Institución Universitaria

Guía metodológica para la verificación de requerimientos de seguridad informática, relacionados con la confidencialidad de la información en aplicaciones móviles del sector salud, bajo lineamientos de normativas nacionales e internacionales

Jorge Orlando Hernández Suárez

Instituto Tecnológico Metropolitano
Facultad de Ingenierías
Medellín, Colombia
2020

Guía metodológica para la verificación de requerimientos de seguridad informática, relacionados con la confidencialidad de la información en aplicaciones móviles del sector salud, bajo lineamientos de normativas nacionales e internacionales

Jorge Orlando Hernández Suárez

Tesis o trabajo de grado presentada(o) como requisito parcial para optar al título de:
Magíster en Seguridad Informática

Director(a):
Ing. Edward Paul Guillén Pinto
Ph.D. Gloria Mercedes Díaz Cabrera

Instituto Tecnológico Metropolitano
Facultad de Ingenierías
Medellín, Colombia
2020

Dedicatoria

Esta tesis es dedicada a mis padres, hermana y esposa, sin ellos no existirían los peldaños para alcanzar con agrado y emoción los objetivos en mi vida. Son mi inspiración y motivación para enfrentar cada reto y para tener la fuerza de seguir luchando cada día.

Agradecimientos

Me gustaría agradecer en estas lineas la ayuda que muchas personas y colegas me han prestado durante el proceso de investigación y redacción de este trabajo. En primer lugar, quisiera agradecer a mis padres, hermana y esposa que me han ayudado y apoyado durante este proceso, a mis asesores, Gloria Díaz y Edward Guillén, por haberme orientado en todos los momentos que necesité sus consejos, también por su gran apoyo y dedicación para que lograra finalizar esta tesis. También deseo dar un especial agradecimiento a Johany Chacón Navas, por ser ese amigo incondicional que me impulsó a iniciar esta gran aventura.

Resumen

Las aplicaciones móviles enfocadas a la salud están creciendo significativamente y algunos estudios han mostrado que las empresas no tienen la claridad de cómo lograr llevar la generalidad de las exigencias legales en la protección de datos personales a lineamientos específicos que permitan mitigar los riesgos por pérdida de confidencialidad de la información, así como tampoco se han establecido procedimientos sobre cómo validar su cumplimiento. El objetivo de este estudio fue proponer el desarrollo de una guía metodológica para la realización de pruebas especializadas de seguridad informática, que permitan verificar el cumplimiento de las exigencias legales en el ámbito de la protección de datos personales en Colombia. Para dar cumplimiento a este objetivo, este trabajo inicia con la definición de lineamientos mínimos de seguridad para el sector, a partir del análisis de la normativa nacional e internacional, y de los estándares de seguridad establecidos para aplicaciones móviles. Luego, se realiza un modelado de amenazas, a partir del análisis de la funcionalidad común de un conjunto de aplicaciones disponibles en tiendas de aplicaciones; para lo cual se empleó un modelo de amenazas híbrido, que toma como referentes las metodologías STRIDE, árbol de ataques y librería de ataques. Posteriormente, se estructura la Guía Metodológica propuesta, a partir de una definición de pruebas especializadas de seguridad informática; y por último se evalúa su aplicabilidad en un caso de prueba de una aplicación móvil del sector.

Palabras clave: Aplicaciones móviles, seguridad informática, salud asistida por el móvil, guías y metodologías.

Abstract

Mobile applications focused on health are growing significantly and some studies have shown that companies do not have the clarity of how to bring the generality of legal requirements in the protection of personal data to guidelines specific measures to mitigate the risks of loss of confidentiality of information, as well as procedures on how to validate compliance have not been established. The aim of this study was to propose the development of a methodological guide for the performance of specialized computer security tests, which would allow the verification of compliance with legal requirements in the field of personal data protection in Colombia. In order to fulfill this objective, this work begins with the definition of minimum safety guidelines for the sector, based on the analysis of national and international regulations, and established security standards for mobile applications. Then, a threat modeling is performed, based on the analysis of the common functionality of a set of applications available in application stores; for which a hybrid threat model was used, which takes as reference the STRIDE methodologies, attack tree and attack library. Subsequently, the proposed Methodological Guide is structured on the basis of a definition of specialized computer security tests; finally, its applicability is assessed in a case of testing of a mobile

application in the sector.

Keywords: Mobile Health, mHealth, security information, guidelines and methodology

Contenido

Lista de Figuras

Lista de Tablas

1. Introducción

El número de aplicaciones móviles en el mercado está creciendo exponencialmente [3], incluyendo las aplicaciones del sector salud. Estas aplicaciones procesan datos personales y confidenciales, que incluyen no sólo los antecedentes sobre el historial clínico de los pacientes, sino también información familiar, de ubicación y hasta financiera; por lo cual, el manejo de la seguridad y la privacidad es un requisito vital, más aún cuando la aplicación móvil se está utilizando en un entorno médico profesional [4]. Por lo anterior, la Superintendencia de Industria y Comercio al momento de definir las categorías especiales de datos en la ley 1581 de 2012 [5], indica en el artículo 5 descrito como datos sensibles, la importancia de proteger, entre otros, los datos relacionados con la salud y de esta manera resguardar la intimidad del titular o el uso indebido que conlleve a una posible discriminación.

Las vulnerabilidades en las aplicaciones móviles son también una realidad que aumenta el riesgo de pérdida de la confidencialidad de la información, como se puede observar en el informe presentado por NowSecure [6], que muestra pruebas de seguridad realizadas a 400.000 aplicaciones descargadas directamente en el Play Store, identificando vulnerabilidades de alto riesgo clasificadas en Sistemas de archivos, Exposición de información sensible y redes. Cabe destacar, que uno de los riesgos altos identificados por la empresa, es el relacionado con la exposición de información sensible, que afecta directamente la confidencialidad de los usuarios de las aplicaciones móviles validadas. Como conclusión del informe, Nowsecure expone puntos de vista de seguridad diferente a lo tradicional, concluyendo que la seguridad no solo debe enfocarse en identificar malware, sino también en los análisis a riesgos por fuga de información sensible, almacenada o trasmitida desde las aplicaciones móviles que afectan directamente la confidencialidad de la información, uno de los pilares de la seguridad informática.

De acuerdo con el informe presentado por HIPAA (Health Insurance Portability and Accountability Act) en Estados Unidos [7], en el sector salud, el riesgo por pérdida de la confidencialidad de los datos personales y de salud de los pacientes es considerable e incremental en el tiempo, con un registro de 270 incidentes de seguridad en 2015, 327 en 2016 y 342 en 2017. De igual manera, destaca el incremento del tipo de causa del incidente de seguridad definido como Hacking, representados en el top 20 de los incidentes de mayor registro en los últimos tres años. Para el 2015, 12 puestos del top 20 corresponden a causas del tipo Hacking, para el 2016 registró 11 puestos y en 2017 registró 17 puestos.

Con el fin de tener claridad sobre qué tipos de datos o registros son expuestos en el sector salud, el informe realizado por Verizon [8], el cual brinda una perspectiva de la realidad que vive Estados Unidos sobre la violación de registros médicos que afectan directamente la confidencialidad de la información de los pacientes.Uno de los puntos relevantes en el informe se relaciona con la forma en que categoriza los incidentes dependiendo del tipo de dato. Esta clasificación consta de Credenciales de acceso, Datos de tarjetas de crédito PCI (Payment Card Industry) , Información personal PII (Personally identifiable information) y Registros médicos. Como resultado de la investigación se observa que el mayor número de incidentes está relacionado con registros médicos.

Diferentes informes como los expuestos anteriormente, ponen en evidencia las falencias que tiene el sector salud para proteger los datos personales y de salud de sus usuarios. Una de estas falencias las describe Mirkovic, Skipenes, Christiansen y Bryhni [9], quienes expresan que la mayoría de las políticas de privacidad para los sistemas que gestionan los registros personales de salud no incluyen detalles descriptivos de la implementación de políticas y el cumplimiento de normas y estándares existentes sigue siendo muy bajo. Otro punto de vista lo presenta el Poneon Institute [10], que indican que las principales razones por las que el sector salud es objetivo de los ciberdelincuentes está en la falta de atención en cómo los socios y terceras partes protegen la información de los pacientes, también consideran que no están contratando suficientes profesionales de seguridad de TI (Tecnología de la información) con las capacidades apropiadas para identificar y mitigar los riesgos de seguridad.

Adicional a este panorama, se suma la baja percepción sobre la seguridad de la información en dispositivos móviles, como lo muestra un estudio realizado por la empresa de Checkpoint [11], el cual, a partir de una encuesta global realizada a 410 participantes que tienen responsabilidades de liderazgo en seguridad o de atención de incidentes de seguridad, representando a cada uno de los cinco continentes, estableció que el 64 % de los participantes dudan de que sus organizaciones puedan evitar un ciberataque móvil, más de 1/3 de las empresas no protegen adecuadamente los dispositivos móviles, más de la mitad cree que el riesgo de pérdida de datos es igual o mayor que los equipos de escritorio o portátiles, el 94 % espera que la frecuencia de los ataques móviles aumente y el 79 % afirma que la dificultad de asegurar dispositivos móviles está creciendo.

En Colombia, las aplicaciones móviles relacionadas al sector salud son variadas y de diferentes enfoques, en una búsqueda general en Google Play se logró encontrar aplicaciones de varias EPS (Entidades prestadoras de salud)con alcances diferentes en sus funcionalidades, IPS (Instituto Prestador de Salud), Empresas de diagnóstico y resultados médicos y algunas clasificadas como salud y bienestar, a través de las cuales se manipula y despliega información sensible como datos personales, resultados clínicos, diagnósticos, antecedentes, entre otros.

A nivel de Colombia no se encuentran estudios o datos que describan las vulnerabilidades o incidentes de seguridad en el sector salud, tampoco se cuenta con estudios que describan la relevancia que las empresas desarrolladoras de aplicaciones móviles dan a los requerimientos de seguridad de la información. Sin embargo, el riesgo de pérdida económica por multas es otra variable que afecta en este caso a las empresas custodias de la información, es así como la Superintendencia de Industria y Comercio (SIC) que dentro de sus funciones de inspección, control y vigilancia relacionada a la protección de datos personales y facultada para imponer sanciones según artículo 23 de la Ley 1581 de 2012, ha impuesto más de 610 sanciones desde el 2010 y multas que superan los $21 mil millones de pesos [12]. Uno de los casos más relevantes corresponde a la sanción millonaria aplicada a COLMEDICA en junio del 2016 por más de mil millones de pesos al no proteger adecuadamente la información de sus pacientes [13].

Las estadísticas descritas en el texto demuestran que los ciberataques se han convertido en una amenaza para el mundo de los negocios y la economía, obligando a las empresas a buscar la mejor manera de proteger sus activos de gran valor, como son los sistemas de información. La forma en que se logra dicho objetivo es identificando vulnerabilidades por medio de la validación de los sistemas de información por medio de pruebas especializadas de seguridad informática realizadas por personal ético, certificado e idóneo, con el fin de evitar la materialización de posibles riesgos de seguridad de forma proactiva. [14]. La aplicación de técnicas adecuadas de pruebas de seguridad es cada vez más importante y esencial para realizar pruebas especializadas de seguridad efectivas y eficientes [15]. Las pruebas de seguridad enfocadas a las aplicaciones móviles pueden ser ejecutadas en cualquier aplicación móvil sin importar su categoría, sin embargo, las pruebas varían dependiendo de las funcionalidades y los controles aplicados, los cuales se definen con base en los tipos de dato o información que se desea proteger. El sector salud, en comparación con otros sectores que gestionan datos sensibles, como es el caso del sector financiero, pueden gestionar algunos datos personales en común, pero a nivel funcional pueden demandar controles de seguridad completamente diferentes. Por ejemplo, en el sector financiero el contenido de un comprobante de pago considerado como sensible, es enmascarado para evitar su exposición en caso de pérdida del documento, como lo es el número de la tarjeta de crédito, débito entre otros, caso contrario sucede con el contenido de un resultado de laboratorio, el cual no puede enmascarar la información sensible por la necesidad de su lectura. Por lo anterior, los controles para mitigar el riesgo de pérdida de confidencialidad de la información son diferentes para cada caso. En consecuencia, el tratamiento del dato relacionado al sector salud, su método de consulta y de presentar la información al paciente, le agrega un componente de complejidad y de nuevos retos a quienes gestionan la seguridad informática en las organizaciones relacionadas a dicho sector [16]. También se debe tener en cuenta las exigencias legales, circulares internas o propias del sector salud que son diferentes a las planteadas para otros sectores, como el sector financiero, por ejemplo, no se contemplaría en este caso los lineamientos PCI para el

manejo de tarjetas de crédito y algunos otros lineamientos expedidos por la Superfinanciera.

1.1. Descripción del problema

El alto crecimiento de información en tránsito por parte de las aplicaciones móviles crea cuestionamientos sobre cómo las organizaciones protegen los datos personales de sus usuarios, de una posible pérdida de la confidencialidad de la información por fuga de datos generada por un incidente de seguridad. En la creciente industria de mHealth, las aplicaciones móviles del sector salud se están convirtiendo en una opción común para la prestación de servicios de salud [17]. Particularmente, estas aplicaciones están diseñadas con el fin de procesar y transmitir datos de salud que son clasificados como confidenciales. Es importante que este tipo de información se mantenga privada y segura a través de regulaciones y legislación. En consecuencia, la seguridad y la integridad de los datos asociados con estas aplicaciones es una preocupación creciente para la industria de las aplicaciones, particularmente en el dominio médico altamente regulado [16].
El problema se enfoca en cómo lograr llevar la generalidad de las exigencias legales en la protección de datos personales relacionadas al sector salud en Colombia, a unos lineamientos específicos que definirán los requerimientos de seguridad informática necesarios para mitigar los riesgos por pérdida de confidencialidad de la información y cómo pueden estos ser debidamente validados para definir la fortaleza del control implementado.

1.2. Hipótesis

La elaboración de una guía metodológica basada en pruebas de seguridad del tipo análisis dinámico, permitirá a las empresas relacionadas al sector salud y a las desarrolladoras o proveedoras de soluciones informáticas, identificar posibles riesgos de pérdida de confidencialidad de la información personal y médica en aplicaciones móviles, por medio de pruebas especializadas de seguridad informática, bajo lineamientos de normativas nacionales como internacionales relacionadas a la protección de datos personales y aplicando las mejores prácticas de seguridad.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar una guía metodológica para la verificación de requerimientos de seguridad informática por medio de pruebas dinámicas, orientadas a la seguridad informática, que permita identificar riesgos por pérdida de confidencialidad de la información personal en las

aplicaciones móviles del sector salud, bajo lineamientos de normativas nacionales e internacionales en la protección de datos personales y buenas prácticas de seguridad.

1.3.2. Objetivos Específicos

1. Establecer los requerimientos mínimos de seguridad en el tratamiento de la información en las aplicaciones móviles del sector salud, que permiten dar cumplimiento a los requisitos de ley relacionados con la protección de datos personales en Colombia.

2. Realizar un modelado de amenazas general que permita relacionar los requerimientos establecidos y los riesgos de seguridad informática asociados a cada uno de ellos.

3. Definir pruebas dinámicas especializadas en seguridad informática, que permitan verificar el cumplimiento de los requerimientos, de acuerdo con el modelado de amenazas planteado.

4. Documentar una guía metodológica que describa el proceso para la verificación de seguridad en aplicaciones móviles, haciendo uso del listado de requerimientos y pruebas técnicas desarrolladas.

5. Evaluar la implementación de la guía propuesta en una aplicación móvil relacionada al sector salud, para verificar su nivel de exposición a riesgos por perdida de confidencialidad de la información.

1.4. Estructura de la tesis

Capítulo 1: En este capítulo se presenta una introducción a los alcances y riesgos actuales de las aplicaciones móviles en el sector salud y la problemática existente de como llevar la generalidad de las exigencias legales en la protección de datos personales a lineamientos específicos para la mitigación de riesgos de seguridad. Finalmente se plantea la hipótesis y se presentan los objetivos.

Capítulo 2: En este capítulo se presentan los conceptos fundamentales relacionado al tipo de aplicaciones móviles, como son las diferentes arquitecturas existentes, los conceptos de seguridad informática relacionada a la gestión del riesgo y modelado de amenazas y pruebas técnicas de evaluación. Por último se realiza una descripción de los trabajos previos en la evaluación de seguridad de aplicaciones en el sector salud.

Capítulo 3: En este capítulo se presentan los requerimientos de seguridad en el sector salud, identificando la normatividad nacional e internacional, la descripción de los estándares de seguridad para aplicaciones móviles y por último se define los lineamientos de seguridad a

tener en cuenta.

Capítulo 4: En este capítulo se describe los diferentes modelados de amenazas actuales, el resultado de un análisis general y funcional de algunas aplicaciones móviles relacionadas al sector salud y la construcción del modelado de amenazas a utilizar.

Capítulo 5: En este capítulo se presenta las definición de las pruebas de seguridad a realizar, planteando un proceso que cubre la adecuación del ambiente, etapa de reconocimiento, pruebas de verificación de cumplimientos a los lineamientos de seguridad y el informe final.

Capítulo 6: En este capítulo se presenta las conclusiones del resultado del trabajo y las recomendaciones que pueden ser tomadas en cuenta para emprender trabajos futuros de investigación.

Capítulo 7: En este capítulo se presenta el resultado de la prueba de seguridad realizada a una aplicación móvil del sector salud, siguiendo la guía metodológica e indicando el cumplimiento o no de los lineamientos de seguridad propuestos. Los resultados se mostrarán de forma general con el fin de mantener la confidencialidad de la empresa que gestiona la aplicación móvil. Finalmente se muestran las conclusiones y recomendaciones, así como las referencias bibliográficas, anexos y demás documentos requeridos.

2. Marco Conceptual y Trabajos Previos

2.1. Aplicaciones móviles

El desarrollo de aplicaciones móviles ha crecido debido a la popularidad que existe entre los usuarios que cuentan con dispositivos móviles inteligentes [18]. Actualmente existen diferentes tiendas de aplicaciones móviles tanto gratuitas como pagas para su descarga. Entre las mas importantes se encuentran Google Play Store y Apple App Store.

2.1.1. Tipos de aplicaciones móviles

Las aplicaciones móviles son principalmente desarrolladas de tres formas, se conocen como aplicaciones nativas, web e híbridas [18].

Aplicaciones móviles nativas

las aplicaciones nativas son desarrolladas para un sistema operativo móvil especifico, por ejemplo entre las mas comunes tenemos Android y iOS. El lenguaje de programación depende de cada sistema operativo, por ejemplo el lenguaje de programación Object-C, o Swift son utilizados para sistemas operativos iOS y el lenguaje Java es utilizado en sistemas operativos Android [2].

En la tabla **2-1** se describe un comparativo entre los dos sistemas operativos mas comunes en el mercado, diferenciando las herramientas de desarrollo utilizadas, el lenguaje de programación y la extensión del binario o instalador.

Ventajas

- Aprovechamiento de todos los recursos que provee el dispositivo móvil obteniendo un mejor rendimiento.
- Permite su publicación en las respectivas tiendas de aplicaciones.

Sistema Operativo	lenguaje de Programación	Plataforma de desarrollo	Binario
iOS	Obj-C, C, C++ y Swift	Xcode	iOS App Store Package (IPA)
Android	Java, C y C++	Android SDK	Android Package Kit (APK)

Tabla 2-1.: Cuadro comparativo entre Sistemas Operativos Móviles.
Fuente: Elaboración propia.

- Proveen un mayor nivel de seguridad.
- Normalmente pueden ser ejecutadas sin requerir una conexión a internet.

Desventajas

- Las aplicaciones solo pueden ser instaladas en el sistema operativo para el cual fueron diseñadas.
- Algunas aplicaciones requieren de aprobación para ser publicadas en las respectivas tiendas.
- Algunas aplicaciones generan un costo adicional para su distribución.
- Debido a que el lenguaje de programación cuenta con cierta complejidad, los desarrollos tienden a demandar mas tiempo y costos.

En la Figura **2-1** se muestra claramente la estructura de una aplicación nativa que realiza los llamados de APIs, que acceden a las diferentes funcionalidades de los dispositivos móviles [19].

Figura 2-1.: Estructura de una aplicación nativa.
Fuente: Elaboración propia.

Aplicaciones móviles web

Las aplicaciones móivles web se ejecutan sobre servidores web, accedidos por medio de un navegador móvil, por ejemplo Safari para iOS, Opera Mobile, Skyfire, Dolphin, Firefox, Android Browser, chrome entre otros.
Son desarrolladas por medio de lenguajes utilizados para el desarrollo web como son html, css y java script. Este tipo de aplicaciones tienen como fin brindar accesibilidad a la información desde cualquier dispositivo, sin depender del sistema operativo [20].

Ventajas

- Evita el desarrollo de diferentes aplicaciones para distintos sistemas operativos.
- No requieren de aprobación por parte de fabricantes para ser publicadas.
- El costo para el desarrollo de estas aplicaciones es mas económico que las nativas.

Desventajas

- No pueden ser publicadas para su distribución o vendidas en las respectivas tiendas de aplicaciones.
- Los recursos del sistema no son accedidos en su totalidad por la aplicación ni aprovechados aprovechados de forma óptima.
- Solo funcionan si cuentan con acceso a internet.

- El nivel de seguridad es menor debido a que depende de la seguridad que proveerá el navegador.

La Figura **2-2** representa una estructura de aplicaciones web, describiendo como el navegador interactúa con los servicios prestados por el sistema operativo móvil.

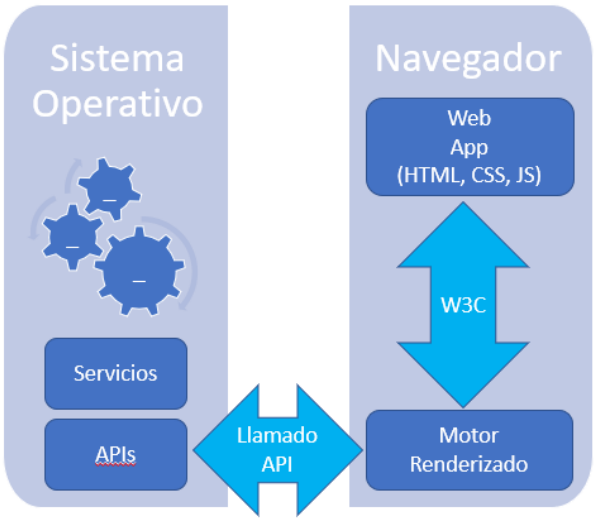

Figura 2-2.: Estructura de una aplicación móvil web.
Fuente: Elaboración propia.

Aplicaciones móviles híbridas

Este tipo de aplicaciones nacen de una combinación entre una aplicación nativa y una aplicación web. Son desarrolladas con lenguajes de programación web y un framework especializado para la creación de aplicaciones híbridas. Algunos frameworks conocidos son phonegap, titanium appacelerator, Steroids, entre otros [2].

Ventajas

- Aprovechamiento de los recursos provistos por los dispositivos móviles y los sistemas operativos.
- Los costos son menores en comparación al desarrollo de las aplicaciones nativas.
- Cuenta con el atributo de ejecución en multiplataforma.
- Permite su publicación en las respectivas tiendas de aplicaciones.

Desventajas

- Cuenta con muy poca documentación en internet.

- El nivel de seguridad es menor al aumentar la complejidad en su desarrollo y por el uso de plugins y Javascript.

En la Figura **2-3** se muestra la estructura para las aplicaciones Híbridas. Se observa la interacción entre las diferentes capas HTML+Javascript, Webview y Plugins, Porción Nativa y el Sistema Operativo [19].

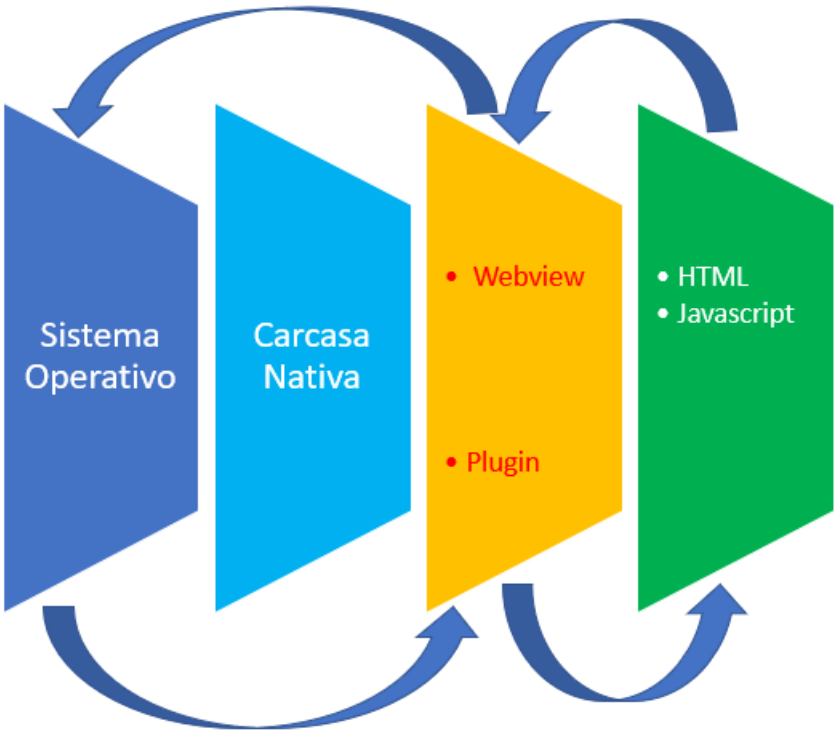

Figura 2-3.: Estructura de una aplicación móvil híbrida.
Fuente: Elaboración propia.

Aplicaciones móviles embebidas

A diferencia de las aplicaciones móviles híbridas, las aplicaciones web embebidas utiliza el componente webview para llamar un sitio web e interactuar con los servicios sin utilizar un navegador, es decir permite agregar un navegador dentro de la aplicación.

Ventajas

- Permite su publicación en las respectivas tiendas de aplicaciones.
- El costo para el desarrollo de estas aplicaciones es mas económico que las nativas.

Desventajas

- No aprovecha de forma óptima los recursos del dispositivo móvil.
- Menor rendimiento que en las aplicaciones nativas.
- Limitaciones en el funcionamiento sin internet.
- El nivel de seguridad es menor debido a que depende de la seguridad que proveerá el navegador.

