permite una lectura global de Preferencias Compartidas, por lo que no se recomienda, excepto si se trata de un archivo que no revela información confidencial. Del mismo modo, no se recomienda establecer Preferencias Compartidas con el permiso "MODE_WORLD_WRITABLE". Un ejemplo de declaración de archivo peligroso puede verse a continuación: SharedPreferencesprefsAllWrite = getSharedPreferences ("MisPreferenciasWrite", MODE_WORLD_WRITEABLE);

Riesgo	Recomendación
La configuración de permisos de lectura global revela la información contenida en un archivo. Si se proporciona el permiso de escritura, cualquier aplicación podría modificar el contenido de dicho archivo, extendiendo la superficie de ataque.	Se recomienda generar archivos con permisos de lectura o escritura globales solo cuando sea estrictamente necesario. En términos generales, se recomienda seguir estas pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores.

	Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Criptografía

En esta sección, se prueban las funcionalidades relacionadas con el uso de criptografías en la aplicación. Esto puede ocurrir al enviar o almacenar datos.

Credenciales codificadas

El uso de contraseñas codificadas o vacías es fácil de extraer para un usuario malintencionado al desensamblar la aplicación.

Riesgo	Recomendación
Un usuario malintencionado podría tener acceso a credenciales codificadas, accediendo a los servicios permitidos por dichas credenciales.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información

Almacenamiento de datos inseguros

El cifrado no es suficiente para garantizar la confidencialidad de la información. Es necesario usar las funciones de encriptación correctamente, usando algoritmos más robustos y la mayor longitud de clave posible. Por ejemplo, es posible usar AES como algoritmo de cifrado en Android, pero se recomienda usar una clave de 256 bits en lugar de 128 bits.

Riesgo	Recomendación
Un usuario malintencionado podría realizar ataques en la criptografía en uso. Tenga en cuenta que un usuario con privilegios de administrador puede consultar la memoria de los procesos del dispositivo, pudiendo tener acceso a toda la información no encriptada.	Se recomienda usar AES con claves de 256 bits. Además, se recomienda generar vectores IV a través de SecureRandom y la clave debe derivar de una contraseña que utiliza protocolos conocidos como PBKDF2 (función de derivación de clave basada en contraseña).

Uso inseguro del protocolo de transporte

Las comunicaciones de información a través de los canales HTTP pueden ser fácilmente interceptadas. En el caso de usar HTTPS, un atacante podría usar técnicas de SSLStrip para acceder a dicha información, haciendo necesario el uso de mecanismos adicionales como forzar, desde el lado del cliente, para permitir el envío de información solo a través de un canal seguro de HTTPS.

Riesgo	Recomendación
Un usuario malintencionado podría realizar ataques en la criptografía en uso. En el caso de usar HTTPS, existen varios tipos de ataques; El más conocido es SSLStrip. Si el atacante tiene éxito, se podría acceder a la información que debía enviarse cifrada.	Se recomienda utilizar HTTPS en lugar de HTTP cuando sea posible. Además, se recomienda utilizar mecanismos del lado del cliente que obligan al envío de información a través de un canal seguro; por ejemplo, redirigir a un canal seguro si se detecta un intento de enviar información a través de un canal no cifrado. Además, se recomienda evaluar la necesidad de usar encabezados de seguridad de transporte estricto de HTTP en los servidores web; con dichos encabezados, el navegador forzaría el uso de canales encriptados en las comunicaciones

Fijación de certificados

Si la aplicación intercambia información a través de SSL y no confía en las autoridades de certificación, no hay razón para usar esta confiabilidad porque no brindan seguridad a las comunicaciones de la aplicación. Además, si alguna de estas autoridades de certificación está comprometida, los usuarios de la aplicación no serían vulnerables.

Riesgo	Recomendación
Si una autoridad de certificación está comprometida, el envío de certificados fraudulentos podría tener un impacto en la confidencialidad de la información transmitida, debido al hecho de que la aplicación confiaría en estos certificados porque los proporciona una autoridad de certificación conocida.	En caso de usar solo SSL para servicios definidos, se recomienda incluir técnicas de identificación de certificados. El navegador Google Chrome ya los usa para los servicios de Google y también para la aplicación oficial de Twitter.

Política de cifrado

Existe una política explicita para el manejo de las claves criptográficas (si se usan) y se refuerza su ciclo de vida. Idealmente siguiendo un estándar del manejo de claves como el NIST SP 800-57.

Fuga de información

En la sección de autenticación, se verifican las fugas de información en los medios. La información sensible podría estar relacionada con el usuario o con el propio teléfono.

Fuga de información a los archivos de registro

Es posible almacenar información en los registros de Android a través de las siguientes funciones:

Log.v (). Para almacenar información ampliada.

Log.d (). Con el fin de almacenar información de depuración.

Log.i (). Para almacenar información básica.

Log.w (). Con el fin de almacenar información sobre las alertas.

Log.e (). Con el fin de almacenar los registros de errores.

No se recomienda almacenar información confidencial en los registros del dispositivo porque cualquier aplicación con el permiso "READ_LOGS" podría tener acceso a dichos registros.

Riesgo	Recomendación
Cualquier aplicación con acceso a los registros podría extraer información confidencial que podría almacenarse en esos registros.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, estas pautas deben seguirse para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda

	maximizar las precauciones y utilizar técnicas de cifrado.

Fuga de información a la tarjeta SD

Es posible almacenar datos en tarjetas SD a través de las siguientes funciones: Con API 7 o inferior. A través de la función getExternalStorageDirectory (). Con API8 o superior. A través de la función getExternalFilesDir (). No se recomienda almacenar datos confidenciales en la tarjeta SD del dispositivo, ya que cualquier aplicación podría extraerlos.