En la Figura **2-4** se observa un ejemplo de las diferentes capas que forman parte en la implementación de un webview con Cordova.

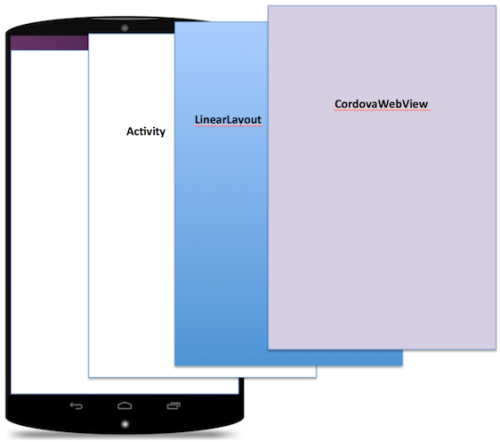

Figura 2-4.: Estructura de una aplicación móvil embebida.
Fuente: Elaboración propia.

2.1.2. Arquitecturas de aplicación

Como concepto, el estándar ANSI/IEEE1471-2000 describe el término de Arquitectura del Software como: "Arquitectura es definida por la práctica habitual como la organización fundamental de un sistema, expresada en términos de componentes, las relaciones entre ellos y el entorno, y los principios que gobiernan su diseño y evolución " [21].

La arquitectura de diseño en un sistema para aplicaciones móviles, generalmente consiste en 2 tipos de arquitectura de software, arquitectura Cliente/Servidor y punto a punto (P2P). La arquitectura de Punto a Punto se muestra en la Figura **2-5** y consiste en que todos los terminales de la aplicación ofrecen y consumen los servicios, es decir todos los nodos son de igual jerarquía [22].

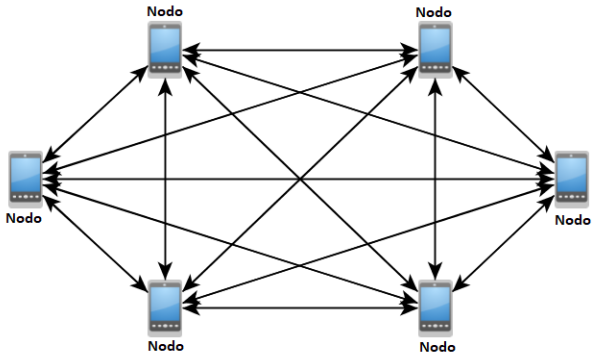

Figura 2-5.: Esquema general de una arquitectura P2P.
Fuente: Elaboración propia.

La arquitectura Cliente/Servidor se describe en la Figura **2-6**, la cual se basa en un Backend o servidor que es el encargado de ofrecer los servicios y es donde se encuentra alojado los datos, reglas y la lógica misma de la aplicación y un terminal (Clientes) donde se encuentra el usuario final, que serán los encargados de consumir dichos servicios [2].

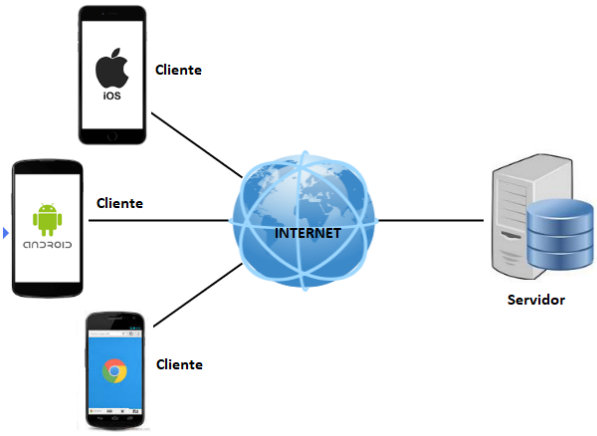

Figura 2-6.: Esquema general de una arquitectura cliente servidor.
Fuente: Elaboración propia.

El desarrollo de esta tesis se enfocará en aplicaciones móviles que operen bajo la arquitectura del tipo Cliente/Servidor, teniendo en cuenta que las aplicaciones relacionadas al sector salud que gestionan datos personales y médicos funcionan bajo este tipo de arquitectura, debido a la necesidad de acceder a la información almacenada en bases de datos que residen en servidores de la empresa prestadora del servicio. Por lo anterior, es importante

conocer que componentes forman parte de esta arquitectura.

La arquitectura Cliente/Servidor puede ser dividido en varios componentes que interactúan entre si, conformado principalmente por Clientes - Hardware, Clientes-Software y Servidores.

Clientes - Hardware

- Los clientes son los encargados de consumir los servicios ofrecidos en Internet a través de aplicaciones específicas o con el navegador.
- A nivel de Hardware se pueden identificar los siguientes clientes:
  - Equipos de Escritorio: Este tipo de equipos acceden desde redes físicas o cableadas. Equipos que cuentan con una amplia gama de funciones, configuraciones y con capacidades de personalización. Normalmente utilizado en entornos corporativos. En los hogares tienden a ser reemplazados por portátiles.
  - Equipos Portátiles: Similar a los equipos de escritorio con la diferencia de poder conectarse a los servicios a través de redes no seguras (WIFI).
  - Dispositivos Móviles: Cuenta con capacidades similares a los portátiles, pero con funciones, configuraciones y con capacidades de personalización más limitadas, por ejemplo los smartphone, dispositivos sobre los cuales se basará la presente tesis.
  - Dispositivos IoT (Internet de las Cosas): La capacidad en comunicación es más limitada. Normalmente consume los servicios a través de dispositivos de mayor capacidad como son los teléfonos móviles inteligentes (smartphone).
  - Servicio de Backend: Este tipo de servicio crea un puente entre el lado frontend de una aplicación y los servicios backend a través de una API (Application programming interface).

Clientes - Software

- Los clientes móviles consumen los diferentes servicios expuestos en la red por medio de:
  - Navegadores web: Los navegadores accede a una aplicación web utilizando el protocolo HTPP. Las aplicaciones web se crean normalmente con HTML5/CSS/JavaScript.
  - Aplicaciones específicas: Se desarrolla una aplicación por plataforma, para acceder al servicio web. La comunicación con el servidor es realizada utilizando una API del tipo HTTP por servicios REST (Representational State Transfer) o SOAP, o por protocolos específicos de la aplicación.

- Se recomienda que cada uno de los clientes consuman el servicio a través de la misma API, con el fin de evitar problemas de seguridad debido a que se desacopla la lógica de negocio de las capas de presentación existente por presentación.

Servidores

- Puede entenderse como servidor tanto el software que realiza ciertas tareas en nombre de los usuarios, como el ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer y gestionar datos de algún tipo de forma que estén disponibles para otras máquinas que se conecten a él [23].

- Los diferentes tipos de servidores existentes, dependen del tipo de aplicación, entre los principales tenemos:
  - Servidor de Base de Datos: Equipo de computo con un motor de base datos encargado de almacenar y gestionar gran cantidad de información.
  - Servidor de Aplicaciones: Ofrece una serie de servicios por medio de diferentes aplicaciones que comúnmente suelen ser del tipo web.
  - Servidor de Correo Electrónico: Como su nombre lo indica, tiene como función el envío y recepción de correos electrónicos.

Comunicaciones Cliente - Servidor

- REST

  REST, cuando se realiza un llamado a la API RESTful, el servidor transfiere la representación del estado del recurso solicitado al cliente. Fue definido por un científico informático Roy Fielding como parte de su tesis doctoral en el 2000. Este tipo de representación del estado se puede encontrar comúnmente en formato JSON o en formato XML (Extensible Markup Language) o HTML [24].

  Al llamar una API, el cliente que puede ser una persona o un software debe suministrar un identificador para el recurso de interés. Dicho recurso hace referencia a cualquier objeto donde la API puede proporcionar información y tiene un identificador único y debe proporcionar la accion que el servidor debe realizar sobre el recurso especificado, por medio de un metodo HTTP que puede ser (POST,GET,PUT, DELETE y PATH) REFERECNIA Arquitectura moviles Tesis.pdf En la Figura **2-7** se observa la arquitectura de una API REST.

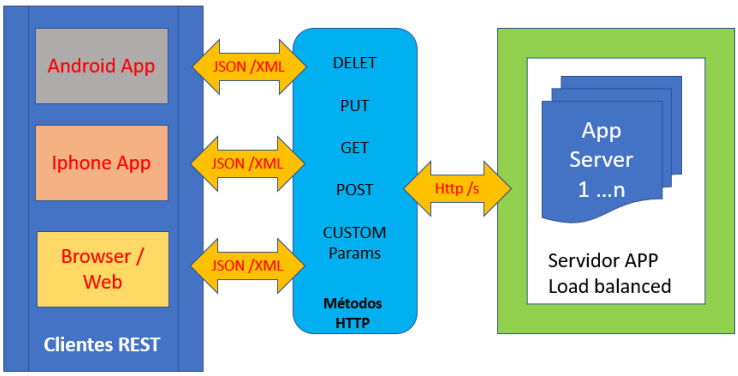

Figura 2-7.: Arquitectura API REST.
Fuente: Elaboración propia.

Una API RESTfull debe cumplir con las siguientes condiciones, Interfaz uniforme, Sin estado, Cliente/Servidor independiente, Sistema de capas y cacheable [24]:

- Interfaz uniforme: Las peticiones de clientes diferentes tienen la misma presentación, sin importar que el cliente sea un navegador web, un servidor, una aplicación móvil entre otras.
- Sin estado: Sin estado hace referencia a que el servidor no tiene memoria sobre los usuarios que consumen la API, es decir cada petición individual cuenta con toda la información requerida por el servidor para realizar una solicitud y entregar una respuesta de manera independiente a otras solicitudes efectuadas por el mismo usuario.
- Cliente/servidor independientes: La interacción entre el cliente y el servidor es basado en las peticiones iniciadas únicamente por el cliente, como consecuencia las repuestas del servidor son generadas como una reacción a las peticiones realizadas por el cliente.
- Sistema de capas: Las diferentes capas que se pueden dar en la comunicación entre el cliente y el servidor que responde la petición, como por ejemplo capa de seguridad, de almacenamiento de cache, balanceo de cargas entre otras, son completamente transparente para el cliente.
- Cacheable: En la información enviada por parte del servidor al cliente, indica si dicha información es cacheable o no, con el fin de poder almacenar los datos en la cache para diferentes fines.

- Servicios Web

Según definición dada por World Wide Web Consortium (W3C) Los servicios web

proporcionan un medio que reúne las características necesarias para interoperar entre diferentes aplicaciones que son ejecutadas en diferentes plataformas, para el intercambio de datos entre si, con el el fin de ofertar servicios y presentar la información de una forma dinámica al usuario final [25]. Como desventaja, existen algunos inconvenientes relacionados al bajo rendimiento comparado con otro tipo de tecnologías como es el caso de CORBA (Common Object Request Broker Architecture), que como factor diferencial no solo puede utilizar protocolo HTTP, también utiliza protocolos propietario.

En la tabla **2-2** se puede observar los diferentes estándares utilizados en los servicios web [2].

Estándares	Descripción
Web Services Protocol Stack	Conjunto de servicios y protocolos de los servidores web.
XML (Extensible Markup Lenguage	Es el formato estándar para los datos que se vayan a intercambiar.
SOAP (Simple Object Access Protocol)	Protocolos sobre los que se establece el intercambio. Este estándar define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.
WSDL (Web Services Description Language)	Está basado en XML y describe la forma en que se va a realizar la comunicación con los servicios web.
UDDI (Universal Description Discovery and Integration	Es el protocolo para publicar los servicios web. Su objetivo es ser accedido por los mensajes SOAP y utilizar WSDL para describir los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web.
WS-Security (Web Services Security)	Es un protocolo de seguridad. Fortalece la autenticación de los participantes en la comunicación establecida y la confidencialidad de los mensajes enviados.

Tabla 2-2.: Estándares empleados en los Servicios Web.
Fuente: Adaptado de [2]

- WebSocket

  Según definición descrita por Internet Engineering Task Force (IETF) en el RFC6455, el protocolo Websocket se diseñó con el fin de reemplazar las tecnologías que utilizan comunicación bidireccional que utilizan HTTP como capa de transporte y lograr de esta forma aprovechar la infraestructura existente, como por ejemplo autenticación, proxies, filtrado entre otros.El protocolo websocket se creó para funcionar a través de los puertos HTTP 80 y 443 y de igual manera admite proxies e intermediarios HTTP [26].

  El World Wide Web Consortium (W3C) está normalizando la API de Websocket. Es una tecnología que crece junto con HTML5. Los navegadores que soportan este tipo de protocolo en sistemas operativos móviles son Android, iOS, Windows Phone, Firefox OS y Blackberry OS.

  La Figura **2-8** se describe como el cliente realiza la primera petición (Handshake) al servidor que se encuentra a la escucha. El servidor responde al cliente con la confirmación (Acknowledgement) y se procede al intercambio de mensajes necesarios. Por último de realiza el cierre de la conexión [26].

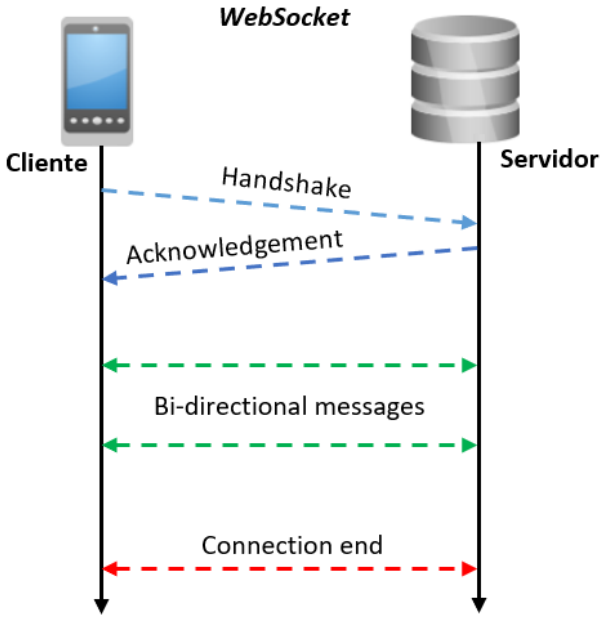

Figura 2-8.: Comunicación WebSocket entre Cliente y Servidor.
Fuente: Elaboración propia.

2.1.3. Seguridad y los dispositivos móviles inteligentes

Los sistemas operativos móviles actuales tienden a ser mas seguros que los sistemas operativos de escritorio convencional, pero los problemas de seguridad no son ajenos a este tipo de tecnologías, los nuevos retos de seguridad durante el desarrollo de aplicaciones móviles debe considerar diversos temas relacionados al almacenamiento de datos, es uso de las API criptográficas, las comunicaciones entre aplicaciones, comunicaciones seguras de red entre otras [27].

Una aplicación móvil puede ser afectada durante su desarrollo a causa de vulnerabilidades y malas prácticas de seguridad que son generadas al no identificar en la fase de análisis la definición de requisitos de seguridad, la falta de participación del equipo de seguridad en los diseños de arquitectura, liberando las aplicaciones de forma inadecuada o al no responder oportunamente los incidentes de seguridad. Estos problemas también pueden afectar otras aplicaciones que se ejecutan bajo un entorno compartido, otros sistemas que interactúan con la aplicación o hasta el mismo sistema operativo.

Conceptos importantes en el desarrollo seguro móvil

El entorno móvil comparte características con las aplicaciones web y con los sistemas de escritorio en cuanto al volumen de usuarios, conectividad a la red, almacenamiento compartido de datos, existencia de aplicaciones vulnerables, malware entre otras [28]. La superficie de ataque en los dispositivos móviles se detalla en la Figura **2-9**.

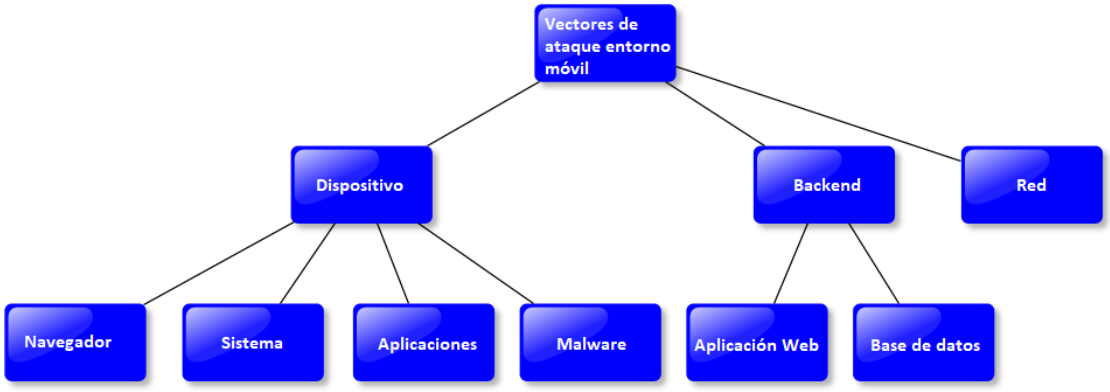

Figura 2-9.: Superficie de ataque en un dispositivo móvil.
Fuente: Elaboración propia.

En un contexto general, las buenas practicas se aplican en el desarrollo seguro de las aplicaciones, principalmente en los siguientes puntos:

- Validación de entrada:
  Toda entrada al sistema debe considerarse potencialmente peligrosa, como por ejemplo los campos de texto, urls, cookies y otra serie de campos HTTP que son susceptibles a inyecciones de caracteres especiales potencialmente peligrosos, ver Figura **2-10**. Estas entradas deben protegerse por medio del uso de listas blancas, que tienen como fin validar que los valores ingresados en una entrada son permitidos bajo la lógica de negocio definida y acorde al tipo de dato que se espera.

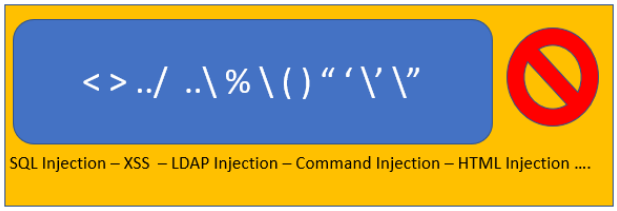

Figura 2-10.: Caracteres potencialmente peligrosos.
Fuente: Elaboración propia.

  Otro mecanismo está relacionada a la codificación de los datos. La codificación debe ser aplicada dependiendo del uso que se le dará a los datos, por ejemplo si el interprete que recibirá los datos es web, se debe controlar la generación de elementos interpretables en CSS, HTML, javascript entre otros. Si el dato de salida es interpretado por otros sistema, se debe controlar la creación de ciertos comandos que pueden generar una inyección del tipo SQL, LDAP, XML entre otros.

- Autenticación
  Por definición, la autenticación es el proceso que permite validar que un individuo, sitio web o entidad es quién dice ser, cuando intentar acceder a una aplicación o consumir un servicio en específico [29].

  La autenticación en la mayoría de los casos consiste en el envío de un nombre de usuario o ID y uno o mas datos de conocimiento propio del usuario, que se determina basado en el grado de fortaleza deseada en la autenticación. Esta fortaleza depende directamente

del tipo factor o factores de autenticación utilizados y se definen según las siguientes características [29]:

- Algo que conozco: contraseñas o detalles de la cuenta.
- Algo que tengo: Token físico, Token enviado por email o Softoken en celulares.
- Algo que soy: Factores biométricos (Huella, Rostro, Iris entre otros).

- Gestión de sesiones:
  OWASP (Open Web Application Security Project) define a la sesión como "una serie secuencial de solicitudes de red HTTP y respuestas de transacción relacionadas a un mismo usuario" [30]. En otras palabras, la sesión permite al servidor identificar a un usuario durante las diferentes peticiones o transacciones realizadas sin la necesidad de enviar en todo momento las credenciales de autenticación. Estas capacidades provistas por la sesión se pueden ofrecer antes y después de la autenticación según se defina en su diseño. Adicionalmente permite aplicar un conjunto de controles que ayudarán a fortalecer la seguridad en el acceso a información que es exclusiva del usuario.

- Control de acceso:
  Su finalidad principal es evitar que los usuarios actúen fuera de los permisos previamente otorgados, como por ejemplo acceder a información a la cual no está autorizado o acceder a funcionalidades sin contar con los privilegios para ello [31]. Las vulnerabilidades asociadas al control de acceso son muy comunes, por esta razón OWASP la incluye en el Top 5 de riesgos web como Pérdida de Control de Acceso y en el Top 6 de riesgo móvil como Autorización Insegura. Este tipo de riesgos suelen ser generados por la falta de detección en pruebas automáticas y funcionales efectivas al momento de su desarrollo y pruebas de seguridad previas a su salida a producción.

- Gestión de errores:
  Según la Real Academia Española, un error, entre otras definiciones, corresponde a la "Acción desacertada equivocada" [32]. En el área del software se conoce como bug, un error de software que se comporta de una manera inesperada generando un resultado indeseado. El escenario de errores puede ir a nivel de la aplicación móvil como de los servicios y si estos errores no se gestionan correctamente, se puede llegar a exponer información privada que sería muy útil para un atacante en su etapa de reconocimiento del objetivo.

- Protección de datos:
  La protección de datos se basa en tres elementos fundamentales que son:
  - Confidencialidad: Los datos deben estar protegidos de acceso no autorizado o la divulgación cuando están almacenadas (Bases de datos) o en tránsito.

- Disponibilidad: Los datos deben estar accesibles para usuarios autorizados cuando sean requeridos.
- Integridad: Los datos deben estar protegidos de modificaciones no autorizadas cuando están almacenadas (Bases de datos) o en tránsito.

El sistema debe estar en la capacidad de proteger los datos en sus diferentes estados, en Reposo, Tránsito y Memoria. Un ejemplo de estado en reposo es el almacenamiento persiste del dispositivo como tarjeta SD o en el sistema interno de archivos. Un ejemplo de estado en tránsito es la data enviada o recibida desde el servidor. Un ejemplo de estado en memoria hace referencia a los datos sobre los que se ejecutan cierto tipo de operaciones y requieren estar almacenadas en la memoria del dispositivo móvil. Lo anterior es con el fin de evitar que sean ilícitamente obtenidos, divulgados o alterados [33].

Un termino importante está relacionado con la ofuscación del código, que "tiene como función principal reducir la legibilidad e interpretación de funciones y procesos que realizan las aplicaciones, mitigando de esta forma los riesgos que genera los ataques de ingeniería inversa del código fuente como por ejemplo la obtención de información importante relacionada a la lógica de aplicación" [34].

Los datos a proteger pueden ser Las Credenciales de Acceso, Tokens, Variables de Sesión, Información Personal, Información Financiera, Registros Médicos, Código Fuente entre otros.

- Seguridad en las comunicaciones de aplicaciones móviles:
  En las comunicaciones móviles podemos encontrar tecnologías para la comunicación de datos como 3G, 4G(LTE), WIFI entre otras. La seguridad de las comunicaciones y la privacidad de sus datos no fueron premisas en sus diseños, permitiendo que una serie de amenazas atente contra la confidencialidad de la información como es la Pérdida o robo del dispositivo móvil, el robo de información, suplantación entre otros [35].

  Por lo anterior es importante implementar la protección de capa de transporte para las aplicaciones móviles, con el fin de proteger los datos cuando se transmiten. Comúnmente las aplicaciones utilizan el protocolo web Hypertext Transfer Protocol (HTTP) para establecer las comunicaciones con el servidor o servidores, pero también cuenta con una versión segura conocida como Hypertext Transfer Protocol Secure (HTTPS) que utiliza principalmente HTTP sobre SSL (Secure Socket Layer) o TLS (Transpor Layer Security). Este tipo de algoritmos de cifrado suele sufrir de diversas vulnerabilidades que puede exponer información sensible, por ejemplo las versiones SSLv2, SSLv3 y TLSv1.0 no se recomiendan debido a diferentes ataques conocidos y documentados co-

mo son Drown, Crime, Beast, Breach, Sweet32, SSL Poodle, Heartbleed entre otros. El otro vector que degrada la seguridad en este tipo de protocolos son los Cipher Suites, que son algoritmos de cifrado que deben contar con longitudes de claves robustas. Algunos Cipher Suites son débiles o inseguros como RC2, RC4 , DES, 3DES, Nul, MD5, CBC, entre otros, que no deben ser utilizados [36].

2.2. Gestión del Riesgo y Modelado de Amenazas

Teniendo en cuenta la necesidad de proteger la información, las organizaciones dentro del Sistema de Gestión de Seguridad de la información, es importante realizar una correcta gestión de riesgos con el fin de identificar potenciales vulnerabilidades de los activos de información y que tipo de amenazas podrían llegar a explotar dichas vulnerabilidades. Con la identificación de riesgos se podrá establecer medidas preventivas que ayuden a mejorar el nivel de seguridad de la información [37]. Para complementar el entendimiento del riesgo en seguridad de la información, se tienen las siguientes definiciones.

2.2.1. Gestión del Riesgo y sus definiciones

Con base a la norma técnica NTC-ISO/IEC 27005 en su numeral 3 Términos y Definiciones:

- "Riesgo en la seguridad de la información: Potencial de que una amenaza determinada explote las vulnerabilidades de los activos o grupos de activos causando así daño a la organización" [38].
- "Identificación del riesgo: Proceso para encontrar, enumerar y caracterizar los elementos de riesgo" [38].
- "Reducción o mitigación del riesgo: Acciones que se toman para disminuir la probabilidad las consecuencias negativas, o ambas, asociadas con un riesgo" [38].

Con base a la norma técnica NTC-ISO/IEC 27000 en su numeral 2 Términos y Definiciones:

- Análisis de riesgo: "Proceso para comprender la naturaleza del riesgo y para determinar el nivel de riesgo" [39].
- Evaluación del riesgo: "Proceso de comparar los resultados del análisis de riesgo con los criterios de riesgo para determinar. si el riesgo y / o su magnitud es aceptable o tolerable" [39].

- Gestión del riesgo: "Aplicación sistemática de políticas, procedimientos y prácticas de gestión a las actividades de comunicación, consultoría, establecimiento del contexto e identificación, análisis, evaluación, tratamiento, seguimiento y revisión del riesgo" [39].

El riesgo a tratar en la tesis está directamente relacionado con la Confidencialidad de la información.La confidencialidad de la información es uno de los pilares de la seguridad de la información y es muy relevante en el desarrollo del presente documento de tesis a la necesidad de proteger los datos de los pacientes como es la información personal y de salud utilizada en las aplicaciones móviles, de individuos no autorizados.