Riesgo	Recomendación
Cualquier aplicación podría tener acceso a los datos confidenciales almacenados en la tarjeta SD.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, se deben seguir las siguientes pautas para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar

	datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Fuga de información a la red

Para enviar datos a la red, es necesario tener el permiso "android.permission.INTERNET" habilitado. Una aplicación con dicho permiso puede enviar información a la red a través de muchas bibliotecas; los más comunes son java.net. * y android.net. *. No se recomienda enviar información confidencial sin cifrar, ya que un atacante podría extraerla detectando el tráfico de la red.

Riesgo	Recomendación
Un atacante podría obtener acceso a información confidencial sin cifrar enviada a la red mediante el rastreo.	Se recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, estas pautas deben seguirse para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. . Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros.

	Conexión de red. Para almacenar datos en servicios a través de la red. En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado.

Fuga de información al IPC

Es posible almacenar información en Intentos a través de las funciones theputExtra () y putExtras () de la clase Intent, así como a través del URI en los parámetros por el método GET a través del parse () de la clase URI. No se recomienda enviar datos confidenciales sin cifrar a través de Intenciones Implícitas, ya que la aplicación que procesará dicha información es desconocida a priori.

Riesgo	Recomendación
Una aplicación ilegítima podría obtener acceso a datos confidenciales al interceptar la Intención.	SSe recomienda almacenar información confidencial en lugares a los que solo pueda acceder la aplicación y, de preferencia, cifrar dicha información. En términos generales, estas pautas deben seguirse para almacenar datos: Preferencias compartidas. Para almacenar opciones como pares de valores. Almacenamiento interno. Para almacenar datos en la memoria del dispositivo. De forma predeterminada, este tipo de almacenamiento no es accesible por aplicaciones de terceros. Almacenamiento externo. Para almacenar datos en ubicaciones compartidas por todas las aplicaciones del dispositivo, como las tarjetas SD. Base de datos SQLite. Para almacenar datos en BBDD SQLite. De forma predeterminada, este tipo de bases de datos no son accesibles por aplicaciones de terceros. Conexión de red. Para almacenar datos en servicios a través de la red.

	En tales casos, se recomienda maximizar las precauciones y utilizar técnicas de cifrado

Validación de datos:

Evaluación de gestión de entrada de usuario En esta categoría, se incluirán las vulnerabilidades relacionadas con la entrada recibida del usuario que es administrada por la aplicación. La validación incorrecta de los datos del usuario es uno de los principales vectores de ataque, ya que podría permitir que un atacante altere el flujo de datos de la aplicación, inyectando código y afectando seriamente la aplicación y los datos almacenados en él. Esta categoría ocupa el 4º lugar en los 10 principales riesgos de la aplicación móvil titulado "Inyección del lado del cliente". A pesar de que el origen de este tipo de vulnerabilidad es el mismo, dependiendo de la ubicación de la interacción de los datos recibidos del usuario, pueden tener lugar diferentes tipos de vulnerabilidades con múltiples impactos.

Cross Site Scripting

Las vulnerabilidades de Cross Site Scripting (XSS) son el resultado de una codificación incorrecta de los datos recibidos de fuentes inseguras antes de usarlos en la salida HTML. En el caso de Android, la vulnerabilidad se deriva básicamente del uso de la función "setJavaScriptEnabled ()" de la clase WebView. Esta función permite incluir objetos Java en el contexto de JavaScript de WebView, por lo que si los datos proporcionados a dicha función no están limpios, se podría ejecutar un código JavaScript arbitrario

Riesgo	Recomendación
Las vulnerabilidades de secuencias de comandos en sitios cruzados (XSS) permiten que un atacante ejecute un código de secuencia de comandos (HTML, JavaScript, VBScript) en el navegador del usuario (en el dominio de la aplicación vulnerable). Por ejemplo, un atacante podría explotar una vulnerabilidad de XSS para extraer	Como regla general, la información recibida del usuario no debe ser confiable. Es peligroso usar los datos recibidos directamente del usuario sin codificarlos correctamente en la generación de la página de respuesta. De forma predeterminada, WebView no ejecuta JavaScript, por lo que debe estar habilitado solo cuando sea

cookies de usuario (para luego suplantar a la víctima en la aplicación vulnerable) o solicitar información a los usuarios (como las credenciales). Esta vulnerabilidad se produce como resultado del uso de información insegura en una salida HTML sin la codificación correcta y previa.	estrictamente necesario, controlando el origen de la carga dinámica del código a ejecutar. Sin embargo, para deshabilitar expresamente la ejecución de JavaScript, se recomienda ingresar la siguiente declaración: webview.getSettings (). SetJavaScriptEnabled (false); En el caso de utilizar la función "addJavascriptInterface ()", se recomienda filtrar los objetos Java proporcionados a WebView a través de listas blancas siempre que sea posible.

Desbordamiento de búfer

Las vulnerabilidades de desbordamiento de búfer son el resultado de no verificar correctamente el tamaño de ciertos datos antes de copiarlos en un área de memoria en particular. En el caso de Android, este tipo de error es fácil de cometer mientras se usa el código nativo si no se valida el tamaño del búfer de datos. Aunque Android incorpora protecciones como ASLR y DEP que reducen la explotabilidad, no resuelven el problema fundamental.

Riesgo	Recomendación
Los riesgos asociados con este tipo de error se consideran muy serios para copiar información en un área de memoria que estaba reservada para otras tareas y almacenar otros datos del programa. Cuando se produce un desbordamiento, los resultados son impredecibles y provocan el final de la ejecución del programa como una regla general, y también es posible provocar una denegación de servicio. Al explotar tales errores, un atacante podría controlar la ejecución del programa, pudiendo acceder al sistema de forma remota por un atacante experimentado.	Como regla general, cualquier operación que implique copiar datos en un área de memoria debe realizarse con cuidado para garantizar que el área de memoria de destino tenga suficiente memoria para contener los datos a copiar. Existe un conjunto de funciones que es más probable que causen errores y no se recomiendan.

Inyección SQL

Las vulnerabilidades de inyección de SQL pueden ocurrir cuando la información proveniente de contenedores inseguros se usa para crear consultas dinámicas de SQL sin implementar controles y / o filtros relevantes para dicha información. Los contenedores inseguros son cualquier fuente de datos del código fuente, como los datos proporcionados por el usuario, las bases de datos no controladas, los servicios web, etc. Android utiliza SQLite para almacenar datos. A continuación se muestra un ejemplo de código vulnerable: String con = "DELETE FROM case_values WHERE _id =" + paramString; localSQLiteDatabase.execSQL (con);:

Riesgo	Recomendación
Las vulnerabilidades de inyección de SQL podrían permitir que un atacante use caracteres especiales en el lenguaje SQL para alterar una declaración que se construye con información externa de una manera dinámica. Como resultado, podría ser posible obtener información arbitraria de la base de datos, alterar esta información y alterar el flujo de trabajo de la aplicación afectada.	Se recomienda validar cada campo a través de listas blancas. Por lo tanto, solo se aceptarán caracteres no peligrosos. Es importante normalizar la entrada para evitar evasiones mediante codificaciones de parámetros. Si es posible, se recomienda usar sentencias preparadas, ya que es la opción más segura actual para construir sentencias SQL con parámetros externos.

Manipulación del camino

Si la generación de archivos depende de la entrada del usuario, es necesario filtrar estos datos porque, de lo contrario, un usuario malintencionado podría tener acceso a información arbitraria utilizando los permisos de la aplicación vulnerable. A continuación se muestra un ejemplo de generación de archivos peligrosos: file = openFileOutput (unfiltered_variable, Context.MODE_WORLD_READABLE); Si "unffiltered_variable" proviene del usuario sin filtrar, el acceso a los archivos por parte de la aplicación podría controlarse, permitiendo que un usuario malintencionado sobrescriba los archivos de la aplicación.

Riesgo	Recomendación
Esta vulnerabilidad pone en riesgo la integridad y la disponibilidad de los	Se recomienda no utilizar datos provenientes del usuario para crear

archivos administrados por la aplicación.	nombres de archivos. Si es necesario, se recomienda filtrar los datos de entrada, preferiblemente de acuerdo con el uso de listas blancas

Dos Null Check en IPC.

Android finaliza las aplicaciones cuando se encuentra una referencia nula. Si no se verifican las referencias nulas al recuperar datos de un Intent, un atacante podría enviar intentos con campos con valores nulos para denegar el servicio a las aplicaciones, especialmente a aquellas que tienen servicios en ejecución. En particular, se debe verificar la existencia de referencias nulas al recuperar datos de un Intent, especialmente los siguientes métodos: getAction () y getExtra ().:

Riesgo	Recomendación
Esta vulnerabilidad pone en riesgo la disponibilidad de las aplicaciones que no realizan comprobaciones para no intentar acceder a valores nulos. En caso de que los servicios ejecuten los servicios, un atacante podría denegar dichos servicios sin que el usuario lo sepa, por lo que es especialmente dañino.	Se recomienda verificar los valores nulos al recuperar datos de Intents, especialmente los siguientes métodos: getAction () y getExtra ().

Inyección de troncos

Un atacante podría incluir ciertos caracteres especiales en una cadena de texto especialmente manipulada, por lo que cuando se usa en la creación de una nueva entrada en los archivos de registro, el atacante resultará en una manipulación libre de la salida. En el caso de Android, las funciones estándar de registro de información son las siguientes:

Log.v (). Para almacenar información ampliada.

Log.d (). Con el fin de almacenar información de depuración.

Log.i (). Para almacenar información básica.

Log.w (). Con el fin de almacenar información sobre las alertas.

Log.e (). Con el fin de almacenar los registros de errores.

Riesgo	Recomendación

Los archivos de registro podrían estar dañados debido a una entrada falsa o no válida. Esto podría dar como resultado registros no confiables y la necesidad de descartar todos los registros sobre las acciones realizadas en el sistema por los usuarios. Los atacantes podrían usar esta situación para ocultar acciones criminales y evitar que se detecten sus acciones y, por lo tanto, evitar su persecución por los delitos cometidos.	Se deben tener en cuenta todos los metacaracteres utilizados para la aplicación al generar registros. Por ejemplo, si la aplicación utiliza el carácter de canalización "|" (sin las comillas) para formatear la entrada, este será un meta-carácter. Es decir, un atacante podría incluir un extra | y sería imposible reconstruir después si eso | Fue ingresado por la aplicación o por el atacante. Esto se aplica para cualquier carácter especial en los registros.