Con base a la norma técnica NTC-ISO/IEC 27001 en su numeral 3 Términos y Definiciones: "Confidencialidad es la propiedad que determina que la información no esté disponible ni sea revelada a individuos, entidades o procesos no autorizados" [40].

2.2.2. Tipos de Modelado de Amenazas

Con el fin de identificar a que riesgos puede estar expuesta la información en términos de confidencialidad y poder definir los lineamientos de seguridad a cumplir para su mitigación, es importante adoptar un modelado de amenazas. Este modelado es un medio de aproximación al análisis de la seguridad en las aplicaciones. Este proceso estructurado facilita la identificación, cuantificación y direccionamiento de los riesgos de seguridad asociados a una aplicación. El modelado se convierte en una herramienta para un proceso posterior en el ciclo de vida de seguridad de la aplicación que es la revisión de código.

De una forma general se pueden identificar tres tipos de técnicas para realizar un modelado de amenazas. La primera es conocida como STRIDE que significa Spoofing (Suplantación), Tampering (Manipulación), Repudation (Repudio), Information Disclure (Revelación de información), Denial of Service (Denegación de servicio) y Elevation of Privilege (Elevación de privilegios).
La segunda corresponde a los Arboles de Amenaza, que es una representación conceptual de diversos ataques contra una aplicación en particular, que ayuda a identificar las posibles amenazas.
La tercera es Bibliotecas de Ataques y hace referencia a un conjunto de ataques que ayudan a encontrar las amenazas contra una aplicación que se encuentra en desarrollo. En esta técnica los ataques deben ser bien detallados [1].
Durante el desarrollo de la presente tesis, se utilizará un modelado de amenazas Híbrido, es decir que adopta algunos conceptos bajo las definiciones de los tres modelos descritos anteriormente.

2.3. Técnicas de evaluación y verificación de requisitos de seguridad

Con el modelado de amenazas y los lineamientos de seguridad debidamente definidos, el siguiente paso consiste en verificar que dichos lineamientos de seguridad se están aplicando de forma correcta y que los diferentes riesgos de seguridad se están mitigando en unos niveles aceptables. La forma de realizar este tipo de verificaciones es por medio de pruebas especializadas de seguridad, test de intrusión o Pentest. Según definición dada por Pablo González Pérez, "Un test de intrusión es un método que evalúa el nivel de seguridad de una red de equipos o sistemas informáticos" [41]. El test de intrusión implica un análisis activo sobre cada uno de los sistemas de información, con el fin de encontrar indicios de posibles vulnerabilidades de cualquier tipo. Este análisis es llevado a cabo desde la posición de un atacante, el cual podría realizar la explotación de las vulnerabilidades de seguridad encontradas.

2.3.1. Tipos de pruebas de seguridad informática

En términos de accesibilidad, las pruebas de seguridad se clasifican validaciones de seguridad tipo caja negra, tipo caja blanca y tipo caja gris [42]:

- Validaciones tipo caja negra: Estas pruebas adquieren el rol de un Hacker debido a que no conoce la arquitectura, códigos fuentes y demás atributos de una aplicación. Tiene los alcances de un usuario normal del servicio.
- Validaciones tipo caja blanca: Estas pruebas adquieren el rol de un usuario interno, que cuenta con la documentación de la arquitectura de la solución y de los códigos fuentes entre otro tipo de información exclusiva de la organización.
- Validaciones tipo caja gris: Estas pruebas son una fusión entre las pruebas de caja negra y caja blanca, debido a que cuenta con cierta información pero no con todo el contexto que tiene las pruebas de caja blanca.

Durante el desarrollo de la tesis, el enfoque de las validaciones de seguridad serán del tipo caja negra.

2.3.2. Técnicas utilizadas en pruebas de seguridad informática

Para el desarrollo de las pruebas de seguridad se cuenta con diferentes técnicas identificadas principalmente como pruebas basadas en modelos de requisitos y modelos de diseño, en

análisis estático en los códigos fuentes de la aplicación, en análisis dinámico en sistemas en ejecución [41].

- Pruebas basadas en modelos de seguridad: Las pruebas basadas en modelos MBT (Model-based testing), es un área activa de investigación y se basa en la selección de algoritmos para genera de forma automática casos de pruebas a partir de modelos de sistemas bajo prueba. Como valor agregado incluye una revisión temprana del comportamiento del sistema y tiene la capacidad de crear automáticamente pruebas útiles entre otras. Un Modelo basado en pruebas de seguridad MBST (model-based security testing.) se basa en el enfoque de un MBT que se encarga de validar los requisitos del software relacionado a la seguridad. Combina las características de seguridad como integridad, confidencialidad, disponibilidad, autorización, autenticación y no repudio con un modelo de un Sistema Bajo Prueba (SUT).

- Pruebas análisis estático: Muchas vulnerabilidades son detectadas al revisar el código fuente de las aplicaciones, razón por la que adquiere un gran valor al detectar vulnerabilidades en etapas tempranas del ciclo de vida del desarrollo seguro del software. Este tipo de pruebas son completamente estáticas y no requiere de pruebas en ejecución. Estas validaciones implica revisión del código fuente de las aplicaciones o del binario. Por exactitud en los resultados de las pruebas, se recomienda que las validaciones se realicen en el código fuente y no en el binario y además permite proporcionar recomendaciones de solución con mejor detalle. Las validaciones de código se pueden realizar de forma manual o automática y son ideales para el tipo de pruebas de caja blanca.

- Pruebas análisis dinámico: Estas validaciones de seguridad son ideales para el tipo de pruebas de caja negra, teniendo en cuenta que no requiere de acceso al código fuente debido a que la interacción es directamente con el software en ejecución, es decir se analiza cada uno de los flujos ejecutados por la aplicación con el servidor en las diferentes funcionalidades que provee, incluyendo las operaciones que realiza la aplicación con el sistema operativo móvil. Este tipo de pruebas son realizadas desde el exterior en un entorno que es comparable a un ataque real de un tercero malicioso. Generalmente las pruebas de seguridad se realizan tanto de forma forma manual como de forma automática.

Durante el desarrollo de la tesis, el enfoque de las validaciones de seguridad se realizará bajo la técnica de análisis dinámico.

2.3.3. Relevancia de las pruebas de seguridad

Los test de intrusión son valiosos y de necesidad en un momento empresarial por las siguientes razones [41]:

- Identificar vulnerabilidades críticas, las cuales son el resultado de la utilización de vulnerabilidades de menor riesgo.
- Identificar vulnerabilidades que pueden resultar difíciles o prácticamente imposibles de detectar con escáneres de vulnerabilidades, los cuales automatizan el proceso.
- Testear los controles de seguridad para verificar su comportamiento ante los ataques y como responden a éstos.

Teniendo en cuenta que las pruebas especializadas de seguridad son del tipo caja negra y bajo un escenario muy similar al que utilizaría normalmente un atacante, es necesario entender el término Hacking.Hacking ético, tal y como lo menciona Pablo González Pérez [42], quien lo define como "una formación profesional en el área de la seguridad informática, que tiene como fin evaluar el grado de vulnerabilidad y el nivel de exposición al riesgo en el que se encuentran los sistemas de información de una organización, basado en unas condiciones previamente acordadas con el cliente". El término Hacker ético fue impuesto por la sociedad para lograr un diferenciamiento entre el comportamiento ético de un profesional, de otros individuos que actúan de forma no autorizada y con acciones mal intencionadas.

2.4. Seguridad en aplicaciones en salud

Las tecnologías de salud móvil, también conocidas como tecnologías mHealth, han surgido, entre los proveedores de servicios de salud, como una de las tecnologías de elección para el siglo XXI al proporcionar no solo un cambio transformador en la prestación de servicios de salud, sino también información crítica de salud que forman parte de sistemas integrados de información sanitaria. Las tecnologías de mHealth fomentan plataformas integrales y herramientas pragmáticas para administrar la información de salud en el continuo tránsito de información entre diferentes proveedores de atención médica. Las tecnologías de mHealth comúnmente utilizan dispositivos médicos móviles, dispositivos inalámbricos y de monitoreo, y / o telemedicina en la entrega de atención médica y la investigación en salud. En la actualidad, las tecnologías mHealth brindan oportunidades para registrar y monitorear las condiciones de los pacientes con enfermedades crónicas como el asma, las enfermedades pulmonares obstructivas crónicas (EPOC) y la diabetes mellitus [43].

2.4.1. Registros

El adquirir estas nuevas tecnologías como mhealth, nos lleva a un nuevo cambio en la forma que gestionamos los datos de los pacientes. El registro de tareas relacionadas al cuidado de la salud de los pacientes se ha realizado históricamente en medio físico (papel). Este tipo de registros lleva consigo una serie de desventajas como es la accesibilidad, legibilidad, recuperación de la información y almacenamiento. Con la evolución de los sistemas informáticos, se implementó una nueva forma de registro para los datos médicos. Con este nuevo formato se logra reducir en gran parte las desventajas del registro en papel. Por lo anterior, la migración de los registros médicos a un formato electrónico es una de las actividades principales que adelantan hoy en día las instituciones de salud a nivel mundial [44].

Algunas definiciones que permiten entender el tipo de registros médicos en formato electrónico que pueden ser susceptibles a sufrir pérdida de confidencialidad de la información son:

- Dato personal: Definido en la ley 1581 en Colombia como la información que tiene la capacidad de relacionarse o vincular a una o varias personas en particular [5].
- Registro Personal de Salud (PHR), es un registro de salud que contiene datos de salud e información relacionada con la atención y es administrado por el paciente. PHR brinda a los pacientes, opciones para acceder, administrar y compartir información sobre su salud [45].
- Información de Identificación Personal (PII): es uno de los conceptos más centrales en la regulación de la privacidad. Define el alcance y los límites de una amplia gama de estatutos y reglamentos de privacidad. la regulación de privacidad se centra en la recopilación, el uso y la divulgación de PII [46].
- Registro Electrónico de Salud (EHR): es un repositorio digital que contiene información sobre la salud de las personas. Esta información se recopila y gestiona en Sistemas EHR [47].

2.4.2. Normatividad

La Real Academia Española define Normativa como un "Conjunto de normas aplicables a una determinada materia o actividad" [32]. En el caso particular del sector salud, se cuenta con una serie de normativas plasmadas en la Constitución Política, Leyes, Decretos, Resoluciones, Circulares Externas, Circulares Conjuntas, Sentencias, Conceptos entre otras, que son inspeccionadas, controladas y vigiladas por la Superintendencia Nacional de Salud.

Durante el desarrollo de la tesis, el enfoque de las diferentes normativas en Colombia y las implementadas en otros países, estará relacionadas a las exigencias de seguridad informática que aplican al sector salud. Por Ejemplo:

- En Colombia se identifica principalmente la Ley 1581 del 2012 relacionada a la protección de datos personales, que no son exclusivas del sector salud. Su cumplimiento y vigilancia está a cargo de la SIC.
- En Estado Unidos se destaca HIPAA que entre otras disposiciones establece los requerimientos de protección y manejo confidencial de información de salud considerada sensible, esta normativa es exclusiva para el sector salud.
- En Europa, se aprobó el GDPR (General Data Protection Regulation) que corresponde a un conjunto nuevas normativas creadas para otorgar a los ciudadanos de la Unión Europea un mayor control sobre sus datos personales, no son exclusivas del sector salud.

2.5. Trabajos previos en evaluación de seguridad de aplicaciones en salud

Al revisar la literatura relacionada a guías de verificación de requisitos de seguridad informática enfocadas a la protección de la confidencialidad de los datos personales y de salud, contemplando el cumplimiento de leyes de protección de datos, se encontró un primer acercamiento con el trabajo propuesto por Mirkovic, Skipenes, Christiansen, y Bryhni, quienes describen una lista de lineamientos de seguridad y privacidad para proteger los registros personales de salud (PHR) en Noruega [48], tomando como referencias las reglamentaciones existentes en dicho país, las definidas por la Comunidad Económica Europea y las que desempeñan un papel importante en la protección de los PHR e incluso los registros electrónicos en salud (EHR) en general, con el fin de cumplir con requisitos legales, reglamentarios y de seguridad. Dentro de su objetivo se encuentra resumir y describir una lista de directrices que puede ser utilizadas durante la planificación e implantación de sistemas PHR en Europa. Esta guía contempla un entorno basado en sistemas distribuidos de tal forma que los pacientes interactúen con el sistema y gestione su información. Como resultado del estudio el autor propone 8 lineamientos que son generales, basados en estándares como la ISO 27005 y 27799. Esta guía, sin embargo, se orienta a los PHR y no hace referencia a las aplicaciones y mucho menos a las arquitecturas para soluciones móviles; Los lineamientos propuestos son muy generales, no se encuentra evidencia que el planteamiento fuera sometido a pruebas especializadas de seguridad en un ambiente específico donde se gestiona este tipo de PHR.

La guía propuesta por Ayubi, Pelletier, Sunthara, Gujral, Mittal y Bourgeois, es descriptiva y se enfoca en el desarrollo de aplicaciones móviles para hospitales basado en políticas de seguridad del tipo BYOD (Bring Your Own Device) [49]. Las directrices van orientas a la protección de la aplicación y los datos, bajo el entorno de un dispositivo móvil, las cuáles fueron agrupadas en 4 categorías (1) autenticación y autorización, (2) gestión de datos, (3) protección del entorno de la aplicación y (4) aplicación remota. El alcance definido por el autor no contempla la mejora en el cumplimiento de leyes para proteger la información personal y registros médicos de los usuarios, solo menciona las recomendaciones especificadas por HIPAA y las directrices van más alienadas al modelo BYOD. Esta guía fue sometida a una evaluación en un entorno cerrado sin interacción de los pacientes, solo para personal interno del hospital mediante el uso de una aplicación móvil específica para el manejo de los registros médicos . No se observa la realización de pruebas de seguridad especializadas a las funcionalidades de la aplicación móvil, necesarias para validar la correcta implementación de los controles. Los lineamientos planteados se enfocan principalmente en el uso del dispositivo móvil.

Al indagar en trabajos un poco más técnicos, los autores Dorazio y Choo plantearon un proceso genérico para identificar vulnerabilidades de seguridad y debilidades en el diseño de aplicaciones del sector salud en sistemas operativos iOS [50]. El proceso fue validado en una aplicación ampliamente utilizada en Australia en el sector salud identificando datos confidenciales del usuario y personales PII almacenadas en el dispositivo. En el artículo se observa que el enfoque es más a nivel de interacción de la aplicación con el sistema operativo que, en la funcionalidad de la misma, basado en las cuatro etapas planteadas por el autor, (1) análisis de configuración del entorno, (2) Descifrado de la aplicación, (3) Des ensamblaje de la aplicación, y (4) Depuración de las aplicaciones. La aplicación propuesta para validar los resultados del proceso planteado fue sometida a pruebas de seguridad especializadas aplicadas y limitadas al alcance. El autor tampoco contempla la mejora en el cumplimiento de leyes para proteger la información personal y registros médicos de los usuarios.

Uno de los trabajos que más se acerca a la propuesta planteada en esta tesis corresponde a Martinez, Soto, Eraso, y Ordóñez, quienes proponen una guía para administrar y custodiar los Registros de Salud EHR en Colombia [51], tomando como referencia un análisis de las regulaciones de seguridad para la protección de los registros médicos en Colombia como la Resolución 3374 del 2000, Ley 1581 de protección de datos personales, ley 1438 de 2011 y la Resolución 1995 of 1999. Estas reglamentaciones son soportadas con regulaciones y estándares internacionales como HIPAA y La directiva de protección de datos europea 95/46/EC. Este trabajo se enfoca en lineamientos generales de Control de Acceso, Seguridad Basada en Roles y Privacidad. Esta guía fue sometida a una evaluación en un entorno cerrado sin in-

teracción de los pacientes, solo para personal interno del hospital aplicada a dos plataformas OpenMRS y Management Dynamics. Estas validaciones no se realizaron sometiendo a las aplicaciones a pruebas especializadas de seguridad informática, no contempla arquitecturas móviles dentro de su planteamiento, ni el vector de riesgo que puede generar una aplicación móvil debido al enfoque general que presenta, incluso propone como trabajo futuro realizar una verificación más detallada de la implementación de las directrices propuestas.

Como se puede apreciar en las aproximaciones, no se cuenta con guías integrales que planteen lineamientos de seguridad más específicos y dirigido a las aplicaciones móviles y su arquitectura en pro de la protección de la confidencialidad de la información, en pocos casos se contempla el uso de las soluciones móviles de salud directamente por pacientes y que estén alineadas a leyes y directrices en Colombia apoyada con estándares y leyes internacionales. También carecen de pruebas especializadas de seguridad informática para verificar la eficacia de los controles de seguridad aplicados, basado en los lineamientos planteados. Por esta razón se desea continuar con el trabajo futuro planteado por Martinez, Soto, Eraso y Ordóñez quienes describen la necesidad de realizar una verificación más detallada de la implementación de las directrices planteadas, agregando a este trabajo un enfoque hacia arquitecturas móviles y el cómo realizar las pruebas de seguridad especializadas [51].

Para la realización de las pruebas especializadas de seguridad, se tomará como referencia la técnica definida como pruebas de penetración y análisis dinámico, establecida por Felderer Buchler [52]. Gracias a este estudio donde se define diferentes tipos de pruebas y técnicas de seguridad, se destaca las pruebas de penetración y análisis dinámico a las aplicaciones móviles, la cual está orientada a pruebas del tipo caja negra, es decir sin conocimiento previo de su código fuente o la arquitectura propia de la aplicación, debido a que se basan solo en el comportamiento de entrada y salida del software. Este escenario es al que se enfrenta normalmente un ciber delincuente cuando decide buscar vulnerabilidades en una aplicación para explotarlas a su beneficio. Esta técnica ayudará a dar un panorama real de los riesgos a los que puede estar expuesto una aplicación móvil relacionada al sector salud y que están publicadas en las respectivas tiendas de Google y Apple para descarga y uso de los pacientes. En el documento no se observó una estrategia basada en un modelado de amenazas, que permita proponer una serie de pruebas de seguridad que contemple las diferentes definiciones de pruebas dinámicas. El contexto utilizado por el autor se limita a un ámbito general de diferentes pruebas de seguridad y no a la ejecución de las mismas en un entorno móvil.

3. Requerimientos de seguridad en el sector salud

En Colombia no se cuenta con estudios o datos estadísticos con accesibilidad al público, que describan las vulnerabilidades o incidentes de seguridad que ocurren en el sector salud, que permitan definir si cumplen con algún tipo de requisitos de seguridad que ayuden a proteger la confidencialidad de la información de los pacientes, especialmente en el entorno de aplicaciones móviles. Por lo anterior se recurre a estudios o reportes realizados a nivel internacional o a las sentencias otorgadas por la Superintendencia de Industria y Comercio relacionado a la violación de la protección de datos personales, con el fin de obtener mayor detalle de los incidentes y el grado de exposición a riesgos por perdida de confidencialidad de la información en las empresas del sector salud, como el ocurrido con COLMEDICA en 2016 [13]. Por lo anterior es relevante no solo identificar si en Colombia cuentan con leyes que soporten el cumplimiento de requerimientos de seguridad para la protección de datos personales y de salud, también debe tomar relevancia el identificar si estos requerimientos legales son suficientemente claros y específicos para las organizaciones que requieren de su cumplimiento.

3.1. Normatividad nacional e internacional aplicables al sector salud

Con el fin de obtener un enfoque relacionado directamente a las necesidades de requerimientos de seguridad en Colombia, se identificó inicialmente los requisitos de ley que se deben cumplir hoy en día relacionado con la protección de datos personales. Adicionalmente se identificó otras normativas o circulares nacionales y leyes internacionales relacionadas al sector Salud y la protección de datos personales, para reforzar las leyes previamente identificadas.

3.1.1. Leyes Nacionales

La Superintendencia de Industria y Comercio dispone en su portal web con un conjunto de normativas de tipo leyes, circulares, decretos entre otras, para el conocimiento y acceso al público en general [53]. Esta base de datos cuenta con una serie de filtros que fueron determinantes para definir las normativas a tener en cuenta según el alcance de la tesis. Los filtros

se definieron de la siguiente forma: Años = todos, Tema = Protección de datos personales y Tipo de norma = Todos, obteniendo un resultado de 20 normativas.

Cada normativa seleccionada fue leída e interpretada con el fin de identificar cual de las 20 normativas contemplaba directrices o requerimientos relacionados a la seguridad informática. como resultado se obtiene un total de 5 normativas que incluyen un total de 20 enunciados. La descripción específica de cada una de las Leyes se encuentra en el Anexo A. Las normativas identificadas son:

- **Resolución N. 1995 de 1999:** Cuenta con un total de 3 ítems [54].
- **Ley 1266 2008:** Cuenta con un total de 7 ítem [55].
- **Ley 1581 del 2012:** Cuenta con un total de 7 ítem [5].
- **Resolución 1531 de 2014:** Cuenta con un total de 2 ítem [56].
- **Titulo V Circular Única SIC Marzo 2018:** Cuenta con un total de 1 ítems [57].

El alcance de la tesis tiene como prioridad el riesgo relacionado a la confidencialidad de la información, siendo necesario identificar los riesgos de disponibilidad, integridad y confidencialidad de la información para cada uno de los ítem previamente identificados en las normas. En algunos casos se logra identificar un riesgo para cada ítem, pero la generalidad en la descripción de las normas conlleva a clasificarlas con los tres riesgos. En la Figura **3-1** se observa el total de riesgos encontrados en las normativas previamente identificadas y en el Anexo A, en la columna de **Riesgo**, se puede observar el detalle por cada uno de los ítems definidos por la Ley.

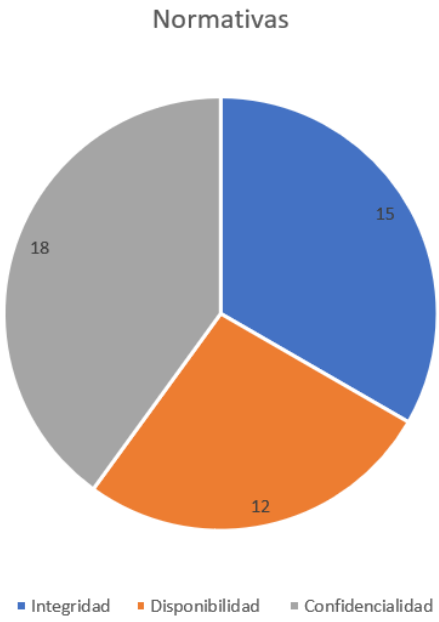

Figura 3-1.: Riesgos vs Normativas.
Fuente: Elaboración propia.

3.1.2. Normativas Nacionales e Internacionales

Teniendo en cuenta la generalidad de las normativas identificadas en la SIC, se realiza una verificación de otras normativas dispuestas por otros sectores como es el caso de la Superintendencia Nacional de Salud y la Superintendencia Financiera de Colombia. Estas entidades que ejercen funciones de vigilancia, inspección y control en cada una de sus áreas, cuenta con un documento denominado Circular única, que tiene como objetivo reunir en un solo cuerpo normativo todas las reglamentaciones e instrucciones generales expedidas por la entidad reguladora.

El resultado del análisis de la circular única de la Superintendencia Nacional de Salud [58], da como resultado la identificación de tres ítem relacionados a lineamientos de seguridad informática. El detalle de las leyes se encuentra en el Anexo B. Las leyes identificadas son:

- Titulo I, Capitulo Primero.
- Titulo II, Capitulo Primero, 1.2.1.2.
- Titulo II, Capitulo Primero, 1.2.1.2.

.
El resultado del análisis de la circular única de la Superintendencia Financiera de Colombia [59], da como resultado la identificación de 42 ítem relacionados a lineamientos de seguridad informática. El detalle de las leyes se encuentra en el Anexo C. Las leyes identificadas son:

- Capitulo I, 1.4, 1.4.1.1.1.5.
- Capitulo I, 1.4, 1.4.1.1.5.1.
- Capitulo I, 2, 2.3.3, 2.3.3.1, Del numeral 2.3.3.1.1 hasta el 2.3.3.1.18 y los numerales 2.3.3.1.20. y 2.3.3.1.21.
- Capitulo I, 2, 2.3.4, 2.3.4.9, Del numeral 2.3.4.9.1. hasta el 2.3.4.9.8.
- Capitulo I, 2, 2.3.4, 2.3.4.11, Del numeral 2.3.4.11.1. hasta el 2.3.4.11.6.
- Capitulo I, 2, 2.3.5, Del numeral 2.3.5.1. al 2.3.5.6.

Con el fin de identificar los lineamientos de seguridad acorde a la necesidad de proteger la confidencialidad de la información en las aplicaciones del sector salud, se consulta los requisitos de seguridad dispuestos por HIPAA y .

El resultado del análisis realizado a los lineamientos dispuestos por HIPAA [60], da como resultado la identificación de 23 ítem relacionados a lineamientos de seguridad informática. El detalle de las leyes se encuentra en el Anexo D. Las leyes identificadas son:

- **Administrative Safeguards:** 164.308(a)(1)(i), 164.308(a)(1)(ii)(A), 164.308(a)(1)(ii)(B), 164.308(a)(1)(ii)(C), 164.308(a)(1)(ii)(D), 164.308(a)(2), 164.308(a)(4)(i), 164.308(a)(4)(ii)(A), 164.308(a)(4)(ii)(B), 164.308(a)(4)(ii)(C), 164.308(a)(5)(ii)(D) y 164.308(a)(8).
- **Technical Safeguards:** 164.312(a)(1), 164.312(a)(2)(i), 164.312(a)(2)(iii), 164.312(a)(2)(iv), 164.312(b), 164.312(c)(1), 164.312(c)(2), 164.312(d), 164.312(e)(1), 164.312(e)(2)(i) y 164.312(e)(2)(ii).

El resultado del análisis realizado a los lineamientos dispuestos por la [61], da como resultado la identificación de 7 ítem relacionados a lineamientos de seguridad informática. El detalle de las leyes se encuentra en el Anexo E. Las leyes identificadas son:

- Reglamentos, numeral 35.
- Reglamentos, numeral 49
- Articulo32, Seguridad del tratamiento,1, a.
- Articulo32, Seguridad del tratamiento,1, b.
- Articulo32, Seguridad del tratamiento,1, d.
- Articulo32, Seguridad del tratamiento,2.