Intent inyectable

Si la entrada del usuario se carga de forma dinámica en los datos de Intención, un usuario malintencionado podría manipular dichos datos para ejecutar el código a través de ellos. En particular, se debe verificar la existencia de datos dinámicos al incluir dichos datos en un Intent, especialmente a través de los siguientes métodos de Intent: addCategory (), setAction (), setClass (), setClassName (), setComponent (), setData () andsetDataAndType ().:

Riesgo	Recomendación
esta vulnerabilidad pone en riesgo la ejecución normal de la aplicación, ya que si un usuario puede modificar los datos almacenados en los Intentos, el rendimiento normal de la aplicación también podría modificarse, así como la interacción entre las aplicaciones mediante el envío de Intenciones	Se recomienda verificar los valores de entrada del usuario, especialmente aquellos que van a Intent data instantiating la aplicación. Para tal fin, se recomienda inspeccionar los datos de usuario que terminan como parámetros en cualquiera de las siguientes funciones de Intención: addCategory (), setAction (), setClass (), setClassName (), setComponent (), setData () y setDataAndType (

Intención Spoofing

Esta categoría cubre todas las vulnerabilidades involucradas en la entrega de intenciones arbitrarias a un componente que espera recibir otro tipo de intención. De esta manera, el atacante utilizará los filtros de la víctima para enviar datos

inesperados y, por lo tanto, aprovechar estas funcionalidades. Para tener éxito con este tipo de vulnerabilidad, los componentes de destino deben ser públicos o exportables como regla general; De lo contrario, no estarán disponibles para otras aplicaciones. Para tal fin, los componentes deben tener el indicador "EXPORTADO" habilitado en el archivo AndroidManifest.xml. Tenga en cuenta que los receptores de difusión pueden declararse en AndroidManifest o en tiempo de ejecución.

Inyección de difusión

Los receptores de difusión se utilizan para registrar intenciones y responderlas. Si uno de estos componentes es público, cualquier aplicación podría enviar intenciones. El registro de Intenciones con acciones del sistema por parte de Broadcast Receivers es peligroso, ya que el hecho de registrar tales Intenciones hace que el elemento sea público y accesible, por lo que una aplicación maliciosa podría enviar Intenciones explícitas sin las acciones correspondientes porque son acciones del sistema.

Riesgo	Recomendación
En muchos casos, los receptores de difusión obtienen Intenciones y activan Actividades o Servicios con sus datos. Las vulnerabilidades de la inyección de difusión podrían permitir a un atacante manipular estos componentes para interactuar con ellos mediante la inyección de datos arbitrarios.	Se recomienda usar componentes públicos solo cuando sea necesario y, en tales casos, validar a fondo cada parámetro recuperado de la Intención. Si Broadcast Receivers registra acciones del sistema, se recomienda validar la acción del intento, ya que un intento malicioso no puede registrar las acciones del sistema, por lo que esta validación se puede utilizar para saber si el intento proviene del sistema Android o no.

Lanzamiento de actividad maliciosa

Actividades pueden iniciarse sobre la base de datos provenientes del usuario o la recepción de intenciones arbitrarias. Esto permite varias superficies de ataque: Si una Actividad utiliza datos del Intención sin verificar su origen, la integridad de los datos de la aplicación se verá afectada.

Si una Actividad entrega datos considerados como confidenciales y pueden ser lanzados públicamente, un Intento especialmente manipulado podría lanzarlos, lo que lleva a una divulgación de información confidencial.ndaciones:

Riesgo	Recomendación
La presencia de esta vulnerabilidad podría confundir a un usuario, ya que los usuarios pensarían que están usando una Actividad legítima, cuando en realidad están realizando acciones que se informan al atacante. Adicionalmente, es posible la filtración de la información utilizada por la Actividad.	Se recomienda usar componentes públicos solo cuando sea necesario y, en esos casos, validar a fondo cada parámetro recuperado de la Intención o del usuario.

Lanzamiento de servicio malicioso.

Se recomienda usar componentes públicos solo cuando sea necesario y, en esos casos, validar a fondo cada parámetro recuperado de la Intención o del usuario. Además, los servicios pueden ser protegidos estableciendo permisos.

Riesgo	Recomendación
La presencia de esta vulnerabilidad podría permitir realizar acciones arbitrarias en la aplicación mediante la ejecución de Servicios y también dar lugar a la divulgación de información utilizada por el Servicio. Como circunstancia agravante, los Servicios generalmente confían más en los datos proporcionados por los usuarios y las intenciones; además, en muchos casos proporcionan interfaces API para que las aplicaciones puedan comunicarse entre sí.	Se recomienda usar componentes públicos solo cuando sea necesario y, en esos casos, validar a fondo cada parámetro recuperado de la Intención o del usuario. Además, los servicios pueden ser protegidos estableciendo permisos.

Inseguro Control de Intención Pendiente

Una aplicación puede delegar tareas a otra aplicación a través de intenciones pendientes. La aplicación que recibe la Intención pendiente no puede modificar sus

datos, pero completa los datos vacíos. Si una aplicación genera una Intención pendiente con tal propósito pero el objetivo no está definido, una aplicación que reciba la Intención pendiente podría agregarla, por lo que dichos datos se enviarían a un componente arbitrario.

Riesgo	Recomendación
Esta vulnerabilidad podría permitir la filtración de la información que está siendo procesada por la aplicación que generó la Intención pendiente. Además, una aplicación maliciosa podría recibir, procesar y enviar dicha información, realizando un ataque Man-in-the-Middle.	Cuando las tareas se delegan a otra aplicación a través de una Intención pendiente, se recomienda establecer siempre el destino en el que se delegan las tareas para que sean ejecutadas por una aplicación de terceros.

Intención no autorizada

Esta categoría cubre todas las vulnerabilidades relacionadas con la resolución de la entrega de Intención implícita. Cuando una aplicación envía un Intent implícito, no hay garantía de que una aplicación malintencionada no recopile dicho Intent, ya que una aplicación maliciosa podría registrar un Intent Filter capaz de pasar la resolución (acción, datos y categoría), a menos que dicho Intent tenga un conjunto de permisos requeridos que el usuario malicioso no tiene. Si una aplicación malintencionada puede interceptar Intenciones implícitas, podría tener acceso a la ejecución del flujo de datos, pudiendo realizar ataques de denegación de servicio o phishing. Esta categoría considera cómo este tipo de vulnerabilidad puede ser expuesta en componentes particulares: Broadcast, Activities y Services.

Robo de transmisiones

Una intención implícita es pública cuando no está protegida con los permisos de tipo "Signature" o "SignatureOrSystem". Cuando una aplicación envía un intento público implícito, dicho componente puede leer dicho intento. Una aplicación malintencionada podría tener acceso a él, obteniendo datos confidenciales si el intento simplemente contiene filtros de intento para todas las acciones, datos y categorías. Esto es especialmente peligroso en Sticky Broadcasts, ya que el Intento

persiste en ellos y puede redirigirse repetidamente a varios destinatarios. Esto le da una mayor ventana de tiempo en la que el componente malicioso puede operar. Además, Sticky Broadcasts no puede protegerse mediante permisos.

Riesgo	Recomendación
Un atacante podría realizar un ataque de denegación de servicio en las transmisiones ordenadas, ya que un Intent solo se puede propagar sobre ellos si el primer componente que recibe el Intent para usarlo como salida. Además, podría usarse para realizar ataques Man-in-the-Middle con su posterior inyección de datos en los Intentos de propagación.	Se recomienda considerar la necesidad de enviar Intentos implícitos. Para comunicar intenciones en una aplicación, siempre se recomienda el uso de intenciones explícitas. Cuando es necesario enviar datos confidenciales en un Intento, se recomienda protegerlos configurando permisos y cifrando dichos datos.

Secuestro de actividad

Al aprovechar esta vulnerabilidad, se lanza una Actividad maliciosa en lugar de la esperada, por lo que el usuario estará en una aplicación incorrecta sin ser consciente. Esto sucede cuando el cargo de una Actividad depende de una Intención implícita. El atacante registra un Intent Filter más preciso y lo controla. Esto no siempre es posible, ya que si varias Actividades son capaces de procesar una Intención implícita, se mostrará un mensaje al usuario para que elija la aplicación. Sin embargo, esto es muy peligroso porque si esto sucede y el usuario configura como acción predeterminada la carga de la aplicación maliciosa, la Actividad maliciosa siempre se abrirá sin que se le solicite.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite ejecutar ataques de phishing, así como también fugas de la información que maneja el usuario en la Actividad involucrada. Además, esta vulnerabilidad permite al atacante modificar los datos, poniendo en riesgo su integridad.	Se recomienda considerar la necesidad de enviar Intentos implícitos. Para comunicar intenciones en una aplicación, siempre se recomienda el uso de intenciones explícitas. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos configurando permisos y cifrando dichos datos.

Servicio de secuestro

Al aprovechar esta vulnerabilidad, se lanza una Actividad maliciosa en lugar de la esperada, por lo que el usuario estará en una aplicación incorrecta sin ser consciente. Esto sucede cuando el cargo de una Actividad depende de una Intención implícita. El atacante registra un Intent Filter más preciso y lo controla. Esta vulnerabilidad es más persistente debido a que es transparente para el usuario porque los servicios no incluyen una interfaz gráfica para ella.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite la filtración de la información procesada por el Servicio involucrado. Además, esta vulnerabilidad permite que un atacante modifique los datos, poniendo en riesgo su integridad	Se recomienda considerar la necesidad de enviar Intentos implícitos. Para comunicar intenciones en una aplicación, siempre se recomienda el uso de intenciones explícitas. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos configurando permisos y cifrando dichos datos.

Debilidad intenciones especiales.

Si se envía un intento implícito para tratar con un proveedor de contenido, puede contener información confidencial en el URI, pudiendo leerse si un componente malicioso intercepta el intento. Por otro lado, un intento pendiente conserva los permisos de su creador, por lo que si un componente malintencionado intercepta un intento pendiente, podría usar los permisos de la creación del intento pendiente para suplantarlo.

Riesgo	Recomendación
La presencia de esta vulnerabilidad permite la filtración de información incluida en el URI. Además, esta vulnerabilidad permite al atacante suplantar los permisos del creador de la intención pendiente.	Se recomienda evaluar la necesidad de enviar intenciones implícitas. Para comunicar intenciones en una aplicación, se recomienda utilizar intenciones explícitas. Si es necesario enviar datos confidenciales en un Intent, se recomienda protegerlos estableciendo permisos y cifrando dichos datos.

Lógica de negocio.

En esta categoría, se incluyen las vulnerabilidades con componentes más centrados en el diseño en lugar de la codificación. Se incluyen tanto la casuística de rendimiento como la capacidad de la aplicación para trabajar de una manera inesperada que afecta su flujo de trabajo.

Los ataques dirigidos a lógicas de negocios de aplicaciones son peligrosos, difíciles de detectar y especialmente dirigidos a la aplicación.