3.2. Estándares de seguridad para aplicaciones móviles

Para la definición de lineamientos de seguridad con un mayor enfoque técnico, que logre fortalecer los lineamientos ya identificados por las diferentes normativas, se consultó cuatro estándares de seguridad informática relacionada a dispositivos móviles, entre los cuales se encuentra OWASP, National Institue of Standars and Technology (NIST), Open Android Security Assessment Methodology (OASAM) y National Information Assurance Partnership (NIAP). Las conclusiones de cada una son las siguientes:

- **National Information Assurance Partnership (NIAP)**: Originalmente nace de la unión de esfuerzos entre la National Security Agency (NSA) y la NIST. Es una iniciativa por parte del gobierno de los Estados Unidos para cubrir las necesidades de pruebas de seguridad de consumidores y productores de tecnología de la información que operan en la NSA. NIAP cuenta con un documento denominado "Requirements for Vetting Mobile Apps from the Protection Profile for Application Software", se encuentra en la versión 1.2 con fecha del 22 de abril del 2016 [62]. Este documento es una linea base que explica los estándares de seguridad utilizados para analizar una aplicación móvil antes de aprobar su implementación en un entorno de gobierno. Está conformada por:
  - 25 Requerimientos funcionales de seguridad: Estos requisitos funcionales son presentados sin las actividades específicas del perfil de protección, con el fin de ayudar al lector con los requisitos de interés. También se observa requisitos funcionales de seguridad que no son de estricto cumplimiento para todos los casos.
  - 29 Requisitos funcionales de seguridad basados en la selección de requisitos básicos: Aplican dependiendo de los requisitos funcionales iniciales y diferenciado con la etiqueta [selección].
  - 3 Requisitos funcionales de seguridad objetiva: Son fuertemente deseados pero tecnológicamente no se encuentra ampliamente disponible a nivel comercial.
  - 2 Requerimientos funcionales de seguridad opcional: Como su nombre lo indica, están disponibles pero no es de estricto cumplimiento.
  - 4 Requerimientos de garantía de seguridad: No están relacionados directamente con lo funcional, se enfoca en el soporte del desarrollo para el producto, requisitos relevantes de seguridad (no funcional), proceso de desarrollo entre otros.

  En conclusión, se observa que al ser aplicaciones orientadas para entornos de entidades del gobierno, los lineamientos cuentan principalmente con un enfoque relacionado al uso de la criptografía, a la limitante de consumos de ciertos recursos de los dispositivos móviles como el micrófono, cámara entre otros. En la información de los lineamientos no da indicaciones de como validar los controles implementados.

- **Open Android Security Assessment Methodology (OASAM):** Nace como una metodología para la realización de pruebas de seguridad a las aplicaciones móviles que corren bajo el sistema operativo Android y su propósito es convertirse en un marco de referencia para este tipo de pruebas de seguridad mediante la elaboración de una taxonomía completa de vulnerabilidades, con el fin de apoyar tanto a los desarrolladores de aplicaciones como a las personas encargadas de validar su seguridad.

  OASAM es un proyecto que tiene como autor a Daniel Medianero, con especialidades en Auditoría de Seguridad en Aplicaciones web, Auditoría en código fuente java para aplicaciones web, Análisis de aplicaciones móviles y Auditoría en código fuente y Respuesta a incidentes informáticos [63]. OASAM cuenta con un total de 46 requisitos de seguridad, su última actualización según GitHub es del 29 de diciembre del 2016 y está conformado por:

  - **Recopilación de información:** Cuenta con 5 lineamientos y está relacionada con la recopilación de información y definición de la superficie de ataque.
  - **Gestión de la configuración e implementación:** Cuenta con 12 lineamientos y está relacionada con la identificación de errores en la configuración de la aplicación o los componentes que comprometen la seguridad de la aplicación.
  - **Autenticación:** Cuenta con 4 lineamientos y está relacionada con la evaluación de las funcionalidades asociadas con el uso de inicios de sesión a través de la aplicación.
  - **Criptografía:** Cuenta con 4 lineamientos y está asociada con la evaluación de las funcionalidades relacionadas con el uso de criptografía en la aplicación. Esto puede ocurrir al enviar o almacenar datos.
  - **Fuga de información:** Cuenta con 5 lineamientos y están asociados con la evaluación de la confidencial por fuga de información sensible que podría estar relacionada con el usuario o con el propio dispositivo móvil.
  - **Validación de datos:** Cuenta con 7 lineamientos y está relacionado con la evaluación de las entradas recibidas del usuario, que es administrada por la aplicación.
  - **Suplantación de intents:** Cuenta con 4 lineamientos y está relacionada con la entrega de intents arbitrarios a un componente que espera recibir otro tipo de intent.
  - **Recepción de intents no autorizados:** Cuenta con 4 lineamientos y está relacionada con la resolución de la entrega de intents implícita, que al ser explotadas puede permitir ataques de denegación de servicio o phishing.

- **Evaluación de la lógica de negocio de la aplicación:** Cuenta con 1 lineamiento y está relacionada a vulnerabilidades con componentes más centrados en el diseño en lugar de la codificación.

En conclusión, OASAM es un framework que sigue en construcción debido a que no todos los lineamientos se encuentran debidamente documentados. Tiene un enfoque técnico y orientado a pruebas de seguridad. Los lineamientos documentados cuenta con una descripción, riesgo, recomendaciones y referencias.

- **National Institue of Standars and Technology (NIST):** fundado en 1901 y forma parte del Departamento de Comercio de los Estados Unidos. Sus laboratorios de ciencias físicas son de los mas antiguos en el país. NIST tiene la responsabilidad del desarrollo de estándares y guías de seguridad de la información, incluidos 61 requisitos mínimos para los sistemas de información federales. Dentro de sus estándares, NIST realizó una publicación especial en Julio del 2018 de un Draft identificado como SP 800-163 con título "Vetting the Security of Mobile Applications" [64]. El documento trata de forma detallada un proceso de validación de aplicaciones móviles, con el fin de garantizar el cumplimiento a los requisitos de seguridad exigidos para una organización y estén bajo un riesgo de seguridad aceptable.

  El documento "Vetting the Security of Mobile Applications" cuenta con un total de 16 requisitos de seguridad, identificados en el documento como:

  - **4 Requisitos generales:** Estos requisitos se apoyan de otros estándares relacionados a la seguridad, definidos por NIAP, OWASP, Criterios de evaluación definidos por MITRE y la NIST en su publicación especial SP 800-53.
  - **Requerimientos específicos de la organización basado en 12 criterios de seguridad:** Estos criterios ayudan a identificar los factores relacionadas con las diferentes vulnerabilidades que pueden afectar las políticas, regulaciones o guías definidas por la organización.

  En conclusión, Los lineamientos de seguridad descritos en el documento "Vetting the Security of Mobile Applications", se basa en otros estándares de seguridad y las pruebas de backend están fuera del alcance del documento. La publicación especial SP 800-53 mencionada por NIST contiene lineamientos generales que abarca cualquier arquitectura y no específicamente la móvil. Se observa un enfoque principalmente en el proceso de validación de las aplicaciones móviles, es decir ayuda a entender el que? pero no el como?.

- **Open Web Application Security Project (OWASP):** Es un proyecto que se caracteriza por ser de código abierto, tiene como función identificar y controlar las causas que afectan la seguridad del software. OWASP es una organización sin ánimo de lucro [65].
  OWASP dentro de sus diversos proyectos cuenta con el Owasp Mobile Security, destinado para ofrecer a desarrolladores y personal de seguridad informática los recursos necesarios para crear y mantener las aplicaciones móviles en un nivel de seguridad aceptable. El enfoque principal está relacionado a la capa de aplicación y se centran en la integración entre la aplicación móvil, los servicios encargados de la autenticación remota y las características específicas de la plataforma en la nube.

  OWASP cuenta con un documento llamado Mobile Application Security Verification Standard v1.1 [66], en su repositorio de GitHub realizan las actualizaciones para sus nuevas versiones con una actividad constante. El documento dispone de 8 dominios y un total de 74 lineamientos de seguridad identificados de la siguiente manera.

  - **10 requisitos en Arquitectura, Diseño y Modelado de Amenazas:** Es el único dominio que no cuenta con los casos de pruebas técnicas, pero resaltan su importancia al momento de pensar e implementar los controles de seguridad y el plan de pruebas a realizar.
  - **12 requisitos en Almacenamiento de Datos y Privacidad:** Su enfoque se relaciona a la protección de los datos confidenciales como es el caso de las credenciales de acceso del usuario y la información privada.
  - **6 requisitos en Criptografía:** Valida que la criptografía utilizada está alienada con las mejores prácticas de seguridad de la industria.
  - **11 requisitos en Autenticación y Gestión de la sesión:** Define algunos requisitos generales relacionado a la administración y gestión de cuentas de usuario y la sesión.
  - **6 requisitos en las Comunicaciones de Red:** Tiene como fin garantizar la confidencialidad y la integridad de la información que es intercambiada entre la aplicación móvil y los endpoints.
  - **8 requisitos en Interacción con la Plataforma:** Tiene como objetivo validar que la aplicación use las API de la plataforma y los componentes estándar de forma segura, incluyendo las comunicaciones entre aplicaciones.
  - **9 requisitos en Calidad de Código y Configuración del Compilador:** El objetivo consiste en garantizar el desarrollo de aplicaciones bajo prácticas básicas de codificación segura y la activación de características ofrecidas por el compilador de forma gratuita.
  - **12 requisitos en Resilencia:** Este tipo de requerimientos está relacionada a las

medidas de defensa en profundidad, especialmente para aplicaciones que gestionan datos o funciones de carácter confidencial.

En conclusión, se observa que OWASP Mobile Application Security Verification v1.1 es un proyecto de constante evolución, 7 de sus 8 dominios abarcan en gran parte los diferentes riesgos que se pueden encontrar en entornos de aplicaciones móviles. También dispone de un documento que describe la forma de evaluar si las aplicaciones móviles cumplen o no con los requisitos de seguridad planteados.

Basados en las conclusiones expuestas en los diferentes estándares propuestos en la tesis, se decide adoptar como base para la definición de los lineamientos se seguridad, el estandar ofrecido por OWASP con su documento Mobile Application Security Verification v1.1. El resumen de los estándares analizados se observa en la Figura **3-2**.

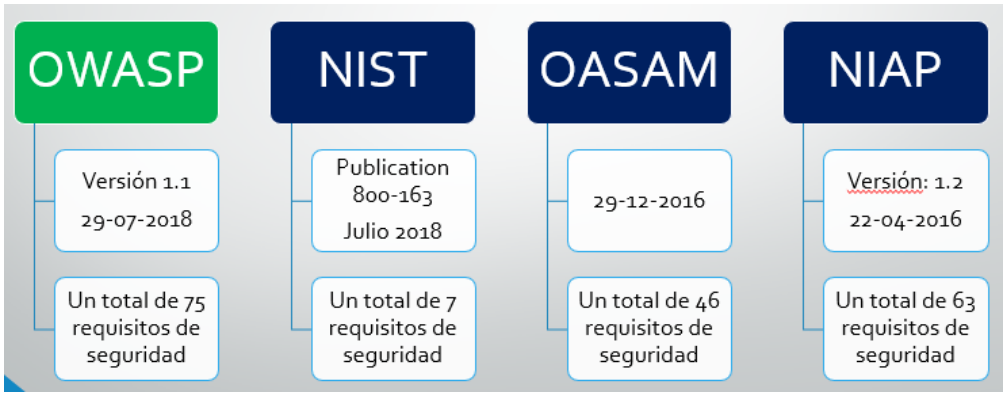

Figura 3-2.: Comparativo entre estándares de seguridad.
Fuente: Elaboración propia.

3.3. Definición de Lineamentos de Seguridad

Con los lineamientos de seguridad a nivel técnico identificados en el documento de OWASP Mobile Application Security Verification v1.1, en la Superintendencia Nacional de Salud, en la Superintendencia Financiera de Colombia, en HIPAA y en el GDPR, se cuenta con el insumo necesario para definir los lineamientos de seguridad a proponer.

3.3.1. Depuración de Lineamientos de seguridad consultados

Se realizó un filtro de cuales lineamientos de las normativas serán adoptadas, cuales no son parte del alcance de la tesis y cuales son reemplazadas por otros lineamientos. El resultado del análisis e identificación de los lineamientos de seguridad es el siguiente:

- **Superintendencia Nacional de Salud:** Los tres lineamientos identificados no aplican para el alcance definido en la tesis debido a que los requisitos mencionados están relacionados a las comunicaciones establecidas entre las entidades del sector salud y la Supersalud y no entre el usuario final y las entidades del sector salud.
- **Superintendencia Financiera de Colombia:** El detalle del resultado se encuentra descrito en el Anexo C en la columna **Resultado**. En resumen se tiene:
  - De los 42 ítem, 15 se identificaron como controles internos, es decir los alcances definidos son propios de las entidades financieras a nivel de sus procesos de TI (Hardware y software), alcances del tipo seguridad corporativa como lineamientos de seguridad para sus empleados y procesos administrativos.
  - Se identificó 6 ítem clasificados como fuera del alcance, debido a que involucran temas de experiencia de usuario, su alcance es estrictamente para el escenario financiero y se observó un caso que está relacionado con denegación del servicio.
  - Se adoptaron 5 ítem para ser incluidos en los lineamientos generales propuestos para el alcance definido en le objetivo 1.
  - Se identificó 16 ítem que no se tuvieron en cuenta debido a que se adoptó ítem similares de otras fuentes como HIPAA y OWASP.

- **HIPAA**: El detalle del análisis realizado se encuentra descrito en el Anexo D en la columna **Resultado**. En resumen se tiene:
  - De los 23 ítem identificados, 12 se adoptaron para ser incluidos en los lineamientos generales propuestos para el alcance definido en el objetivo 1.
  - se identificaron 5 ítem como controles internos, es decir los alcances definidos son propios de las entidades financieras a nivel de sus procesos de TI (Hardware y software), alcances del tipo seguridad corporativa como lineamientos de seguridad para sus empleados y procesos administrativos.
  - Se identificó 2 ítem clasificados como fuera del alcance, debido a que involucran riesgos relacionados a la integridad de la información.
  - los 4 ítem faltantes se reemplazaron por otros lineamientos definidos en la circular única de la Superfinanciera, por OWASP y por los definidos en los mismos lineamientos de HIPAA.

- **General Data Protection Regulation (GDPR):** El detalle del análisis realizado se encuentra en el Anexo E. En resumen se tiene:
  - De los 7 ítem identificados se adoptó 2 parcialmente con el fin de incluir la resiliencia de los sistemas y servicios, un enfoque relativamente nuevo integrado a los aspectos de seguridad de la información.
  - los 5 ítem restantes no se adoptaron debido a que el alcance estaba cubiertos por lineamientos de HIPAA, OWASP y la Superfinanciera.

En total se adoptaron 19 normativas nacionales e internacionales. El mayor número de adopciones se obtiene de HIPAA y particularmente no se adopta ningún lineamiento de la Superintendencia Nacional de Salud. El resultado general de las adopciones de las normativas expuestas anteriormente se pueden visualizar en la Figura **3-3**.

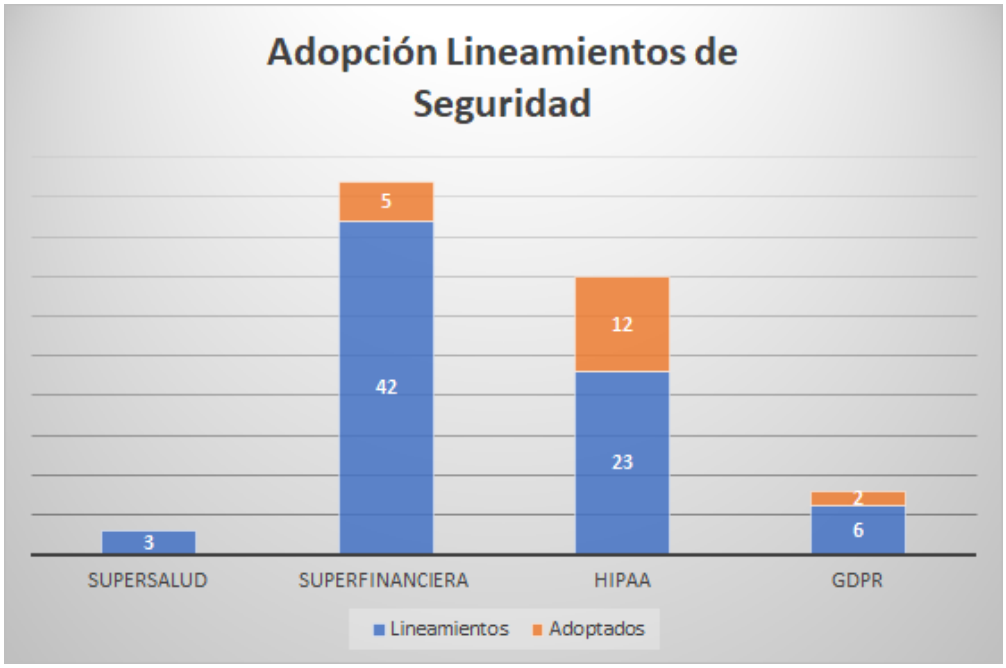

Figura 3-3.: Lineamientos de seguridad adoptados.
Fuente: Elaboración propia.

3.3.2. Construcción de los Lineamientos de seguridad

Con la información obtenida del análisis realizado, se construyó los lineamientos tomando como base los requisitos de seguridad descritos por OWASP. Ver Figura **3-4** y los definidos

por las normativas nacionales e internacionales. Una forma gráfica de observar cuales requisitos se adoptaron, cuales no y que requisitos comparten en común se pueden observan en la Figura **3-5**.

Figura 3-4.: Esquema construcción de lineamientos.
Fuente: Elaboración propia.

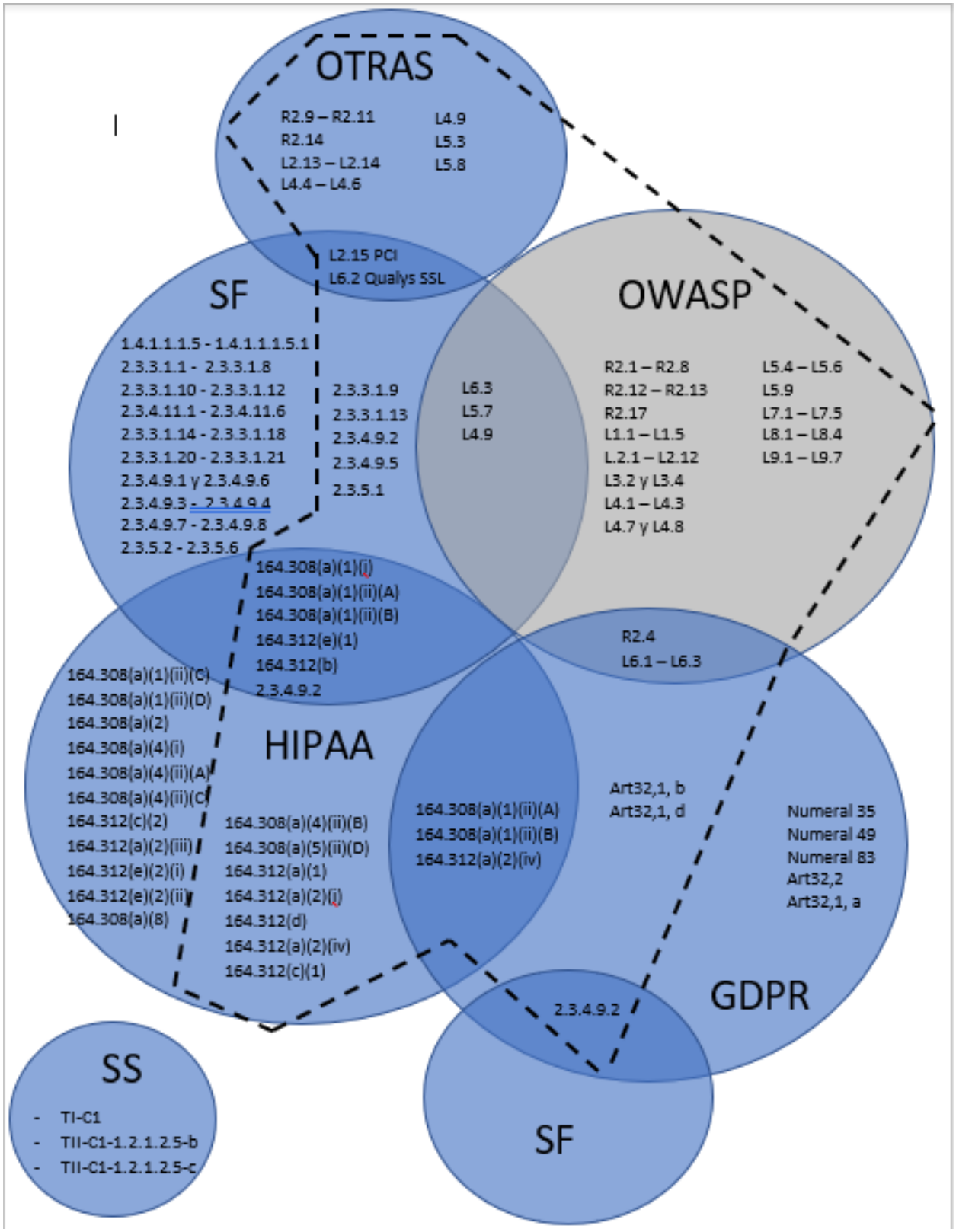

Figura 3-5.: Definición lineamientos de seguridad.
Fuente: Elaboración propia.

Como resultado final, se obtiene el total de requerimientos de seguridad a implementar, dividido en 11 dominios de la siguiente manera:

- **Gestión de la seguridad:** En este dominio los requerimientos de seguridad están directamente relacionados a la Administración de la seguridad, la implementación de políticas, procedimientos, Gestión del Riesgo, Resiliencia en los sistemas entre otras tareas importantes que ayudarán a soportar los lineamientos técnicos expuestos en la tesis. Los lineamientos están principalmente soportado por lo dispuesto en HIPAA. Estos lineamientos no contarán con una guía de validación.

- **Arquitectura, Diseño y Modelado de Amenazas:** Los requerimientos de seguridad se enfocan en la necesidad de contar con un proceso de Ciclo de Vida de Desarrollo Seguro del Software Secure SDLC (Secure Software Development Life Cycle), garantizando la participación de seguridad en la definición de la arquitectura y diseño de la aplicación, la necesidad de disponer de un modelado de amenazas que permita identificar los diferentes vectores de ataque y la existencia de sus controles. Desde el Diseño se pueden evitar fallos de seguridad al estar en las etapas tempranas de su desarrollo. Estos lineamientos no contarán con una guía de validación.

- **Backend:** Los requerimientos de este dominio se pensaron principalmente en la interacción de la aplicación con el endpoint o backend que expone los servicios. Su alcance solo contempla las direcciones IP y puertos expuestos para la aplicación y los controles a nivel de lógica de negocio que debe validar el backend. En otras palabras, los requisitos se definen bajo los permisos otorgados por el Backend a la aplicación móvil antes y después de una autenticación.

- **Almacenamiento de Datos y Privacidad:** Los requerimientos de seguridad se enfocan en el almacenamiento no seguro de información sensible, bajo el contexto del sistema operativo móvil. (Logs, sandbox, memoria, entre otros) y la data enviada entre la aplicación móvil y el Backend.

- **Criptografía:** El presente dominio cuenta con requerimientos de seguridad relacionado a los mecanismos de cifrado utilizados por la aplicación, recomendaciones relacionadas a su uso y la robustez del cifrado necesaria para evitar exposición de la información sensible o protegida.

- **Autenticación:** Los requerimientos de seguridad están relacionados con las buenas prácticas específicamente al momento de establecer un mecanismo de autenticación en la aplicación móvil, contemplando los principales vectores de ataque en esta funcionalidad tanto para el usuario como para la contraseña y posible segundo factor de autenticación.

- **Manejos de Sesiones y Autorización:** Los requerimientos de seguridad expresos en este dominio contemplan la verificación de la correcta visibilidad que tiene tanto el servidor como el usuario final legítimo, de las actividades que ocurren en las diferentes funcionalidades de la aplicación, para evitar accesos no autorizados o posible suplantación de identidad.

- **Comunicación a través de la red:** El dominio contiene requerimientos de seguridad enfocados al cifrado de las comunicaciones, la configuración y uso correctos de algoritmos seguros y validaciones de integridad que eviten exposición de información por medio de ataques de hombre en el medio.

- **Interacción con la Plataforma:** Los requerimientos de seguridad están relacionados específicamente a riesgos que se pueden presentar en las comunicaciones internas entre la aplicación y el sistema operativo móvil, conocidas como Inter-Process Communication (IPC), permisos mínimos de la aplciación sobre el sistema operativo y riesgos por vulnerabilidades en la gestión de los webview.

- **Calidad de Código y Configuración del Compilador:** Los requerimientos expuestos en este dominio están relacionados a las buenas prácticas en la compilación de la aplicación y codificación básica segura como el manejo de excepciones y registros de logs. Estos requerimientos son transversales a la aplicación y aporta resiliencia a los requerimientos expuestos previamente.

- **Impedir el Análisis Dinámico y la Manipulación:** Como lo menciona OWASP, este tipo de requerimientos está relacionado a las medidas de defensa en profundidad, especialmente para aplicaciones que gestionan datos o funciones de carácter confidencial. Estos requerimientos son transversales a la aplicación y aporta resiliencia a los requerimientos expuestos previamente.

El resumen de los lineamientos propuestos se encuentran descritos en la tabla **3-1** y el detalle de cada uno de los requisitos que forman parte de cada uno de los dominios propuestos, se encuentra en el Anexo G.

Lineamientos Consolidados	Total Requisitos	Categoría
Gestión de la seguridad	11	Recomendaciones
Arquitectura, Diseño y Modelado de Amenazas	18	
Backend	5	Lineamiento
Almacenamiento de Datos y Privacidad	15	
Criptografía	4	
Autenticación	9	
Manejos de Sesiones y Autorización	9	
Comunicación a través de la red	3	
Interacción con la Plataforma	5	
Calidad de Código y Configuración del Compilador	4	Lineamiento transversal
Impedir el Análisis Dinámico y la Manipulación	7	
Total	90	

Tabla 3-1.: Estructura - lineamientos de seguridad propuestos.
Fuente: Elaboración propia.