6. CONCLUSIONES, RECOMENDACIONES

6.1. Conclusiones

Las cuestiones relativas al aseguramiento de la información son una de las cuestiones más críticas en lo que se refiere al desarrollo de aplicaciones móviles mHealth, en primer lugar, porque existen una masiva cantidad de aplicaciones huérfanas en las tiendas de Google que representan una amenaza para información de los usuarios de las aplicaciones mHealth, por tanto este tipo de aplicaciones no tiene soporte, además no existe una regulación unificada a través del cual se haga un control riguroso de las especificaciones de las pruebas no funcionales para este tipo de aplicaciones, y una de las cuestiones más cruciales es la amenaza permanente de los ciberdelincuentes que ponen en riesgo, no sólo la privacidad de la información de los pacientes, sino que pueden asimismo tener un impacto negativo en su salud. Por tanto es apremiante avanzar en el perfeccionamiento de metodologías de testing de seguridad específicas para las aplicaciones mHealth, para que esta tecnología pueda alcanzar los estándares de seguridad mínimos y generar el impacto favorable, que como vimos a lo largo del trabajo de investigación, este tipo de aplicaciones puede tener.

Uno de los elementos relevantes que se hizo evidente a través de la formulación del trabajo de investigación, es que en el ámbito latinoamericano no son muchas las investigaciones que se orientan a reflexionar sobre las aplicaciones mHealth, es de destacar que en este campo de acción específico son los europeos y los norteamericanos quienes han hecho avances más significativos con respecto a la mHealth, circunstancia que es evidentemente contraproducente, puesto que en el ámbito latinoamericano las aplicaciones mHealth podrían tener un impacto positivo en términos de aumentar la cobertura de los servicios sanitarios y mejorar las condiciones de bienestar de los pacientes de esta región del continente.

En lo que tiene que ver con las metodologías dedicadas al testing de seguridad, lo que se evidencia a través de la revisión bibliográfica realizada en este trabajo de investigación, es que no son muchas las metodologías que aborden específicamente las aplicaciones móviles, y todavía menos las que se diseñen concretamente para la plataforma Android, que es paradójicamente una de las áreas de desarrollo de software que más expansión tienen la actualidad, lo cual tiene evidentemente un impacto negativo en cuestiones referentes a la seguridad.

Con respecto a los estándares de seguridad necesarios para garantizar la fiabilidad de los datos y la funcionalidad de las aplicaciones mHealth, es crucial avanzar con paso decidido en dos sentidos fundamentalmente, por un lado las aplicaciones deben mejorar sustancialmente para garantizar el funcionamiento seguro, es decir, que se generen garantías de seguridad para que el comportamiento de las aplicaciones sería el que se espere de ellas, puesto que el buen funcionamiento de las aplicaciones, en ciertos casos depende la salud de los usuarios que hacen uso de ellas. Otra de las perspectivas en las cuales se debe incidir positivamente es en la garantía de mantener la cadena de custodia de la información privada de los pacientes, puesto que estos datos son considerados críticos.

Los esfuerzos tendientes a diseñar e implementar metodologías de testing de seguridad para aplicaciones mHealth, si bien son relevantes, todavía existen áreas en las cuales deben hacerse avances sustantivos para que estas metodologías tengan el impacto real que pueden tener, entre esas áreas se destacan, la concreción de estándares regulatorios y la incorporación de aplicaciones mHealth en las organizaciones prestadoras del servicio de salud.

6.2. Recomendaciones

Como recomendaciones y sugerencias para trabajos futuros, a través del trabajo de investigación se hizo evidente que es necesario ahondar en el campo de investigación y desarrollo de metodologías de testing de seguridad concretamente adaptadas a los requerimientos del aseguramiento de la información de las

aplicaciones mHealth, puesto que es un campo no suficientemente explorado desde el ámbito académico.

Otra recomendación que se deriva de este trabajo de investigación es que se hace necesario desarrollar aplicaciones mHealth que exploten de manera cada vez más sistemática las oportunidades que un mercado en expansión está en capacidad de ofrecer, no sólo por las oportunidades económicas que evidentemente existen en dicho mercado, sino también por los inmensos beneficios que este tipo de aplicaciones podría reportar para la salud y el bienestar de las personas en el contexto de una sociedad del conocimiento.

Otra de esas recomendaciones, es que se hace necesario ahondar en la investigación tendiente a determinar cuáles son las categorías fundamentales para el aseguramiento de la información de las aplicaciones mHealth, profundizar en este campo de investigación es válido, y sobre todo pertinente puesto que es a partir de ello que se logrará diseñar metodologías de testing de seguridad cada vez más apropiadas impertinentes.

En un sentido más general, una de las recomendaciones que podría derivarse de este trabajo de investigación, es la necesidad que existe de que las instituciones prestadoras de los servicios de salud incorporen efectivamente aplicaciones mHealth en la prestación de sus servicios sanitarios, toda vez que esta incorporación podría reportar beneficios para pacientes y personal médico, pues facilitarían y harían accesible a muchas más personas, servicios que por no hacerse de una manera más eficiente, excluyen a muchos usuarios y resultan injustificadamente onerosos.

BIBLIOGRAFÍA

Alonso-Arévalo, J., & Mirón-Canelo, J. A. (2017). Aplicaciones móviles en salud: potencial, normativa de seguridad y regulación. *Revista Cubana de Información en Ciencias de la Salud (ACIMED)*, *28*(3), pp. 1-13.

Alonso-Arévalo, J., (2016), Aplicaciones móviles en medicina y salud. *XII Jornadas APDIS*, Universidad de Salamanca. Facultad de Traducción y Documentación Grupo de Investigación E-LECTRA. Salamanca, España Recuperado de: <https://gredos.usal.es/jspui/bitstream/10366/130118/1/Aplicaciones%20m%C3%B3viles%20en%20medicina%20y%20salud.pdf>

Amaya, D., (2013). Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. *Estado actual. 12*, pp. 111-124.

Aponte, S. & Dávila, C. (2011). *Sistemas operativos móviles: Funcionalidades, efectividad y aplicaciones útiles en Colombia*. (Tesis de Pregrado) Universidad EAN, Facultad de Ingeniería de Sistemas, Bogotá

Aranaz Tudela, J. (2009). *Desarrollo de aplicaciones para dispositivos móviles sobre la plataforma Android de Google* (Tesis de Maestría). Universidad Carlos Tercero, Madrid.

Aristegui, J. (2010). Los casos de prueba en las pruebas de software, *Revista digital Lámpsakos*, pp. 27-34.

Báez, M., et al. (s.f.). *Introducción a Android*, E.M.E. editorial, Madrid. Recuperado de http://www.it-docs.net/ddata/18.pdf

Baz, A., Ferreira, I., Álvarez, M. & García, R. (s.f.). Dispositivos móviles, *Universidad de Oviedo*, Recuperado de: http://isa.uniovi.es/docencia/SIGC/pdf/telefonia_movil.pdf

Betancur, O. & Eraso S., (2015), *Seguridad en dispositivos móviles Android*. (Tesis de especialización). Universidad Nacional Abierta y a Distancia. UNAD, Bogotá.

Blanco, P., et al., (2009). Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y iphone. *Revista ibérica de sistemas y tecnología de la información.* Universidad politécnica de Madrid, Madrid: pp. 1-50.

Brick&Mobile, (2018), Estadísticas de marketing móvil. Toronto, Canada: *Brick & Mobile* <https://www.brickandmobile.com/es/mobile-stats/>

Cálad, A., Ruiz, J. (2009). *Metodologías de testing de software y su aplicación en el centro de informática de la Universidad EAFIT* (Tesis de pregrado) Universidad EAFIT, Medellín.

Cepeda, J., Meijome, X. & Santillán, A. (2012) Innovaciones en salud y tecnologías: las cosas claras. *RevistaEnfermeríaCyL*, *4*(1). pp. 28-32

Cerdeño, E. (2013). *Evolución y revolución en la telefonía*, Madrid. Recuperado de http://www.mapfre.com/mapfrere/docs/html/revistas/trebol/n65/pdf/Articulo2.pdf

Cornell, D. (2014). *Pruebas y seguridad: Elementos clave en el desarrollo de aplicaciones*. Recuperado de https://searchdatacenter.techtarget.com/es/ehandbook/Pruebas-y-seguridad-elementos-clave-del-desarrollo-de-aplicaciones.

Cristiá, M. (2009). Introducción al testing de software. *Ingeniería de Software* Facultad de Ciencias Exactas, Ingeniería y Agrimensura. Universidad Nacional del Rosario, Rosario, Argentina. Recuperado de <https://www.fceia.unr.edu.ar/ingsoft/testing-intro-a.pdf>

Cruz, B., Hernández, A., Fernández, J. & Toval, A. (2014) (2014). Seguridad y Privacidad en Carpetas Personales de Salud para Android e iOS. *RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação*, (13), pp. 35-50.

Deloitte, (2018), Consumo móvil en Colombia. Siempre conectados: ¿Bendición o maldición?. Recuperado de https://www2.deloitte.com/content/dam/Deloitte/co/Documents/technology-media-telecommunications/Consumo%20movil%202018.pdf

Domingo, M. (s.f). *Segurdiad en dispositivos móviles*. Catalunya, España Recuperado de https://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_dispositivos_moviles/Tecnologia_y_desarrollo_en_dispositivos_moviles_(Modulo_6).pdf.

Eraso, S. & Betancur, O. (2015). *Seguridad en dispositivos móviles Android* (Tesis de especialización), Universidad Nacional Abierta y a Distancia UNAD, Bogotá. Recuperado de https://stadium.unad.edu.co/preview/UNAD.php?url=/bitstream/10596/3614/1/59836994.pdf

ESET, (2018), ESET analiza el estado de la seguridad en plataformas móviles. España: *ESET* https://noticias.eset.es/eset-analiza-el-estado-de-la-seguridad-en-plataformas-moviles

Fernández, L., (2005). Un sondeo sobre la práctica actual de pruebas de software en España. *Revista española de innovación, calidad e ingeniería del software*. *1*, pp. 1-54.

Fernández, M. (2013). La salud 2.0 y la atención de la salud en la era digital. *Revista médica Risaralda*, *20*, pp. 41-46.

Franco, J. (2010). *Metodología para testing de software basado en componentes* (Tesis de pregrado) Universidad EAFIT, Medellín, Colombia. Recuperado de https://core.ac.uk/download/pdf/47237302.pdf

Galeano Marín, María Eumelia (2007). *Estrategias de la investigación social cualitativa.* Medellín, Colombia: La Carreta Editores

Gallego, A., (2014). *Desarrollo de aplicaciones para Android*. Universidad de Alicante, Recuperado de: http://www.jtech.ua.es/cursos/apuntes/moviles/daa2013/wholesite.pdf

Gavilondo, X. & Niurka, M. (2016). Salud móvil: retos y perspectivas de aplicación en Cuba. *Revista cubana de enfermería*, pp. 98-106.

Guevara, Y., (2016), Android 7.0: Un turrón maduro, *Suplementos informática* http://www.juventudrebelde.cu/index.php/suplementos/informatica/2016-11-23/android-7-0-un-turron-maduro

Guillen-Pinto, E. P., Ramírez-López, L., & Cifuentes-Sanabria, Y. P. (2017). Modelo de evaluación de requerimientos de privacidad, seguridad y calidad de servicio para aplicaciones médicas móviles. *Universidad y Salud*, *19*(2), pp. 280-292.