Con el fin de lograr un acercamiento claro y describir un panorama completo ante la alta gerencia y el área de cumplimiento de una organización, sobre los lineamientos de seguridad a cumplir para fortalecer los requisitos de seguridad que exige una Ley en particular, se realiza un mapeo de cada una de las leyes identificadas vs Lineamientos de seguridad definidos. Como resultado se observa que la mayor parte de las leyes requieren el cumplimiento del total de lineamientos propuestos a validar (61), debido a la generalidad dispuesta en los requerimientos de seguridad. El resultado general se puede observar en la Figura **3-2** y el detalle de los lineamientos de seguridad a cumplir por cada una de las leyes se puede observar en el Anexo F.

Con este resultado se logra establecer los requerimientos mínimos de seguridad para el tratamiento de la información gestionada por las aplicaciones móviles relacionadas principalmente en el sector salud, que ayudarán al cumplimiento de los requisitos de ley enfocados en la protección de datos personales en Colombia.

Normativa	Dominio									Total
	L1	L2	L3	L4	L5	L6	L7	L8	L9	
Resolución N. 1995 /Capítulo III/Articulo 18	2	0	4	9	7	1	2	4	7	36
Ley 1266 /Titulo I/Artículo 4/c)	4	11	4	9	9	3	2	4	7	53
Ley 1266 /Titulo I/Artículo 4/f)	5	15	4	9	9	3	5	4	7	61
Ley 1266 /Titulo III/Artículo 7/3.	4	15	4	9	9	3	0	4	7	55
Ley 1266 /Titulo III/Artículo 7/6.	4	15	4	9	9	3	5	4	7	60
Ley 1266 /Titulo IV/Artículo 11/3.	5	15	4	9	9	3	5	4	7	61
Ley 1581/Titulo II/Artículo 4/g.	5	15	4	9	9	3	5	4	7	61
Ley 1581/Titulo VI/Artículo 17/d)	5	15	4	9	9	3	5	4	7	61
Ley 1581/Titulo VI/Artículo 18/b)	5	15	4	9	9	3	5	4	7	61
Ley 1581/Titulo VI/Artículo 18/j)	4	11	4	9	9	3	2	4	7	53
Resolución 1531/Articulo 3	5	15	4	9	9	3	5	4	7	61
Resolución 1531/Articulo 8	5	15	4	9	9	3	5	4	7	61

Tabla 3-2.: Leyes vs Lineamientos de seguridad.
Fuente: Elaboración propia.

4. Modelado de Amenazas

La seguridad en las aplicaciones en un entorno de conectividad creciente, tiende a ser cada vez más crítico. Las razones pueden ser diversas, ya sea por fallas constantes en el diseño de arquitectura que son aprovechadas por un atacante, por el nacimiento de nuevas técnicas para la explotación de vulnerabilidades en las aplicaciones, por la necesidad del negocio de exponer nuevas funcionalidades para la mejora de los servicio, por la no verificación periódica de la seguridad de las aplicaciones, entre otras, nos lleva a la necesidad de incluir la seguridad desde el diseño de la aplicación, con el fin de determinar de una forma predictiva y repetible el estado de exposición a las amenazas y el riesgo en un entorno tecnológico [67].

Para lograr un análisis repetible, es necesario contar con mecanismos que ayuden a la implementación de una metodología basada en buenas prácticas del mercado. Uno de estos mecanismos está relacionado con el modelamiento de amenazas. este proceso permite identificar, cuantificar y direccionar los diferentes riesgos de seguridad que está asociados a una aplicación, ayudando a priorizar y focalizar los esfuerzos de validación de seguridad.

4.1. Definición del modelado de amenazas a utilizar

Para la realización del modelado de amenazas se tiene como referente STRIDE, Arbol de Ataques y Librería de Ataques. Con el fin de obtener el mejor resultado en el proceso, se decide utilizar un modelo híbrido expuesto por Sriram Krishnan en un articulo publicado en febrero de 2017 [1]. La propuesta hace referencia a un modelado de amenazas híbrido, que consiste en agrupar lo mejor de cada uno de los tres modelos expuestos anteriormente y obtener un resultado mejor estructurado, capturando los detalles más óptimos y representar los datos de una manera mas entendible.

Una descripción general de los tres modelos se detallan a continuación:

STRIDE: Es un modelo de amenazas que surge en 1992 generada por Microsoft. Es una metodología que ha sido ampliamente aceptada en el mercado y su nombre proviene de las palabras en inglés: Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, Elevation of privilege [68].

Explicación de los atributos STRIDE:

- **Usurpación de identidad (Spoofing):** Es uno de los riesgos más críticos para las aplicaciones que proveen identificadores para cada usuario, pero solamente un usuario de conexión a la base de datos. En particular, un usuario jamás debería poder convertirse en otro usuario o asumir atributos de otro usuario.
- **Modificación de datos (Tampering with data):**Los usuarios pueden potencialmente modificar datos que les han sido entregados a ellos, retornarlos y potencialmente controlas las validaciones del lado del cliente o validaciones que no se hacen en el servidor.
- **Repudio(Repudiation):**Los usuarios pueden rechazar transacciones si no hay suficiente auditoría o mantenimiento de registros de su actividad.
- **Divulgación de información (Information disclosure):**Los usuarios tienen derecho a mantener la privacidad de sus transacciones y de sus datos. No debería ser posible hacer pública información que no le compete al usuario.
- **Denegación de servicio (Denial of service):**Los diseñadores de las aplicaciones deben ser conscientes de que su aplicación puede ser sujeta a un ataque de denegación de servicio, esto se puede dar ofreciendo archivos muy grandes, cálculos complejos, búsquedas muy pesadas o con resultados extensos, a usuarios anónimos o no autenticados.
- **Elevación de privilegios (Elevation of privilege):**Cuando en una aplicación que tiene diferentes roles un usuario busca tener más privilegios de los que actualmente tiene.

STRIDE cuenta con dos variantes, STRIDE per Elelement y STRIDE per Interaction [1]. Para el alcance de esta tesis se tiene especial interés en STRIDE per Interaction, teniendo en cuenta que se puede identificar los diferentes escenarios de amenaza cuando se tiene aplicaciones con diferentes roles. Un ejemplo de su estructura general se observa en la tabla **7-1**.

Según lo indica Sriram Krishnan en su artículo, las limitaciones de STRIDE se enfoca en la no captura del cómo llegar a mitigar las amenazas identificadas. Es importante mapear unas contramedidas sólidas para las amenazas previamente identificadas. El autor también aclara que las medidas de codificación segura no es una característica de esta técnica.

Elementos	Interacción	Escenario de Amenaza	S	T	R	I	D	E
Flujo de información	Admin	Un atacante puede enviar un documento XML malicioso que contendría una expansión de entidades anidadas para producir una salida XML amplificada que bloqueará la CPU / memoria del procesador (ataque de expansión de la entidad XML), ya que la aplicación permite el uso de DTD					X	
Flujo de información	Empleado	Un atacante puede excluir la firma XML del mensaje para enviar solicitudes arbitrarias e invocar funciones para obtener detalles salariales de los empleados.	X				X	

Tabla 4-1.: STRIDE por Interacción.
Fuente: Elaboración propia.

Árbol de Ataques: Se ha presentado como un enfoque completo para el modelado de amenazas y se define como un modelo que permite por medio de una representación gráfica estructurada en forma de árbol, enumerar las acciones que se deben realizar para comprometer un objetivo específico y como se relacionan entre sí [69]. Este modelo permite identificar las amenazas bajo el rol de un atacante.

Para crear un Árbol de Ataques, se debe realizar los siguientes pasos básicos:

- **Decidir una forma de representación:** Existen dos tipos de representación de un Árbol de Ataque, tipo AND y tipo OR. La mayoría de los árboles son del tipo OR es decir, un nodo es verdadero si alguno de sus subnodos es verdadero, caso contrario del los tipo AND, donde un nodo es verdadero si todos los subnodos que lo anteceden son verdaderos.
- **Crea un nodo raíz:** El nodo raíz corresponde a un objetivo que desea un atacante y normalmente puede elaborarse diferentes objetivos en árboles independientes.
- **Crear subnodos:** Los subnodos pueden ser creados basado en una serie de ideas que tiene como premisa lograr alcanzar el objetivo. La relación entre los subnodos puede ser de tipo AND y de tipo OR.
- **Afinación:** En este punto se recomienda realizar una validación al Árbol de Ataque creado, con el fin de identificar si existe la necesidad de agregar árboles adicionales o

revisar si en cada nodo se contempló todos los posibles vectores de ataque.

- **Pode el árbol:** Este paso tiene como fin recorrer cada uno de los nodos e identificar si vale la pena o no contemplar el nodo propuesto, teniendo en cuenta la existencia de mitigantes o que no es un escenario posible bajo el contexto del alcance del proyecto analizado, marcando el nodo con un color diferente. De esta forma queda la evidencia de que sí se contemplo el escenario en particular.
- **Revisar la presentación:** Es relevante tener en cuenta la presentación de los arboles de ataque, de tal forma que se pueda visualizar en una hoja para su mejor entendimiento y análisis. En caso de presentarse dificultad en su lectura, se recomienda dividir el árbol.

En la Figura **4-1** como ejemplo, se observa el diagrama de un Árbol de Ataque.

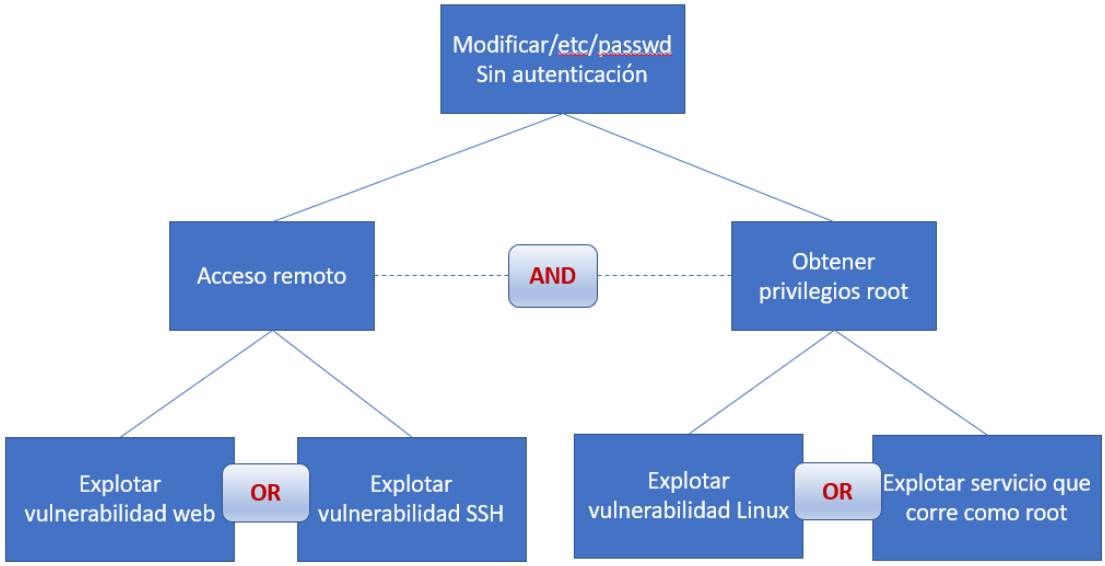

Figura 4-1.: Árbol de Ataque.
Fuente: Elaboración propia.

Por otro lado, las limitaciones del Árbol de Ataque se enfoca en la no consideración de ciertos atributos de los ataques como son la probabilidad, el costo y las contramedidas [1]. El intentar capturar estos datos complicaría la estructura y legibilidad del Árbol. Esta técnica de modelado al utilizarse de forma independiente no genera el mejor resultado.

Librería de Ataques: Es un conjunto de ataques utilizados para identificar posibles amenazas contra las aplicaciones. Esta técnica de modelado de amenazas se desarrolla bajo la

perspectiva de un atacante. El enfoque está en obtener el mayor número de detalles para un tipo de ataque en específico, que permita entender el panorama de amenazas e identificar las posibles contramedidas [70].

Una Librería de Ataques puede ser construida contemplando los siguientes puntos:

- **Lista de posibles ataques contra la aplicación:** Este tipo de listas pueden ser creadas por las mismas organizaciones o utilizar las listas formales publicadas por comunidades como Common Weakness Enumeration (CWE), OWASP y Common Attack Pattern Enumeration and Classification (CAPEC).
- **Lista de vulnerabilidades potenciales en la aplicación:** El enfoque principal se relaciona a vulnerabilidades del software corresponde a una falla en la aplicación que puede ser aprovechada por un atacante para lograr por ejemplo un acceso, o la denegación de un servicio.
- **Lista de debilidades introducidas en la aplicación:** Este punto se enfoca especialmente en los errores de programación, que inyectan vulnerabilidades en la aplicación.

El ejemplo de una librería de Ataque para Autenticación, se observa en la tabla **4-2**.

Las limitaciones de la Librería de Ataques se relaciona a la imposibilidad de aplicarse en la fase de diseño, al ser un modelo basado en listas de verificación, con información abundante que requiere de referencias, lo cual conlleva a procesos largos y tediosos, incluso radica en lo abstracto y en muchos casos no concuerdan con los lenguajes de programación o tecnologías que utiliza una organización [1].

Modelo Híbrido: El modelo Híbrido tiene como fin lograr adoptar lo mejor de cada una de los modelos expuestos anteriormente y para lograr el objetivo, el autor propone incluir tres aspectos importantes, Enfoque estructurado, Detalle óptimo y legibilidad [1].

- **Enfoque estructurado:** Para obtener un resultado deseado, se debe establecer un objetivo, realizar la captura de detalles requeridos e implementar un procedimiento que permita alcanzar el objetivo planteado [1].
- **Detalle óptimo:** El exceso de información o el contar con información al mas mínimo detalle, afectará el resultado del ejercicio y por ende la seguridad de la aplicación, razón que motiva al autor a proponer la optimización de la información con el fin de obtener los mejores resultados [1].
- **Legibilidad:**Se destaca la importancia que tiene la adecuada representación de los datos para garantizar un buen resultado del ejercicio y para ello se recomienda el uso de un Diagrama de Flujo de Datos (DFD) [1].

Debilidades	Vulnerabilidades	Ataque
CWE-287: Improper Authentication CWE-200: Exposure of Sensitive Information to an Unauthorized Actor. CWE-312: Cleartext Storage of Sensitive Information. CWE-319: Cleartext Transmission of Sensitive Information. CWE-602: Client-Side Enforcement of Server-Side Security. CWE-925: Improper Verification of Intent by Broadcast Receiver.	M5-Poor Authorization and Authentication	CAPEC-114: Authentication Abuse CAPEC-115: Authentication Bypass CAPEC-49: Password Brute Forcing CAPEC-302: Authentication Bypass by Assumed-Immutable Data CWE-305: Authentication Bypass by Primary Weakness CWE-308: Use of Single-factor Authentication

Tabla 4-2.: Ejemplo de Librerías de Ataque OWASP, CWE y CAPEC.
Fuente: Elaboración propia.

En la Figura **4-2** se observa un proceso de modelado de amenazas propuesto para elaborar un Modelado de Amenazas Híbrido.

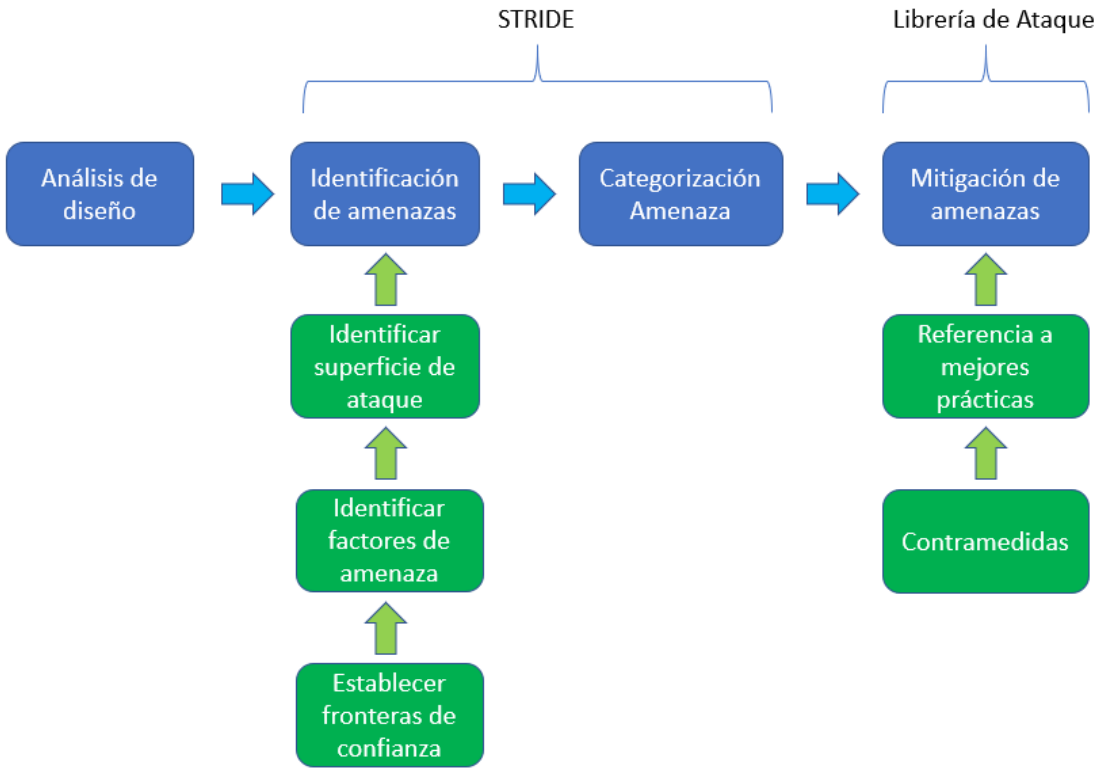

Figura 4-2.: Proceso de modelado de amenazas.
Fuente: Adaptado de [1]

La plantilla para mostrar los datos relacionados al modelado de amenazas híbrido es la siguiente:

Columna-1: Ref. No: Nomenclatura utilizada para identificar un escenario de amenaza de forma única.

Columna-2: Elementos: Hace referencia al elemento que puede ser del tipo entidad externa, flujo de datos, almacenamiento de datos y lógica de negocio, al que asigna la amenaza.

Columna-3: Interacción: Corresponde a un flujo específico en la aplicación, que forma parte del escenario de amenaza.

Columna-4: Escenario de amenaza, describe el caso de abuso teniendo en cuenta el elemento y el flujo específico en la aplicación.

Un ejemplo de las primeras cuatro columnas se observa en la tabla **4-3**:

Ref No.	Elementos	Interacción	Escenario de amenaza
BK-01	Backend (Servicios)	Paciente	Un atacante logra acceder a información sensible o privada del paciente al lograr obtener las cookies de sesión de la aplicación basado en un ataque de inyección de javascript XSS (Cross Site Scripting)
CT-02	Criptografía	Paciente	Un atacante logra acceder a información sensible o privada del paciente que se encuentra cifrada, utilizando las llaves de descifrado quemadas en el código, expuestas en la mensajería , o almacenadas de forma no segura.

Tabla 4-3.: Modelado de amenazas híbrido - Parte 1.
Fuente: Elaboración propia.

Columna-5 a la 10: STRIDE: Las 6 columnas son utilizadas para categorizar la amenaza identificada. (Spoofing, Tampering, Repudiation, Information Disclosure, Denail of Service y Elevation of Privilege).
Columna-11: Contramedidas: Se describe el detalle conceptual sobre el o los controles de seguridad que deben ser implementados para mitigar el riesgo generado por la amenaza identificada.

Un ejemplo de las siguientes 7 columnas se observa en la tabla **4-4** :

Columna-12 a la 14: Librería de Ataque: Describe las mejores prácticas al entregar una descripción de los posibles ataques y prácticas a seguir a nivel de codificación segura, para mitigar el riesgo que genera la amenaza previamente identificada. Las soluciones que proporciona el autor son: CAPEC, CWE y OWASP Top ten.

Un ejemplo de las últimas 3 columnas se observa en la tabla **4-5** :

Bajo el alcance de la presente Tesis, el modelo de amenazas a utilizar corresponde al Modelado de Amenazas Híbrido [1]. La única diferencia corresponde a que solo se tomará como

S	T	R	I	D	E	Contramedidas
X			X			L1.1: Configurar de forma correcta las cabeceras en el servidor web.
X	X		X			L2.14: No utilizar credenciales quemadas en el código (Hard-coded). L3.2: La aplicación no depende únicamente de criptografía simétrica, con claves quemadas en el código. Se recomienda el de cifrado asimétrico.

Tabla 4-4.: Modelado de amenazas híbrido - Parte 2.
Fuente: Elaboración propia.

Librería de Ataque		
CWE	CAPEC	OWASP Top Ten
CWE-926: Improper Export of Android Application Components. CWE-927: Use of Implicit Intent for Sensitive Communication.	CAPEC-629: Unauthorized Use of Device Resources	A6 Uso inapropiado de plataforma.
CWE CATEGORY: 7PK - Code Quality	CAPEC-175: Code Inclusion	A7- Calidad código lado cliente.

Tabla 4-5.: Modelado de amenazas híbrido - Parte 3.
Fuente: Elaboración propia.

referencia la librería de ataque propuesta por OWASP Mobile Top 10, teniendo en cuenta que su alcance es exclusivo para dispositivos móviles.

4.2. Análisis de aplicaciones relacionadas al sector salud

Se analizó el comportamiento funcional en común de un conjunto de aplicaciones móviles relacionadas al sector salud en Colombia desde la tienda de Google Play, con el fin de identificar el tipo de datos, arquitectura y servicios que consumen normalmente este tipo de aplicaciones. Los resultados obtenidos son:

- **Arquitectura:** Se observa que las aplicaciones móviles del sector salud consultadas, se encuentran bajo una arquitectura de cliente/servidor. Los servicios web son accedidos por Simple Object Access Protocol (SOAP) o REST.

Las aplicaciones utilizan Stateless authentication o Stateful authentication, es decir algunas aplicaciones utilizan la tradicional autenticación basada en cookies y otras utilizan una autenticación moderna basada en tokens. La autenticación se realiza con usuario y contraseña, donde el usuario siempre es el número de cédula.
Las aplicaciones móviles del sector salud fueron desarrolladas bajo la clasificación de aplicaciones del tipo Híbridas y Nativas. Las tecnologías utilizadas para el desarrollo y exponer los servicios son variadas, se encontró por ejemplo Oracle Application Server, Apache, IIS, Dojo, Java y ASP Net.

- **Tipo de datos:** Se realiza una navegación por las diferentes funcionalidades de las aplicaciones móviles relacionadas al sector salud y se identificó los siguientes datos del tipo privados y sensibles.Los datos son: Número de cédula o tarjeta de identidad, nombre completo de cotizante y afiliados, dirección de residencia, número de teléfono y celular, fecha de nacimientos del cotizante y afiliados, correo electrónico, número de tarjeta de crédito parcial, datos y estado de afiliación, descripción de consultas médicas, detalles cita médica (Lugar, fecha y hora), descripción autorizaciones, resultados de laboratorio y facturas.

- **Funcionalidades:** Se realiza un recorrido por las aplicaciones con el fin de identificar que tipo de funcionalidades entregan este tipo de aplicaciones a sus usuarios. Las funcionalidades identificadas son: Recordar contraseña, registrarse como usuario, consultar y crear citas médicas, descargar autorizaciones, consultar y descargar resultados de laboratorio, opción de recordar usuario y clave, consultar estado de afiliación, consultar información de afiliado, consultar procedimientos médicos, descarga de archivos en pdf y consulta y descarga de certificados.

4.3. Construcción del modelado de amenazas

Dentro del proceso de construcción del modelado de amenazas es importante definir las siguientes áreas para comprender las posibles amenazas a las que puede estar expuesta la aplicación:

- **Análisis de diseño:** En este punto se describirá un diagrama de arquitectura general con el fin de identificar el flujo de los datos en los diferentes componentes de la aplicación. También se resaltará las áreas que deben ser consideradas bajo el contexto de seguridad de la aplicación móvil.

El Diagrama de arquitectura a alto nivel propuesto, se divide en tres grandes áreas a tener en cuenta al momento de abarcar los temas relacionados a la seguridad en las aplicaciones móviles. A continuación se describe cada uno de ellos:

Diagrama Parte 1: El primer escenario corresponde a la interacción que tiene la aplicación móvil con el sistema operativo. Por ejemplo las librerías, que son módulos que ofertan servicios al application framework y el Application Framework oferta servicios a las aplicaciones instaladas. Los componentes de una aplicación en el caso de Android son los Activity, Service, Content Provider y Broadcast Receiver, para el caso de iOS son AppDelegate, ViewControlers y Storyboards. Este tipo de servicios se convierten en posibles vectores de ataque, al igual que el almacenamiento de datos en bases de datos o medios externos. Las áreas a considerar se describen en la Figura **4-3**.

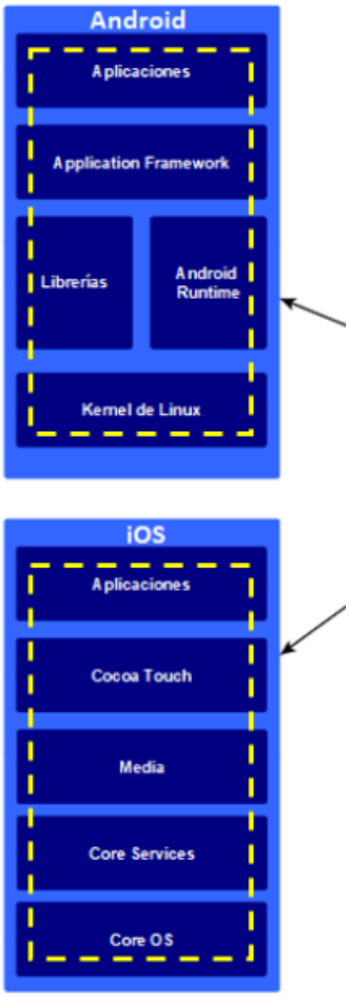

Figura 4-3.: Diagrama de diseño parte 1.
Fuente: Elaboración propia.

Diagrama Parte 2: El segundo escenario corresponde a las conexiones realizadas por la aplicación hacía internet.En este punto se contempla la interceptación de las comunicaciones, el cifrado de las comunicaciones y de los datos en tránsito, los canales de comunicación utilizados, la fortaleza de los protocolos TLS y ciphersuite entre otros. Las áreas a considerar se describen en la **4-4**.