Hadfeg, Y., & Vega, V. (2017). Pruebas de seguridad: Estudio de herramientas. *Lámpsakos, 1*(17), pp 83-90. DOI: http://dx.doi.org/10.21501/21454086.1957

Herranz, L., 2017, *"¿Qué es eso?" Aplicación móvil para añadir nombres de accidentes geográficos a imágenes captadas desde lo alto* (Tesis de pregrado). Universidad Politécnica de Madrid, Madrid, España. Recuperado de http://oa.upm.es/44928/1/TFG_LUIS_HERRANZ_JEREZ.pdf

Herzog, P., (2010), OSSTMM 3 The Open Source Security Testing Methodology Manual Recuperado de http://www.isecom.org/mirror/OSSTMM.3.pdf#%5B%7B%22num%22%3A106%2C%22gen%22%3A0%7D%2C%7B%22name%22%3A%22XYZ%22%7D%2C28.39999%2C736.89999%2C0%5D

Herzog, P., (2017). *Open Source Security Testing Methodology Manual (OSSTMM).* Recuperado de http://www.isecom.org/research/osstmm.html

Horn, L. H. (2014). *A security framework for mobile health data collection* (Tesis de maestria) Universidade de São Paulo. Sao Pablo, Brasil.

Iglesias, D., Gómez, V., & Hernández, A. (2017). Apps y medicina intensiva. *Medicina intensiva*, *41*(4), pp. 227-236.

López, L. (2013). Generaciones de la telefonía celular. *SlideShare* https://es.slideshare.net/CPT1stAngel/generaciones-de-la-telefona-celular

Malave, K. & Beauperthuy, J (julio 2011) "Android" el sistema operativo de Google para dispositivos móviles Negotium, *7*(19), Maracaibo, Venezuela: pp. 79-96

Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC), (2018). Boletín trimestral de las TIC. Cifras del tercer trimestre de 2017. Recuperado de http://colombiatic.mintic.gov.co/602/articles-62299_archivo_pdf.pdf

OASAM, (2016), OASAM. *GitHub today*. https://github.com/b66l/OASAM

OWASP, (2018). *Mobile AppsSec Verification. Versión 1.0 (Spanish Translation).* Recuperado de https://github.com/OWASP/owasp-masvs/releases/download/1.0-ES/OWASP_Mobile_AppSec_Verification_Standard_v1.0-ES.pdf

Pérez, B. (2006). *Proceso de testing funcional independiente.* (Tesis de maestria) Universidad de la República, Montevideo, Uruguay.

PMOinformatica. (Enero 2013). Requerimientos No Funcionales: Porque son importantes. *PMOinformatica.com La oficina de proyectos de informática*. http://www.pmoinformatica.com/2013/01/requerimientos-no-funcionales-porque.html.

Ramírez, E., (2011). *Desarrollo de aplicaciones para dispositivos con sistema operativo Android*. (Tesis de pregrado). Universidad Politécnica de Valencia, Valencia, España.

Ramírez, L., Guillen, E., Cifuentes, Y. (2016). Estrategia de validación para aplicaciones móviles de salud. *Actas de ingeniería*, *1*. pp. 325-333.

Ramudo, M., (23 febrero, 2018). Condicionantes para desarrollar una aplicación de 'mHealth'. Barcelona: *Diario médico*. https://www.diariomedico.com/profesion/condicionantes-para-desarrollar-una-aplicacion-de-mhealth.html

Rodríguez, A. (2010). *Evaluación de la plataforma Android para dispositivos móviles.* (Tesis de doctorado). Universidad de Guayaquil, Ecuador.

Rodríguez, R., Fernández, J. & Tovar, A. (2013). Desarrollo de aplicaciones de salud para aplicaciones para plataformas móviles: Un mercado emergente. *Revista esalud*, *9*, pp. 2-8.

Rojas, P., (2015). Historia de las aplicaciones móviles. [Mensaje en un blog]. Creación de aplicaciones para celulares. Recuperado de http://pedromrojas12.blogspot.com.co/2015/12/historia-de-las-aplicaciones-moviles.html

Ruesca, P., (2016) Telefonía Celular – Telefonía Móvil. Zaragoza, España: *Radio Comunicaciones*. http://www.radiocomunicaciones.net/radio/telefonia-movil/

San Mauro M., González, M., & Collado, L. (2014). Aplicaciones móviles en nutrición, dietética y hábitos saludables: análisis y consecuencia de una tendencia a la alza. *Nutrición Hospitalaria*, *30*(1), pp. 15-24.

Santamaría, G., Hernández, E. (2015). Aplicaciones médicas móviles: definiciones, beneficios y riesgos. *Salud Uninorte*, *31*, pp. 599-607.

Tabares, O. (2016). *Buenas prácticas para mitigar los riesgos de malware en smartphones con sistema operativo Android* (Tesis de pregrado). Instituto Tecnológico Metropolitano, Medellín, Colombia.

The App Intelligence, (2014). Informe de las mejores 50 apps de salud en español. Observatorio Zeltia, Universidad Rey Juan Carlos. Recuperado de: www.ucci.urjc.es/wp-content/uploads/Informe-Apps-Salud.pdf

Toro, C., Vargas, J. & Hernández, M. (2015). *Guía para validar el nivel de seguridad de los permisos y uso de recursos de una aplicación móvil bajo plataformas Android* (Tesis de especialización). Universidad Católica de Colombia, Bogotá, Colombia.

Vega, R. F. (2018). mHealth: el uso de aplicaciones móviles en medicina. *Revista Electrónica AnestesiaR*, *10*(9).