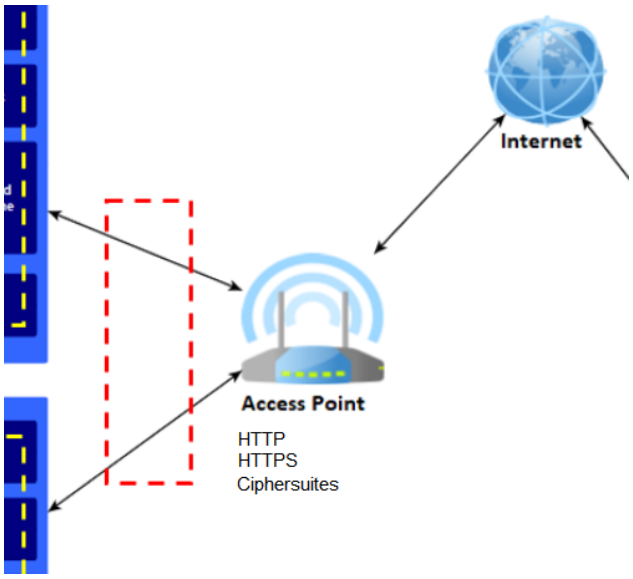

Figura 4-4.: Diagrama de diseño parte 2.
Fuente: Elaboración propia.

Diagrama Parte 3: El tercer escenario corresponde a los servicios expuestos en el backed y los controles que deben adoptar para mantener la lógica de negocio definida desde el frontend. También se contemplan las configuraciones a nivel de cabeceras y el control de acceso requerido para evitar el consumo de servicios de forma arbitraria o acceder a información privada o sensible de los usuarios o incluso información relacionada al negocio o la plataforma. Adicionalmente se debe contemplar el tipo de data almacenada en los logs del servidor entre otros. Las áreas a considerar se describen en la Figura **4-5**.

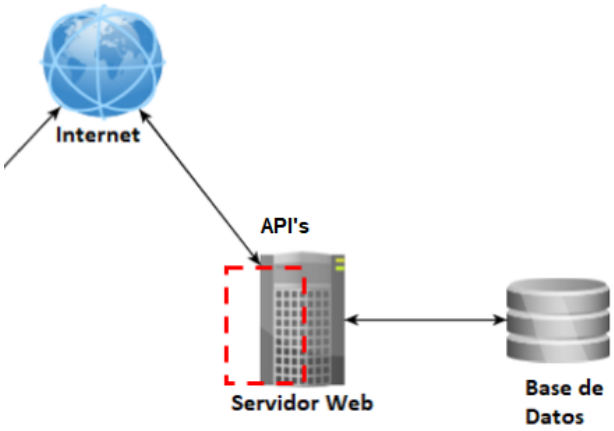

Figura 4-5.: Diagrama de diseño parte 3.
Fuente: Elaboración propia.

Diagrama General: Con el diagrama de diseño definido de forma general, se logra identificar las diferentes áreas o superficies de ataque a tener en cuenta al momento de realizar el modelado de amenazas. El diagrama de diseño con el flujo de datos se puede observar en la Figura **4-6**.

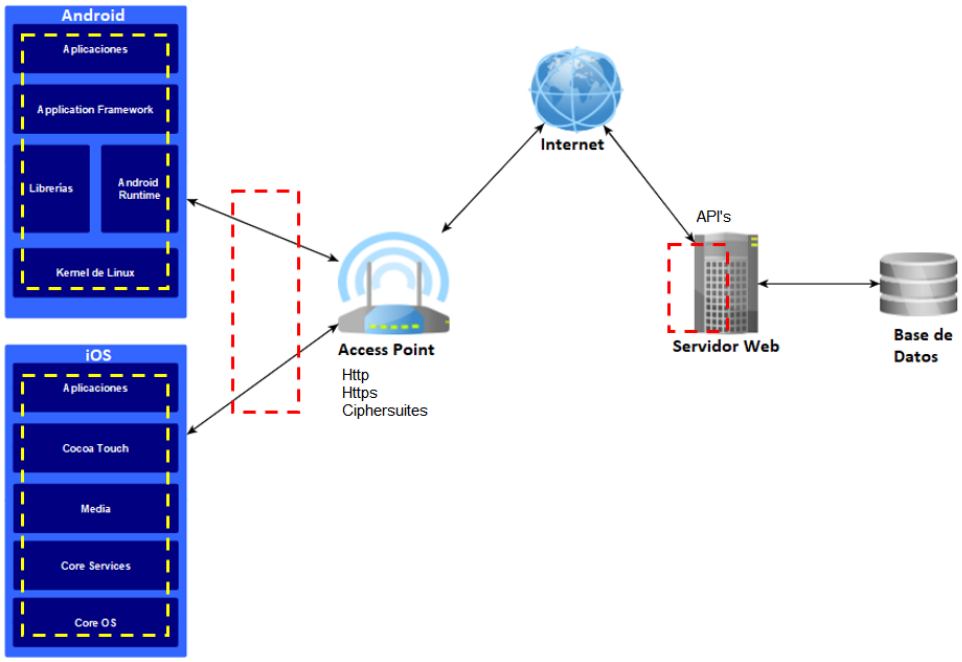

Figura 4-6.: Diagrama de diseño.
Fuente: Elaboración propia.

- **Datos móviles:** se describe en un contexto general, qué datos almacena y procesa la aplicación y posibles flujos de trabajo de datos enfocado a datos personales y de salud, teniendo en cuenta la clasificación de la información según los indica la Guía para la clasificación de la información de acuerdo con sus niveles de seguridad, expedida por la Presidencia de la República en marzo de 2017 [71].

  La clasificación de la información para el alcance de la presente tesis, se encuentra categorizada como Información Pública Clasificada, dividida en Información Privada e Información Semiprivada.

  Información pública clasificada: Según la Ley 1712 de 2014, artículo 6: "Es aquella información que estando en poder o custodia de un sujeto obligado en su calidad de tal, pertenece al ámbito propio, particular y privado o semi-privado de una persona natural o jurídica por lo que su acceso podrá ser negado o exceptuado, siempre que se trate de las circunstancias legítimas y necesarias y los derechos particulares o privados consagrados en el artículo 18 de esta ley" [72].

  En la tabla **4-6** se observa en que lugar se encasillan los datos que son gestionados por las aplicaciones móviles relacionadas al sector salud, con base a la clasificación de la

información descrita en la Guía para la clasificación de la información de acuerdo con sus niveles de seguridad.

CLASIFIACIÓN			EJEMPLO DE INFORMACIÓN	EJEMPLO DE POSIBLES IMPACTOS
INFORMACIÓN PÚBLICA	Información Pública Clasificada	Privada	Dictámenes médicos, reserva médica, reserva bancaria, reserva legal, identidad de menores involucrados en hechos delictivos o en situación de desplazamiento, la información exceptuada por daño de derechos a personas naturales. Datos personales sensibles que afectan la intimidad del titular o cuyo uso indebido puede generar discriminación, tales como aquellos que revelan el origen racial o étnico, la orientación política, las convicciones religiosas o filosóficas, la pertenencia a sindicatos, a organizaciones sociales, de derechos humanos o que promuevan intereses de cualquier partido político o que garantice los derechos y garantías de partidos políticos de oposición, así como los datos relativos a la salud, la vida sexual y los datos biométricos. Secretos comerciales, industriales y profesionales, así como los estipulados en el parágrafo del artículo 77 de la Ley 1474 de 2011. Información de la intimidad de las personas (bajo las limitaciones propias que impone la condición de servidor público). Información que pueda afectar la vida, la salud o la seguridad de una persona Los Datos personales de las Hojas de Vida que no son públicos tales como dirección, teléfono, grupo familiar etc.	El uso indebido puede generar discriminación. Lesiones serias o irreparables. - Perturbaciones en la función misional en caso de no estar disponible. - Acciones administrativas en contra de la entidad. - El uso indebido puede generar discriminación. - Pérdida de la confianza en el gobierno. - El uso indebido puede generar discriminación
		Semi-Privada	Toda información que pertenece al ámbito propio, particular y semiprivado de una persona natural. Borradores de solicitud de cotización. Planes operacionales. Información operativa del DAPRE (Turnos de vigilancia, detalles de configuración de equipos, ubicación de áreas protegidas). Documentos que contengan información personal (número de cuenta de ahorros, dirección de residencia, teléfonos personales, datos de núcleo familiar).	

Tabla 4-6.: Clasificación de la información.
Fuente: Elaboración propia.

Con el fin de identificar que datos procesa cada una de las funciones que normalmente contiene las aplicaciones móviles del sector salud, se realiza un mapeo general de las funciones con los datos y como está clasificada la información. Es de aclarar que la siguiente propuesta es general y cada organización la puede adecuar según la información que la aplicación móvil almacena, transmite o procesa. En las tablas **4-7** y **4-8** se describe el resultado obtenido:

- **Identificar agentes de amenaza y posibles ataques:** En este numeral se describe en un contexto general, cuáles son las amenazas para la aplicación móvil y quiénes son

Funciones	Datos	Clasificación
Login	Número de cédula	Privada
	Nombre completo de cotizante	Privada
Recordar contraseña	Número de cédula o tarjeta de identidad	Privada
	Correo electrónico	Privada
Registro de usuario	Número de cédula	Privada
	Correo electrónico	Privada
	Nombre completo de cotizante	Privada
	Fecha de nacimiento del cotizante	Privada
	Número de teléfono y celular	Privada
	Dirección de residencia	Privada
Consultar y crear citas médicas	Número de cédula o tarjeta de identidad	Privada
	Nombre completo de cotizante y afiliados	Privada
	Descripción de consultas médicas	Privada
	Detalles cita (Lugar, fecha, hora)	Privada
Descarga autorizaciones	Número de cédula o tarjeta de identidad	Privada
	Nombre completo de cotizante y afiliados	Privada
	Fecha de nacimientos del cotizante y afiliados	Privada
	Descripción autorizaciones	Privada
Consultar resultados de laboratorio o procedimientos médicos	Nombre completo de cotizante y afiliados	Privada
	Fecha de nacimientos del cotizante y afiliados	Privada
	Resultados de laboratorio general	Privada
	Número de cédula o tarjeta de identidad	Privada

Tabla 4-7.: Clasificación de los datos vs Funcionalidades. Parte 1.
Fuente: Elaboración propia.

Funciones	Datos	Clasificación
Opción de recordar usuario y clave	Número de cédula	Privada
	Correo electrónico	Privada
Consultar estado de afiliación	Datos y estado de afiliación	Privada
	Número de cédula o tarjeta de identidad	Privada
	Nombre completo de cotizante y afiliados	Privada
	Fecha de nacimientos del cotizante y afiliados	Privada
Consultar información de afiliado	Número de cédula o tarjeta de identidad	Privada
	Nombre completo de cotizante y afiliados	Privada
	Correo electrónico	Privada
	Fecha de nacimiento	Privada
	Dirección de residencia	Privada
Descarga de archivos en pdf	Nombre completo de cotizante y afiliados	Privada
	Fecha de nacimientos del cotizante y afiliados	Privada
	Resultados de laboratorio en detalle	Privada
	Número de cédula o tarjeta de identidad	Privada
Consulta y descarga de certificados	Nombre completo de cotizante y afiliados	Privada
	Número de cédula o tarjeta de identidad	Privada
	Datos y estado de afiliación	Privada

Tabla 4-8.: Clasificación de los datos vs Funcionalidades. Parte 2.
Fuente: Elaboración propia.

los agentes de amenaza que participan.

Se toma como referencia los agentes de amenazas según clasificación descrita por App-Sec Application security, clasificadas principalmente como Usuario malicioso y Malware en diferentes escenarios [73]:

- Usuario malintencionado atacando la aplicación cliente: Bajo este escenario, el

atacante se enfoca principalmente en la aplicación y no precisamente en los servicios que consume en el backend. El usuario malintencionado puede ser un usuario legítimo en la aplicación o un atacante con acceso físico al dispositivo. Un esquema para este escenario se observa en la Figura **4-7**.

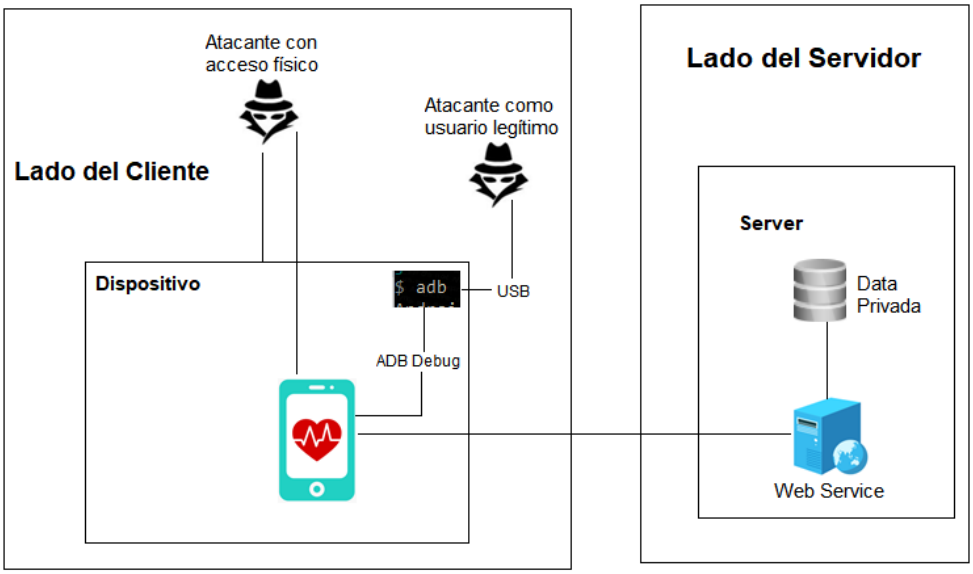

Figura 4-7.: Ataque dirigido a la aplicación móvil - usuario.
Fuente: Elaboración propia.

- Usuario malintencionado que utiliza la aplicación cliente para atacar el servidor: Bajo este escenario un atacante que es legítimo para consumir los servicios desde la aplicación, podrá realizar inyecciones de código o modificación de la mensajería con el fin de comprometer el servidor o alterar la lógica de negocio. Un esquema para este escenario se observa en la Figura **4-8**.

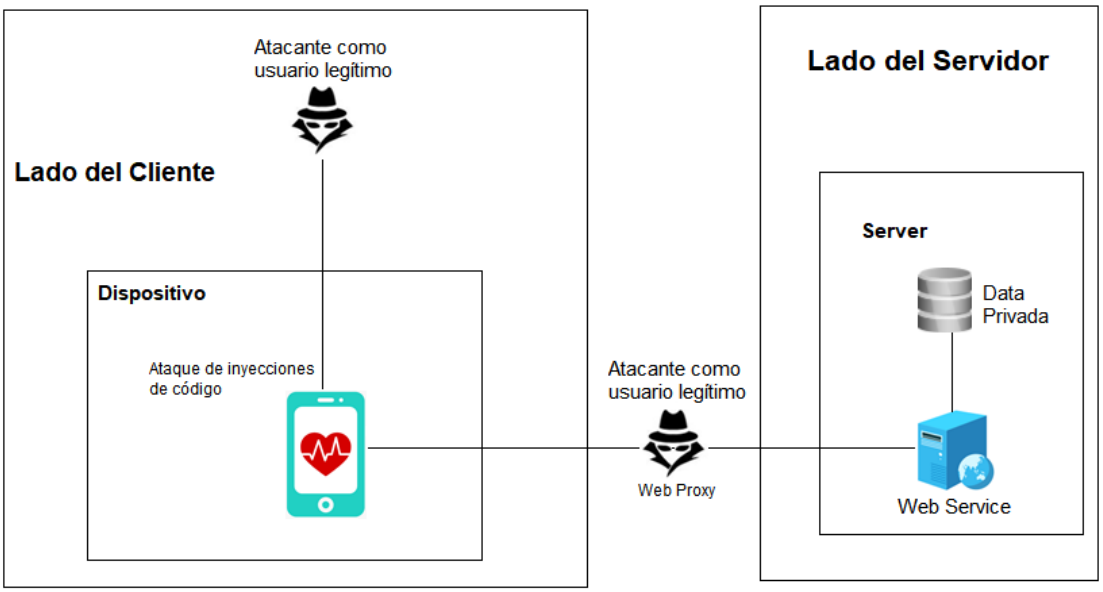

Figura 4-8.: Ataque dirigido a los servicios. - usuario.
Fuente: Elaboración propia.

- Aplicación maliciosa que ataca al usuario final: Bajo este escenario un malware puede obtener las actividades realizadas por el usuario como lo que escribe, correos, llamadas historia de navegación entre otros. También puede llegar a obtener acceso y privilegio a ciertas funciones de la aplicación como es el envío de mensajes o llamadas telefónicas. Otro vector está relacionada a la suplantación de aplicaciones o funcionalidades particulares como es el caso de la pantalla de ingreso del segundo factor de autenticación. Un esquema para este escenario se observa en la Figura **4-9**.

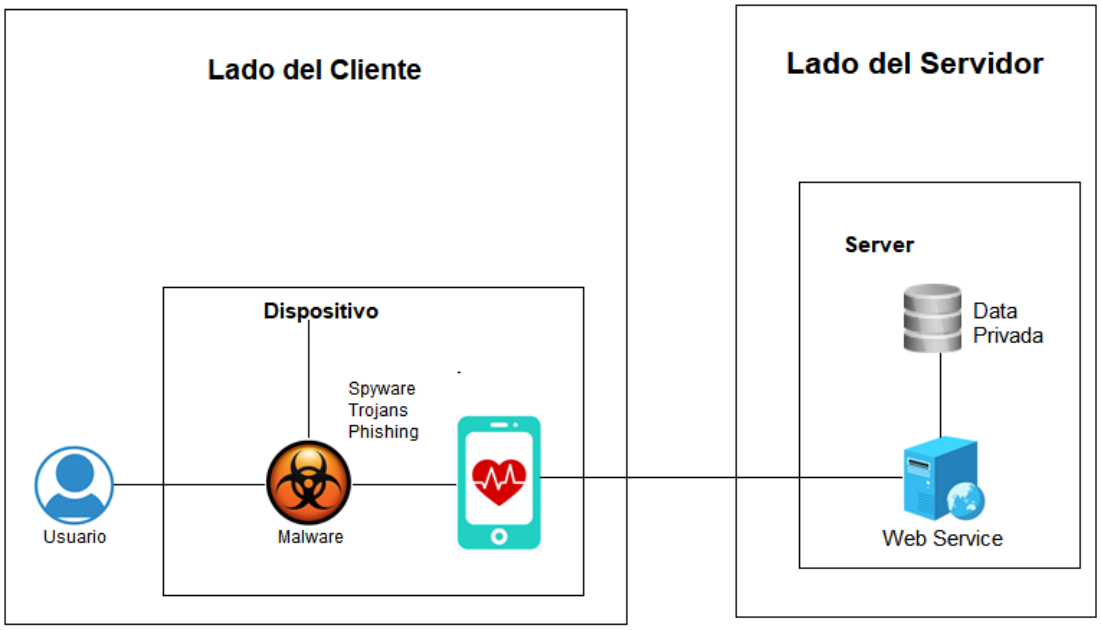

Figura 4-9.: Ataque malware a usuario final.
Fuente: Elaboración propia.

- Aplicación maliciosa que ataca otras aplicaciones en el mismo dispositivo: Bajo este escenario un malware ataca directamente los procesos internos de la aplicación con el fin de obtener información sensible, mediante el aprovechamiento de vulnerabilidades en la aplicación o del mismo sistema operativo, que le permita accederá la sandbox de la aplicación o realizar inyecciones sql a las bases de datos propias de la aplicación. Un esquema para este escenario se observa en la Figura **4-10**.

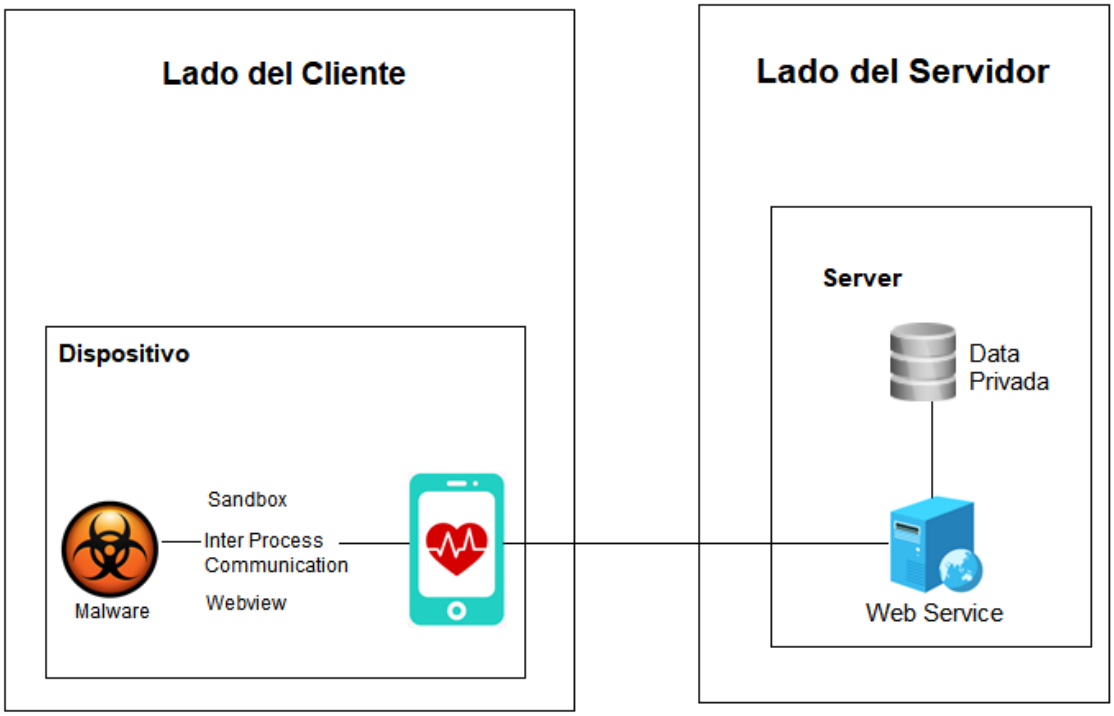

Figura 4-10.: Ataque malware contra aplicación móvil.
Fuente: Elaboración propia.

- **Escenario de amenaza:** Se describe en un contexto general, ¿cuáles son los escenarios más comunes utilizados por los agentes de amenaza?. Esta área define dichos escenarios para que se puedan alinear con los requerimientos de seguridad definidos para proteger la confidencialidad de la información. Por lo anterior, se define el escenario de amenaza por cada uno de los dominios definidos previamente como lineamientos de seguridad. El resultado se observa en las siguientes tablas **4-9**,**4-10**,**4-11**,**4-12**,**4-13**,**4-14**,**4-15**,**4-16**,**4-17**.

Item	**Escenarios de amenaza - Backend**
1	Un atacante logra acceder a información sensible o privada del paciente al lograr obtener las cookies de sesión de la aplicación basado en un ataque de inyección de javascript XSS.
2	Un atacante logra acceder a información sensible o privada del paciente al capturar tráfico basado en interceptación de las comunicaciones (Man in The Middle).
3	Un atacante logra acceder a información sensible o privada del paciente, por medio de ataques del tipo clickjacking.
4	Un atacante logra acceder a información sensible o privada del paciente, al realizar un MIME-sniffing para enviar un ataque del tipo XSS.
5	Un atacante logra acceder a información sensible o privada del paciente, al inyectar código javascript en el contenido de la aplicación, para el cargue de páginas o contenido desde dominios de terceros con el fin de aprovechar por ejemplo ataques del tipo XSS o inyectar payloads que comprometan el servidor.
6	Un atacante logra determinar el tipo y versiones de componentes o webservers que forman parte de la arquitectura de la aplicación, con el fin de enfocar un ataque mas efectivo y acceder a información sensible o privada del paciente.
7	Un atacante logra acceder a información sensible o privada del paciente, mediante la exploración de directorios y consumo directo de los servicios en el Backend, como usuario anónimo o autenticado.
8	Un atacante logra acceder a información sensible o privada del paciente, al lograr inyectar en el Backend código javascript, sentencias sql, comandos de sistema operativo, LDAP o caracteres que no son permitidos según su lógica de negocio, definidas normalmente en el frontend.
9	Un atacante interno o externo logra obtener información sensible o privada del paciente, al acceder a los web logs registrados en el Backend, el cual fue comprometido previamente.

Tabla 4-9.: Escenarios de amenaza - Backend.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Almacenamiento de datos y la Privacidad**
1	Un atacante logra acceder a información sensible o privada del paciente, al lograr capturar en código, o en archivos de configuración de la aplicación, claves de autenticación o llaves de cifrado y descifrado de información sensible.
2	Un atacante logra acceder a información sensible o privada del paciente o de la aplicación, al monitorear los logs registrados en el dispositivo móvil.
3	Un atacante logra acceder a información sensible o privada del paciente, al estar expuesta por métodos HTTP inseguros al momento de interceptar las comunicaciones.
4	Un atacante logra acceder a información sensible o privada del paciente, al ser compartida y transmitida por medio de librerías de terceros como parte funcional de la aplicación.
5	Un atacante logra acceder a información sensible o privada del paciente, mediante el análisis de la caché generada por la aplicación en la sandbox.
6	Una aplicación mal intencionada logra capturar información sensible o privada del paciente gestionada por la aplicación, mediante la captura de datos guardada en el portapapeles.
7	Un atacante o malware logra acceder a información sensible o privada del paciente, que es almacenada en base de datos ubicada en la sandbox de la aplicación.
8	Un atacante logra obtener información sensible o privada del paciente por medio de ataques del tipo shoulder surfing o por un malware que accede a capturas de pantallas de la aplicación.
9	Un atacante logra acceder a información sensible o privada del paciente, al extraer del dispositivo un backup de la aplicación o acceder a backups existentes en otras unidades de almacenamiento.
10	Un atacante o malware logra acceder a información sensible o privada del paciente, que es almacenada en memoria del dispositivo.
11	Un atacante logra acceder fácilmente a las aplicaciones instaladas en un dispositivo robado o extraviado, debido a la no configuración de un código de acceso en el móvil.
12	Un atacante o un malware logra acceder a información sensible o privada del paciente, al lograr extraer archivos o base datos de la sandbox de la aplicación, en móviles que no se encuentran con jailbreak o rooted, debido a una asignación errónea de permisos a los archivos.
13	Un atacante logra obtener información financiera del paciente, en flujos de la aplicación que permiten realizar pagos, en canales o servicios que proveen terceros.

Tabla 4-10.: Escenarios de amenaza - Almacenamiento de datos y la privacidad.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Criptografía**
1	Un atacante logra acceder a información sensible o privada del paciente, en transito entre el móvil y el Backend o almacenada en el dispositivo sin cifrar.
2	Un atacante logra acceder a información sensible o privada del paciente que se encuentra cifrada, utilizando las llaves de descifrado quemadas en el código, expuestas en la mensajería , o almacenadas de forma no segura.
3	Un atacante logra acceder a información sensible o privada del paciente que se encuentra cifrada, explotando vulnerabilidades en las funciones criptográficas.
4	Un atacante logra obtener información sensible o privada del paciente, debido a que la llave criptográfica comprometida es utilizada para varios propósitos o en común para todos los usuarios.

Tabla 4-11.: Escenarios de amenaza - Criptografía.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Autenticación**
1	Un tacante logra obtener información sensible o privada del paciente, al lograr consumir servicios de forma anónima.
2	Un atacante logra acceder a la aplicación y consumir los servicios del paciente, suplantando los identificadores del dispositivo (deviceID), el cual es utilizado como mecanismo de autenticación por parte del Backend. El acceso también se puede dar por robo o pérdida del móvil.
3	Un atacante logra acceder a la aplicación y consumir los servicios del paciente, manipulando la respuesta del servidor en el flujo de autenticación, mediante la inyección de una respuesta válida de login, con el fin de engañar a la aplicación.
4	Un atacante logra acceder a la aplicación y consumir los servicios del paciente, al lograr adivinar fácilmente la contraseña de la víctima, utilizando un listado de contraseñas mas comunes. O generar una denegación de servicios masivo en el intento.
5	Un atacante por medio de diferentes técnicas, logra obtener un listado de usuarios existentes para el servicio, con el fin de potencializar otros ataques y materializar una suplantación de identidad.
6	Un atacante logra capturar las credenciales de acceso de la víctima, al realizar una interceptación de las comunicaciones o al encontrarse almacenadas de forma insegura en el dispositivo.
7	Un atacante con las credenciales de acceso de la víctima ya capturadas, logra realizar transacciones de riesgo desde la aplicación, obteniendo información sensible o privada del paciente.
8	Un atacante logra acceder a información sensible o privada del paciente, al acceder a servicios protegidos con un segundo factor de autenticación, reutilizando un One Time Password (OTP) que tiene un tiempo de vida muy alto.
9	Un atacante logra acceder a información sensible o privada del paciente, al acceder a servicios protegidos con un segundo factor de autenticación, adivinando el OTP por medio de un ataque de fuerza bruta.
10	Un atacante logra obtener o cambiar las credenciales de acceso a la aplicación, aprovechando debilidades en el mecanismo de recordar usuario o contraseña.

Tabla 4-12.: Escenarios de amenaza - Autenticación.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Manejo de Sesiones y autorización**
1	Un atacante logra acceder a información sensible o privada de otros pacientes, al lograr manipular las referencias a un objeto en la aplicación, para acceder a otros objetos sin autorización. Por ejemplo al realizar consultas, descargas de archivos o acceso a recursos privilegiados.
2	Un atacante logra acceder a información sensible o privada del paciente, al consumir servicios con sesiones entregadas por la aplicación sin autenticar.
3	Un atacante logra acceder a información sensible o privada del paciente, al lograr predecir el token de sesión y por ende acceder a los servicios de la aplicación.
4	Un atacante logra acceder a información sensible o privada del paciente, debido a que la aplicación no utiliza una sesión y envía las credenciales de acceso en cada petición.
5	Un atacante logra obtener información confidencial del paciente, almacenada en el Json Web Token.
6	Un atacante logra acceder a información sensible o privada del paciente, mediante la alteración del los valores contenidos en el Json Web Token, al utilizar algoritmos inseguros en la firma.
7	Un atacante logra acceder a información sensible o privada del paciente, mediante la alteración del los valores contenidos en el Json Web Token, al lograr obtener el "secret" de la firma por medio de un ataque de fuerza bruta o diccionario.
8	Un atacante logra acceder al sistema y obtener información sensible o privada del paciente, al robar un token mediante la interceptación de las comunicaciones y lo utiliza para suplantar al usuario legítimo.
9	Un atacante logra acceder a información sensible o privada del paciente, al consumir servicios con una sesión previamente cerrada por la víctima desde la aplicación.
10	Un atacante logra acceder a información sensible o privada del paciente, al lograr acceder a la aplicación, mediante la extracción del token almacenado en base de datos sqlite.
11	Un atacante logra acceder a información confidencial o privada del paciente, al secuestrar la sesión o ingresar de forma simultanea a la aplicación.

Tabla 4-13.: Escenarios de amenaza - Manejo de sesiones y autorización.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Comunicación a través de la red**
1	Un atacante logra acceder a información sensible o privada del paciente, al capturar la información transmitida entre la aplicación y el servidor sin cifrar.
2	Un atacante logra acceder a información confidencial y privada capturando información transmitida entre la aplicación y el servidor por medio de la degradación de los ciphersuite o de los protocolos de comunicación seguros.
3	Un atacante logra acceder a información confidencial o privada del paciente, capturando información transmitida entre la aplicación y el servidor, utilizando certificados que no son confiables (autofirmados).
4	Un atacante logra acceder a información confidencial o privada del paciente, capturando información transmitida entre la aplicación y el servidor, utilizando certificados que son de confianza pero no corresponden al suministrado por el backend.
5	Un atacante logra acceder a la aplicación aprovechando funciones para recuperar el usuarios/contraseña o realizar operaciones críticas protegidas por un segundo factor de autenticación, al suplantar la SIMCARD de la víctima, para obtener el OTP y continuar con el flujo. (SIM SWAP Attack).
6	Un atacante logra acceder a información sensible o privada del paciente, accediendo a servicios protegidos con un segundo factor de autenticación, mediante la extracción del mensaje de voz que deja el mecanismo de llamada telefónica para la entrega del OTP, al momento de no responder la llamada.
7	Un atacante logra acceder a información sensible o privada del paciente, al explotar vulnerabilidades encontradas en librerías de terceros (Heartbleed, ShellShock).

Tabla 4-14.: Escenarios de amenaza - Comunicación a través de la red.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Interacción con la Plataforma**
1	Un atacante logra acceder a información sensible o privada del paciente, basado en una inadecuada o innecesaria asignación de permisos en la aplicación, hacia los recursos del sistema operativo.
2	Un atacante logra acceder a información sensible o privada del paciente, al momento de realizar una interceptación de las comunicaciones internas IPC de la app, para capturar información o inyectar valores para engañar a la víctima.
3	Un atacante logra acceder a información sensible o privada del paciente, al lograr acceder a información sensible almacenada en la sandbox de la aplicación, mediante la inyección de java script en los webview.

Tabla 4-15.: Escenarios de amenaza - Interacción con la plataforma.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Calidad de Código y Configuración del Compilador**
1	Un atacante logra acceder a información sensible o privada del paciente, mediante la inyección de un payload malicioso en la aplicación, teniendo en cuenta que solo fue firmada con la versión 1 en Android. (janus).
2	Un atacante logra acceder a información sensible o privada del paciente o de la lógica del negocio, al firmar la aplicación con algoritmos considerados débiles o vulnerables.
3	Un atacante logra obtener información relacionada a la lógica funcional de la aplicación, al encontrar en el código fuente, funciones y métodos con nombres descriptivos, útiles para planear el bypass de un control de seguridad.
4	Un atacante logra acceder a información sensible o privada del paciente, mediante la modificación de valores en las variables utilizadas en ciertas funciones de la aplicación, en modo ejecución y sin modificar el binario.
5	Un atacante logra acceder a información sensible o privada del paciente, al tener acceso a los mensajes de debug expuestos por la aplicación, que no se desactivaron en el modo release.
6	Un atacante logra acceder a información sensible o privada del paciente o de la plataforma, al visualizar los mensajes de las diferentes excepciones no capturadas y tratadas por la aplicación.
7	Un atacante logra acceder a información sensible o privada del paciente, al lograr el acceso a funciones sin la debida autorización, al generarse un error en los controles de seguridad.
8	Un atacante logra acceder a información sensible o privada del paciente, al lograr inyecciones del tipo sentencias sql, Esquemas URL personalizados, códigos QR o llamadas IPC.

Tabla 4-16.: Escenarios de amenaza - Calidad de código y configuración del compilador. Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Impedir el Análisis Dinámico y la Manipulación**
1	Un atacante logra acceder a información sensible o privada del paciente, evadiendo los controles de seguridad de acceso y autorización, mediante la instalación de herramientas de análisis dinámico de la aplicación y el acceso a recursos exclusivos para usuarios root.
2	Un atacante logra acceder a información sensible o privada del paciente, activando intencionalmente el modo debug de la aplicación, alterando su código fuente.
3	Un atacante logra acceder a información sensible o privada del paciente, evadiendo los controles de seguridad de acceso y autorización, mediante la manipulación del código fuente del binario.
4	Un atacante logra acceder a información sensible o privada del paciente, evadiendo los controles de seguridad basado en el análisis y manipulación de la lógica funcional, expuesta en el código fuente de la aplicación.

Tabla 4-17.: Escenarios de amenaza - Impedir el análisis dinámico y la manipulación. Fuente: Elaboración propia.

- **Requerimientos de seguridad:** Para completar el modelado de amenazas, se realizó un mapeo de los requerimientos de seguridad previamente identificados con los diferentes escenarios de amenaza y el Top 10 de OWASP [31]. Como resultado final se obtiene un modelado de amenazas genérico representado en un cuadro conformado por los siguientes 12 campos y que se encuentra en detalle en el Anexo H:
  - Ref No:
  - Elementos
  - Interación
  - Escenario de amenaza
  - **S**poofing
  - **T**ampering
  - **R**epudiation
  - **I**noformation Disclure
  - **D**enial of service
  - **E**levation of privilege

- Contramedidas
- OWASP Top Ten

5. Definición de pruebas especializadas de seguridad informática

Con el modelado de amenazas previamente definido y el listado de lineamientos de seguridad consolidados, se procede a definir el conjunto de pruebas dinámicas especializadas en seguridad informática a ejecutar, describiendo al detalle cada una de las pruebas técnicas a realizar para cada uno de los lineamientos definidos en los dominios Backend (Servicios), Almacenamiento de datos y la Privacidad, Criptografía, Manejo de Sesiones y autorización, Comunicación a través de la red, Interacción con la Plataforma, Calidad de Código y Configuración del Compilador, Impedir el Análisis Dinámico y la Manipulación. Para esta actividad se tendrá en cuenta los diferentes trabajos adelantados en la literatura, estándares o guías relacionadas a pruebas de seguridad enfocadas a las arquitecturas móviles, en pro de la protección de la confidencialidad de la información.

La descripción de las pruebas especializadas de seguridad serán orientadas en las pruebas del tipo caja negra, es decir bajo el contexto de un ciberatacante, principalmente con un enfoque de pruebas dinámicas sin descartar algunas pruebas estáticas relacionadas al binario.

El proceso a seguir para la validación del cumplimiento de los lineamientos de seguridad propuestos en el documento, nace de una forma de trabajo que se realiza en la empresa donde laboro actualmente con 5 Ethical Hackers, realizando pruebas de seguridad en aplicaciones móviles y web del sector financiero. Por esta razón se propone dicho proceso teniendo en cuenta los buenos resultados obtenidos en las pruebas. El proceso se puede observar mediante el diagrama expuesto en la Figura **5-1**.

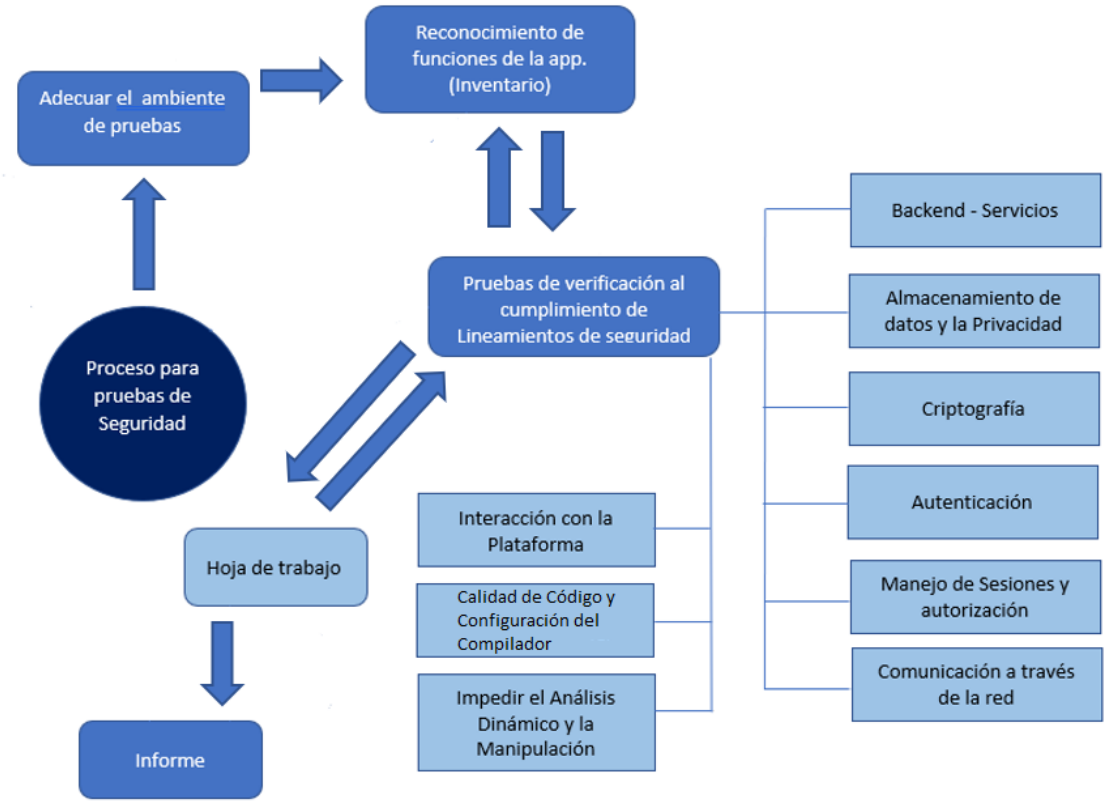

Figura 5-1.: Diagrama proceso para pruebas de seguridad.
Fuente: Elaboración propia.

5.1. Adecuar el ambiente de pruebas

Con el fin de realizar las pruebas de seguridad del tipo dinámicas a las aplicaciones móviles, se debe adecuar un ambiente de pruebas que permita interceptar tráfico, acceder a rutas internas del sistema operativo entre otras. Se recomienda las siguientes dos arquitecturas para montar el ambiente de pruebas:

5.1.1. Con dispositivo móvil físico

En la Figura **5-2** se observa un esquema que permite realizar interceptaciones de las comunicaciones con el fin de modificar el tráfico e interactuar de diferentes formas con el dispositivo móvil y de esta manera contar con lo necesario para lograr realizar las respectivas validaciones de seguridad.

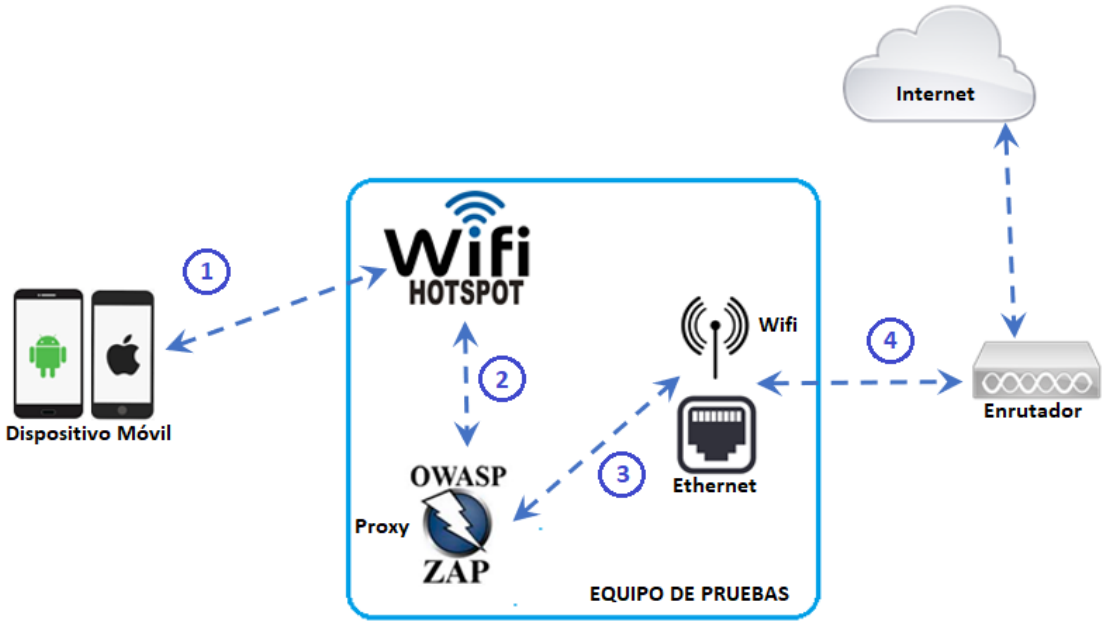

Figura 5-2.: Arquitectura con dispositivo móvil físico.
Fuente: Elaboración propia.

La descripción de cada uno de los puntos de la arquitectura de conexión para las pruebas de seguridad se detallan en el Anexo I.

5.1.2. Con sistema operativo Android en máquina virtual:

En la Figura **5-3** se observa un esquema que permite realizar interceptaciones de las comunicaciones, utilizando una máquina virtual Android en lugar de un dispositivo móvil físico, con el fin de modificar el tráfico e interactuar de diferentes formas con el sistema operativo y de esta manera contar con lo necesario para lograr realizar las respectivas validaciones de seguridad. La ventaja principalmente consiste en disponer de un ambiente muy cercano a la realidad sin la necesidad de invertir en en un dispositivo móvil físico y en aplicar técnicas de rooted. Este escenario solo aplica para Android debido a que Apple no dispone de imágenes de sistemas operativos iOS para ser cargadas en máquinas virtuales.

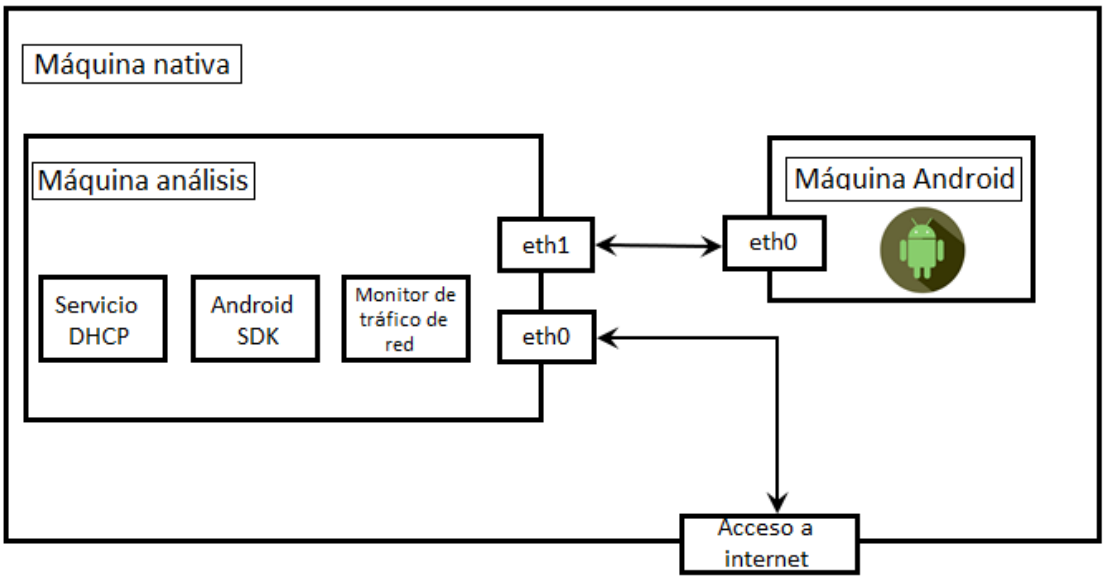

Figura 5-3.: Arquitectura con Maquina virtual Android.
Fuente: Elaboración propia.

La arquitectura está conformada por una *máquina nativa* que contiene dos (2) máquinas virtuales, una de ellas corresponde a la *máquina análisis* y la otra corresponde a la *máquina Android*. El esquema se adopta de un modelo utilizado para realizar el análisis de malware en dispositivos móviles Android [74]. En el Anexo I se describe de forma general cada uno de los componentes.

5.2. Reconocimiento de la APP:

En pruebas de caja blanca, el pentester contará con información de primera mano sobre las diferentes funcionalidades con las que cuenta la aplicación, los diferentes servicios que consume, el tipo de controles de seguridad que tiene implementado, el tipo de infraestructura utilizada e incluso el código fuente. Esta información es de gran importancia antes de iniciar las pruebas de seguridad en una aplicación móvil. En las pruebas del tipo caja negra, se debe realizar el levantamiento de la información por medio del uso de la aplicación como un usuario normal [42]. Por lo anterior en la Figura **5-4**, se sugieren cuatro (4) vectores generales a revisar antes de iniciar las pruebas de seguridad.

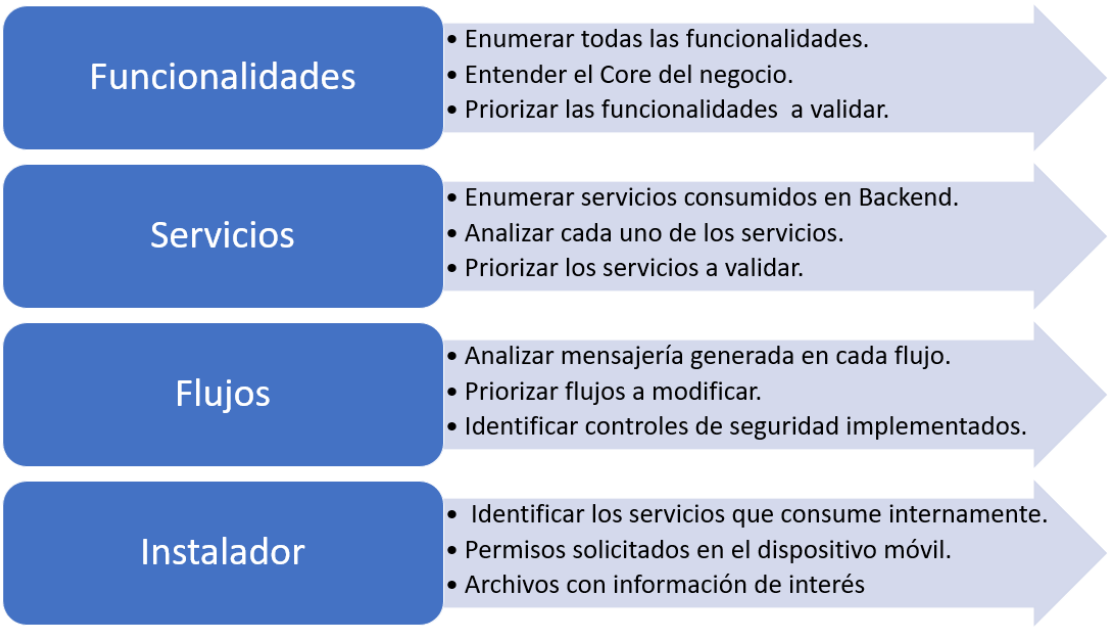

Figura 5-4.: Esquema reconocimiento de funciones de la aplicación móvil.
Fuente: Elaboración propia.

Basado en el esquema expuesto en la Figura **5-4**, antes de iniciar las pruebas de seguridad se debe realizar un levantamiento de información lo mas completa posible y una de las formas para lograrlo en un tipo de pruebas de caja negra es ser un usuario más de la aplicación móvil. El ser un usuario de la aplicación a validar permite recorrer toda las funcionalidades que la contiene, permitiendo enumerar cada una de las funciones que serán validadas, identificar y enumerar la información que gestiona la aplicación (datos personales, de salud y financieros) basado en el core del negocio y priorizar el orden de validación. El siguiente punto corresponde a los servicios, se debe analizar todo el tráfico generado por la aplicación, con el fin de identificar y enumerar los diferentes servicios que consume la aplicación contra el Backend, analizar cada uno de los servicios en busca de información relevante a la plataforma o tecnologías utilizadas y priorizar las respectivas validaciones de seguridad. Posteriormente se procede a realizar el análisis de la mensajería en cada uno de los flujos funcionales de la aplicación con el fin de identificar las reglas de negocio, controles de seguridad a nivel frontend y los que se puedan identificar a nivel de backend como por ejemplo un segundo factor de autenticación, cifrado de la mensajería entre otros, de igual forma se deben priorizar los flujos que generen mayor interés para las pruebas. Por último se debe analizar el instalador, que en cierta forma se convertirán en una prueba estática pero bajo un enfoque diferente. Estas pruebas que incluso se puede realizar con herramientas automatizadas, permiten identificar los servicios que consume la aplicación con el sistema operativo, con el fin de realizar inyecciones a bases de datos locales, consumir directamente funciones

de la aplicación sin autorización, abusar de malas configuraciones en los permisos otorgados a la aplicación, entre otro tipo de vectores que forman parte de las pruebas dinámicas.

Con esta información se podrá establecer las pruebas a realizar basado en el modelado de amenazas propuesto anteriormente, facilitando la identificación de los riesgos de seguridad a los que puede estar expuesta la aplicación y que controles deben ser evaluados.

5.3. Pruebas de verificación

La siguiente sección tiene como fin realizar una descripción de como validar cada uno de los lineamientos propuestos en capítulos anteriores, con el fin de determinar su cumplimiento y posibles riesgos de seguridad a los que puede estar expuesta una aplicación móvil, por medio de ataques y técnicas de intrusión utilizadas hoy en día tanto por Ciberdelincuentes como por Hackers Éticos. También se describirá el uso de algunas herramientas que pueden ser utilizadas para dicho propósito, aclarando que no son las únicas. El alcance contempla los lineamientos definidos como:

- L1. Backend - Servicios.
- L2. Almacenamiento de datos y la Privacidad.
- L3. Criptografía.
- L4. Autenticación.
- L5. Manejo de sesiones y autorización.
- L6. Comunicación a través de la red.
- L7. Interacción con la Plataforma.
- L8. Calidad de Código y Configuración del Compilador.
- L9. Impedir el Análisis Dinámico y la Manipulación.

A continuación se realiza una descripción general de las pruebas a realizar y el detalle de las mismas junto con las herramientas de apoyo a utilizar, se pueden observar en el Anexo J.

5.3.1. Backend - Servicios

5.3.1.1. Lineamiento L1.1

Control definido: Configurar de forma correcta las cabeceras en el servidor web.
Pruebas de Validación: La validación de este lineamiento consiste en identificar si el servidor responde con la cabeceras requeridas para evitar diferentes tipos de ataques que puede llegar a comprometer la seguridad del cliente final.

5.3.1.2. Lineamiento L1.2

Control definido: Configurar controles que permitan asegurar que los usuarios solo pueden acceder a las URLs para las cuales poseen autorización.
Pruebas de Validación: Con la enumeración de los servicios realizada en la fase de reconocimiento, se debe validar si alguno de los servicios puede ser accedido de forma directa sin ningún tipo de autenticación. El fin de esta validación es lograr consumir servicios en el backend sin utilizar la aplicación móvil.

5.3.1.3. Lineamiento L1.3

Control definido: Las validaciones realizadas en los diferentes puntos de entrada de la aplicación móvil, deben ser igualmente validados de lado del backend, para evitar inyecciones XSS, SQLi, XML, XXE (XML external entity) o LDAP.
Pruebas de Validación: Las aplicaciones suelen tener controles de seguridad a nivel de frontend para evitar posibles inyecciones de código en los diferentes puntos de entrada en la aplicación móvil. En algunos casos, estos controles no son aplicados en la capa de servicios.

5.3.1.4. Lineamiento L1.4

Control definido: Los logs registrados en el servidor no deben contener información considerada sensible o confidencial.
Pruebas de Validación: El foco de la validación consiste en identificar si los servidores que gestionan la información de los usuarios de la aplicación, almacenan datos sensibles en los registros de logs. Este tipo de validaciones se realizan principalmente como parte de las pruebas de seguridad del tipo caja blanca pero también pueden aplicar como pruebas de caja negra teniendo en cuenta que el servidor puede ser comprometido y accedido de forma remota.

5.3.1.5. Lineamiento L1.5

Control definido: Implementar controles que eviten el listado de directorios y archivos en los servidores con los que interactúa la aplicación (Hardening).
Pruebas de Validación: Las pruebas consisten en identificar problemas de hardening en el backend, relacionada a los diferentes servicios web expuestos para el consumo de la aplicación.

5.3.2. Almacenamiento de datos y la Privacidad

5.3.2.1. Lineamiento L2.1

Control definido: Las funcionalidades de almacenamiento de credenciales del sistema son utilizadas para almacenar la información sensible, como credenciales del usuario, Información de identificación personal PII, datos de salud o claves criptográficas.

Pruebas de Validación: Algunos desarrolladores almacenan información de forma persistente en el dispositivo móvil y no utilizan las API de seguridad proporcionadas por el sistema operativo para proteger la información, que puede estar relacionada a los usuarios, claves de cifrado y descifrado de datos entre otras.

5.3.2.2. Lineamiento L2.2

Control definido: No se escribe información sensible en los registros de la aplicación en el sistema operativo.
Pruebas de Validación: Normalmente los desarrolladores activan el registro de logs en la aplicación, con el fin de realizar seguimiento a posibles fallos o errores generados durante el desarrollo. En ocasiones las aplicaciones salen a producción con los logs activos, generando el registro de información relacionada a datos personales o de salud del paciente, que puede ser accedida por cualquier aplicación.

5.3.2.3. Lineamiento L2.3

Control definido: Para aplicaciones Híbridas o Web APP no se debe exponer información sensible por método GET, se debe enviar solo por Método POST.
Pruebas de Validación: En aplicaciones híbridas, móviles web o web móvil embebida, suelen enviar información privada del usuario en la url, registros de inscripción a la plataforma como es el caso de las credenciales de acceso u otro tipo de información como identificadores de sesión entre otros. Esta información puede ser capturada por un atacante que esté realizando un sniffing de la red, o en históricos propios de la aplicación almacenado de forma local en los servidores (logs webserver).

5.3.2.4. Lineamiento L2.4

Control definido: No se comparte información sensible con servicios externos salvo que sea una necesidad de la arquitectura.
Pruebas de Validación: Con el fin de mejorar la experiencia de usuario, algunas aplicaciones cuentan con librerías o sdk de terceros para realizar monitoreo de navegación o de uso de la aplicación por parte de los usuarios, la agregación de nuevos servicios como por ejemplo la georeferenciación, incluir controles de seguridad, entre otros, que en ocasiones suelen compartir información privada de los usuarios sin ninguna protección.

5.3.2.5. Lineamiento L2.5

Control definido: Se debe desactivar el caché del teclado en los campos de texto donde se maneja información sensible.
Pruebas de Validación: Algunas aplicaciones permiten la sugerencia de ingreso de datos en algunos formularios, como una forma de facilitar la interacción entre el usuario final y

la aplicación. En ocasiones se permite el uso del cache del teclado en campos que contienen información privada, que puede ser accedida por un atacante con acceso físico al dispositivo o por la acción de un malware.

5.3.2.6. Lineamiento L2.6

Control definido: Se debe desactivar el portapapeles en los campos de texto donde se maneja información sensible.
Pruebas de Validación: La opción copiar y pegar es muy utilizada por los usuarios para evitar diligenciar información en algunos campos. Permitir esta opción en campos de entrada con data sensible, abre la posibilidad a que información privada sea capturada por un malware desde el portapapeles. Un ejemplo es la aplicación Clipper en Android.

5.3.2.7. Lineamiento L2.7

Control definido: No exponer información sensible mediante mecanismos entre procesos (IPC).
Pruebas de Validación: El Content Providers tienen como propósito compartir entre aplicaciones, el acceso a diferentes tipos de datos estructurados. Algunas aplicaciones utilizan este servicio de Content Providers para almacenar diferentes tipos de información como cookies, llaves, información personal entre otras bajo el formato de base de datos como sqlite, que pueden llegar a exponer información privada a otras aplicaciones al no estar protegida correctamente.

5.3.2.8. Lineamiento L2.8

Control definido: No exponer información sensible como contraseñas, datos personales o números de tarjetas de crédito a través de la interfaz o capturas de pantalla.
Pruebas de Validación: Las aplicaciones normalmente solicitan información a sus usuarios que puede ser considera privada. La información puede ser expuesta mediante el ingreso de datos en formularios gestionados por la aplicación, como por ejemplo las credenciales de acceso, información financiera por pagos de facturas, entre otras. También puede ser expuesta como resultado a las diferentes consultas que son realizadas contra el backend.

5.3.2.9. Lineamiento L2.9

Control definido: No incluir información sensible en los respaldos generados por el sistema operativo, como es el caso de resultados de laboratorio almacenados en la sandbox de la aplicación en formato pdf.
Pruebas de Validación: Los dispositivos móviles Android y iOS permiten realizar copias

de respaldo automáticas de la información y configuraciones de las aplicaciones. Ésta funcionalidad podría almacenar información sensible que podría ser extraída por un atacante desde el equipo de computo que la almacena, o puede ser utilizada como una opción para acceder a la información que se encuentra almacenada en la sandbox de las aplicaciones, en caso de no lograr el acceso desde el dispositivo móvil.

5.3.2.10. Lineamiento L2.10

Control definido: La aplicación remueve la información sensible de la vista cuando la aplicación pasa a un segundo plano.
Pruebas de Validación: Una funcionalidad particular de los dispositivos móviles consiste en realizar una captura de pantalla cuando las aplicaciones entran en segundo plano. Estas capturas suelen ser almacenadas de forma local en el dispositivo y puede contener información sensible como resultados de laboratorio entre otros, que tienen el riesgo de ser extraídas por aplicaciones maliciosas o por técnicas forenses.

5.3.2.11. Lineamiento L2.11

Control definido: La aplicación no conserva la información sensible en memoria más de lo necesario y la memoria es limpiada luego de su uso.
Pruebas de Validación: Para ciertos tipos de transacciones, las aplicaciones móviles utilizan la memoria del dispositivo para la ejecución de procesos particulares que suelen dejar información sensible que no es eliminada de forma correcta. El inadecuado uso de este recurso abre la puerta a que un atacante por medio de diferentes técnicas logre obtener datos en memoria con información que puede ser de carácter privado del usuario o información sensible de la aplicación como es el caso de las llaves de cifrado.

5.3.2.12. Lineamiento L2.12

Control definido: La aplicación obliga a que exista una política mínima de seguridad en el dispositivo, como por ejemplo configurar un código de acceso.
Pruebas de Validación: Se debe definir un política de seguridad o condiciones mínimas que debe tener el dispositivo móvil para permitir el acceso a la aplicación. Este tipo de controles permite resguardar en cierta forma al usuario final por pérdida del dispositivo y por una posible acción de un atacante o malware.

5.3.2.13. Lineamiento L2.13

Control definido: Asignar permisos apropiados a los archivos y Base de datos gestionados por la aplicación.
Pruebas de Validación: La importancia de contar con los permisos apropiados en los archivos que se encuentran alojados en la sanbox de la aplicación, es evitar que aplicaciones

externas logren extraerlos y por ende obtener información privada del usuario o parámetros de configuración propios de la aplicación que se puede traducir en un posible incidente de seguridad.

5.3.2.14. Lineamiento L2.14

Control definido: No utilizar credenciales quemadas en el código (Hard-coded).
Pruebas de Validación: Considerando que este lineamiento se encuentra bajo la clasificación de pruebas del tipo estáticas, es relevante mencionar el gran beneficio que representa para las pruebas dinámicas, la verificación de código con un alcance específico. El enfoque no es el encontrar posibles XSS o SQli, lo principal es lograr encontrar las llaves de cifrado y descifrado de datos, el password para abrir una base de datos sqlite cifrada, credenciales de acceso para el consumo de servicios de terceros entre otras, que permiten generar nuevos vectores de ataque en las pruebas de seguridad tipo dinámicas en la aplicación.

5.3.2.15. Lineamiento L2.15

Control definido: Cumplimiento de PCI en caso de gestionar números de tarjeta de crédito o débito en algún flujo de la aplicación.
Pruebas de Validación: En algunas de las aplicaciones evaluadas con el fin de identificar que tipo de información intercambia con el usuario, se observó funcionalidades relacionadas a a pagos. Estas funcionalidades cuentan con un registro de Tarjetas de crédito y como requisito de seguridad, el usuario debe ingresar manualmente el código CVC. Por lo anterior, se debe garantizar el cumplimiento a las definiciones de PCI para evitar la exfiltración de datos financieros.

5.3.3. Criptografía

5.3.3.1. Lineamiento L3.1

Control definido: Implementar un mecanismo para proteger la información sensible personal y la relacionada a la salud, con la utilización de mecanismos de cifrado o de seudonimización.
Pruebas de Validación: Tomando como base la Figura **??**, se define como información privada relacionada a datos personales y de salud a: Dictámenes médicos, reserva médica, así como los datos relativos a la salud y datos personales sensibles que afectan la intimidad del titular. Este tipo de información debe ser identificada y protegida adecuadamente, con el fin de evitar que personas no autorizadas logren acceder a dicha información e identificar a quien pertenecen.

5.3.3.2. Lineamiento L3.2

Control definido: La aplicación no depende únicamente de criptografía simétrica, con claves quemadas en el código. Se recomienda el cifrado asimétrico.
Pruebas de Validación: Algunas aplicaciones utiliza funciones criptográficas, con el fin de proteger la información sensible gestionada por la aplicación de posibles accesos no autorizados. Por ejemplo, el cifrado simétrico es utilizado normalmente para cifrar los request y response intercambiados entre la aplicación y el servidor para evitar la exposición de información o alteración de la misma. También es utilizada para cifrar data sensible como usuarios, contraseñas, tokens de sesión entre otros.

5.3.3.3. Lineamiento L3.3

Control definido: La aplicación no utiliza protocolos o algoritmos criptográficos que son considerados deprecados para aspectos de seguridad.
Pruebas de Validación: Se debe tener presente que algunos algoritmos definidos como seguros en el pasado, a hoy ya no lo son. Con el pasar del tiempo los algoritmos tienden a ser inseguros. Como parte de las buenas prácticas descritas por OWASP [75], se debe validar que la aplicación no utiliza algoritmos que son considerados vulnerables o cuentan debilidades que pueden exponer la información que se desea proteger.

5.3.3.4. Lineamiento L3.4

Control definido: La aplicación no reutiliza una misma clave criptográfica para varios propósitos.
Pruebas de Validación: En ocasiones por temas de simplicidad, a nivel de arquitectura utilizan las mismas llaves criptográficas para diferentes propósitos. Esta mala práctica extiende el campo de compromiso en caso de comprometerse las llaves de cifrado. Por ejemplo, si un organización cuenta con mas de dos aplicaciones móviles que proveen servicios diferentes, utilizan las mismas llaves de cifrado para proteger la data y la mensajería intercambiada entre la aplicación móvil y el backend. Por una vulnerabilidad en la aplicación A que expuso las llaves de cifrado, se compromete también el cifrado utilizado en la aplicación B. Otro escenario ocurre cuando se utilizan las mismas llaves de cifrado tanto en ambiente de Desarrollo, QA (Quality Assurance) y Producción. Normalmente bajo lo escenarios de desarrollo y QA, las llaves de cifrado son compartidas y conocidas para un grupo mayor de personas y los ambientes no cuentan con la misma rigurosidad a nivel de implementaciones de seguridad.

5.3.4. Autenticación

5.3.4.1. Lineamiento L4.1

Control definido: Si la aplicación provee acceso a un servicio remoto, un mecanismo aceptable de autenticación como usuario y contraseña es realizado en el servidor remoto.
Pruebas de Validación: El proceso de autenticación en primer nivel debe requerir al menos el ingreso de un usuario y una contraseña que cumpla con las característica de primer factor de autenticación, es decir algo que solo el usuario conoce y deben ser validadas de lado del servidor. En una de las aplicaciones analizadas utilizaban únicamente el número de cédula y la fecha de nacimiento para poder acceder a la aplicación de salud, datos que no son solo de conocimiento del usuario de la aplicación.

5.3.4.2. Lineamiento L4.2

Control definido: Existe una política de contraseñas y es aplicada en el servidor.
Pruebas de Validación: El cumplimiento de una política de contraseñas, permite definir la complejidad con la que el usuario debe crear la contraseña para evitar ser adivinada fácilmente por un ciber-delincuente, mediante el uso de ataques automatizados.

5.3.4.3. Lineamiento L4.3

Control definido: El servidor implementa controles, cuando credenciales de autenticación son ingresadas una cantidad excesiva de veces. (Automatizacion).
Pruebas de Validación: Cuando un atacante dispone de un conjunto de nombres de usuarios (nicknames) válidos en la aplicación, el paso a seguir es intentar adivinar la contraseña por medio del envío de múltiples request de forma automatizada. En este punto la aplicación o controles externos a ella deberán tener la capacidad de detectar y contener este tipo de ataques, conocidos como ataques de fuerza bruta o de diccionario y los controles normalmente utilizados son captchas y la implementación de soluciones WAF que por medio de reglas pueden identificar este tipo de peticiones por IP, número de peticiones vs tiempo entre otras.

5.3.4.4. Lineamiento L4.4

Control definido: El servidor implementa controles cuando se intenta realizar enumeración de usuarios en el login, recuperar contraseña, recuperar usuario o cualquier otro endpoint donde requiera el uso del usuario, número de cédula o identificador de fácil adivinación.
Pruebas de Validación: La enumeración de usuarios tiende a ser viable debido a que la prueba se enfoca en encontrar un usuario válido para la aplicación sin importar la contraseña utilizada. Normalmente los controles son establecidos en números de intentos fallidos de ingreso de contraseña por usuario y no por la cantidad de usuarios que solicitan el acceso en un determinado tiempo desde un mismo punto de origen. El contar con un grupo de usuarios

válidos permite a un atacante fortalecer no solo la validación de posibles vulnerabilidades, también permite exponer la información de un grupo mayor de usuarios al explotar una vulnerabilidad en particular. Se debe tener en cuenta que la enumeración de usuarios no depende directamente de la automatización, la automatización aumenta el impacto en caso de lograr explotar una vulnerabilidad.

5.3.4.5. Lineamiento L4.5

Control definido: Establecer políticas para creación de usuarios que evite la fácil enumeración o adivinación de los mismos (Por ejemplo no utilizar números de cédula como usuario, nombre del usuario con números consecutivos del tipo pedro123, entre otros).
Pruebas de Validación: Para ataques de enumeración de usuarios se debe definir un formato que ayude al usuario a crear nicknames que no sean de fácil adivinación para un atacante. Las validaciones propuestas ayudan a identificar si la aplicación cuenta con una política para la creación de usuarios.

5.3.4.6. Lineamiento L4.6

Control definido: La contraseña y el usuario deben ser sometidos a funciones criptográficas que incluyan un valor salt, antes de ser enviados al servidor.
Pruebas de Validación: Una forma de evitar la exposición de la contraseña en la mensajería y los ataques de enumeración dirigidos a las contraseñas y los usuarios de la aplicación, consiste en aplicar funciones criptográficas robustas con valores aleatorios que ayuden a proteger los datos tanto en reposo como en tránsito.

5.3.4.7. Lineamiento L4.7

Control definido: La autenticación biométrica, si hay, no está atada a un evento del tipo "true" o "false" y almacena de forma segura las credenciales de acceso. keychain (iOS) o en el keystore (Android).
Pruebas de Validación: Algunas aplicaciones aprovechan las funcionalidades nativas de biometría (Touch Id - Face Id) utilizadas por los dispositivos móviles, para facilitar el proceso de autenticación de los usuarios. El lineamiento se enfoca principalmente en la integración de la aplicación móvil con los servicios de biometría soportados por el móvil de forma segura.

5.3.4.8. Lineamiento L4.8

Control definido: Existe un mecanismo de segundo factor de autenticación (2FA) en el servidor y es aplicado consistentemente en transacciones o acciones que manejan información sensible en la aplicación.
Pruebas de Validación: Las aplicaciones utilizan un segundo factor de autenticación al momento de realizar operaciones consideradas sensibles, como puede llegar a ser un cambio

de contraseña, realizar una transferencia financiera o al consultar información de salud como es el caso de resultados de laboratorio entre otros. Dentro del esquema de autenticación se define el segundo factor de autenticación como algo que tenemos, por ejemplo un token físico, un token tipo software, mensajes SMS o notificaciones push [76]. Algunas aplicaciones implementan un segundo factor de autenticación bajo una arquitectura que no garantiza su cumplimiento.

5.3.4.9. Lineamiento L4.9

Control definido: Controlar riesgos relacionados a la recuperación de credenciales de acceso, a través de funcionalidades de recordar credenciales.
Pruebas de Validación: Algunas aplicaciones cuentan con funcionalidades de autogestión que permiten recuperar el usuario o la contraseña en caso de olvido. En ocasiones a nivel de diseño, las implementaciones cuentan con debilidades de seguridad que pueden ser aprovechadas fácilmente por un atacante, con el fin de lograr suplantar un usuario.

5.3.5. Manejo de Sesiones y autorización

5.3.5.1. Lineamiento L5.1

Control definido: Implementar procedimientos para verificar que una persona o entidad que busca acceso a la información de salud electrónica protegida, se encuentra debidamente autorizado. (Referencia insegura directa a objetos).
Pruebas de Validación: Normalmente al momento del diseño de arquitectura y desarrollo de las aplicaciones aplicaciones móviles, se utilizan referencias a objetos que pueden ser un fichero, base de datos o directorios, para dar acceso a un recurso en específico [77]. Por ejemplo la variable “id” en una petición (Request) al backend, puede contener un valor (número de cédula) que hace referencia a un registro en un directorio de forma directa, permitiendo por ejemplo la descargar un archivo con resultados de exámenes de laboratorio. Al no implementar adecuadamente un control de autorización, un atacante con una sesión establecida correctamente, podrá descargar los exámenes de otros usuarios con solo modificar el valor de la variable “id”. Otro vector consiste en la posibilidad de consumir servicios de forma directa, sin contar con los debidos privilegios o autorización para hacerlo.

5.3.5.2. Lineamiento L5.2

Control definido: Informar al cliente, al inicio de cada sesión, la fecha y hora del último ingreso a este canal y la dirección IP .
Pruebas de Validación: Informar al usuario final la fecha y hora del último acceso a la aplicación y desde que dirección IP se originó, ayuda de forma preventiva a identificar posibles accesos no autorizados e informar oportunamente a la entidad del servicio y realizar cambio de contraseña para evitar o mitigar una posible exposición de información por suplantación

de identidad. La validación no solo consiste en verificar la existencia de la información al iniciar una sesión, también se debe validar su correcta implementación para evitar riesgos de seguridad.

5.3.5.3. Lineamiento L5.3

Control definido: Una sesión solo puede ser entregada al cliente por parte del servidor previa autenticación válida. Si la aplicación maneja sesiones antes de autenticación, debe existir un mecanismo que autentique una nueva sesión antes de consumir los servicios
Pruebas de Validación: Dependiendo de la arquitectura utilizada para las aplicaciones móviles, la sesión es entregada una vez supera correctamente la autenticación. En algunos casos el backend entrega una sesión al momento de abrir la aplicación o después de ingresar el usuario cuando la autenticación de usuario y contraseña se realiza en 2 pasos, es decir, en una primera pantalla valida el usuario y en una segunda pantalla valida la contraseña. Este tipo de diseño puede generar brechas de seguridad si no se cuenta con los debidos controles.

5.3.5.4. Lineamiento L5.4

Control definido: Si se utiliza la gestión de sesión por estado, el servidor remoto usa tokens de acceso aleatorios para autenticar los pedidos del cliente sin requerir el envío de las credenciales del usuario en cada uno.
Pruebas de Validación: Comúnmente las aplicaciones utilizan la autenticación con estado o conocida en el idioma ingles como "stateful authentication" [78], tiene como principio generar un valor de sesión aleatorio a un usuario que ha superado satisfactoriamente la autenticación. El id de sesión generando identificará al usuario en las siguientes peticiones que se realicen al backend y no contiene ningún tipo de información. Los datos de sesión del usuario (tipo de usuario, rol, acceso permitido entre otros) es almacenada en el backend.

5.3.5.5. Lineamiento L5.5

Control definido: Si se utiliza la autenticación basada en tokens sin estado, el servidor proporciona un token que se ha firmado utilizando un algoritmo seguro, no expone información sensible e implementa controles para evitar la suplantación.
Pruebas de Validación: La autenticación basada en tokens corresponde al envío de un token debidamente firmado o cifrado, que es validado de lado del backend, con el fin de definir que servicios puede consumir y con que tipo de privilegios cuenta el solicitante. El token es enviado en cada petición realizada al backend y no es almacenado en el servido [78]. El estandar RFC 7519 hace referencia a Json Web Token (JWT), que es unos de los formatos comúnmente utilizado para controlar la autorización de acceso a los servicios.

5.3.5.6. Lineamiento L5.6

Control definido: Cuando el usuario cierra la sesión en la aplicación, la sesión debe ser caducada en el servidor.
Pruebas de Validación: Al autenticarse correctamente un usuario en la aplicación, el backend entrega una sesión que permite identificarlo en las próximas peticiones que se realicen sin tener que enviar nuevamente las credenciales de acceso. Por esta razón, cuando el usuario decide cerrar la sesión desde la aplicación, el backend debe tener la capacidad de caducar la sesión con el fin de evitar que la sesión sea reutilizada por un atacante.

5.3.5.7. Lineamiento L5.7

Control definido: Las sesiones y los tokens de acceso expiran luego de un tiempo predefinido de inactividad.
Pruebas de Validación: Existen diversos motivos por el cual un usuario no realiza de forma voluntaria un cierre de sesión. Por ejemplo una de las causas corresponde al olvido que puede tener el usuario de cerrar la sesión, otro motivo puede ser un cierre no esperado por parte de la aplicación debido a un defecto propio de la aplicación o por un cierre forzado por parte del mismo sistema operativo. En todos estos casos la sesión quedará activa de lado del backend, abriendo la posibilidad de que un ciberdelincuente realice una suplantación de identidad, reutilizando una sesión válida de la víctima.

5.3.5.8. Lineamiento L5.8

Control definido: Controlar las sesiones simultáneas sin importar los diferentes canales que provean el servicio.
Pruebas de Validación: Normalmente los servicios que expone una organización cuenta con un canal móvil y un canal web para su acceso. Dependiendo de la arquitectura utilizada, los servicios pueden ser los mismos o independientes. A nivel funcional un usuario no entra de forma simultanea al mismo canal o a los dos canales, lo cual si representa un riesgo de seguridad, debido a que un atacante que ha logrado capturar la sesión de la víctima o las credenciales de autenticación por medio de diferentes vectores de ataque, podrá acceder de forma simultánea a la aplicación y consumir los servicios sin que la víctima se de por enterado.

5.3.5.9. Lineamiento L5.9

Control definido: La aplicación informa al usuario acerca de los accesos a su cuenta desde dispositivos no registrados. El usuario es capaz de ver una lista de los dispositivos conectados y bloquear el acceso desde ciertos dispositivos.
Pruebas de Validación: El fin del lineamiento es brindar al usuario la posibilidad de reaccionar ante una posible suplantación de identidad. Para lograr el objetivo la aplicación debe

estar en la capacidad de realizar un proceso de registro e identificación único del dispositivo y contar con una alertar que le informe al usuario por push o vía email sobre accesos realizados a la aplicación en tiempo real. Con este proceso el usuario podrá tener un mayor control sobre cuales dispositivos están permitidos autenticar e identificar si alguien no autorizado ingresó a la aplicación.

5.3.6. Comunicación a través de la red

5.3.6.1. Lineamiento L6.1

Control definido: Implementar medidas de seguridad técnicas y canales seguros para protegerse contra el acceso no autorizado a información médica protegida electrónica que se transmite a través de una red de comunicaciones electrónicas.
Pruebas de Validación: La información además de ser protegida a nivel del dato (L3-Criptografía) debe ser enviada por medio de un canal seguro, utilizando protocolos y cifrado definidos en el Lineamiento L6.2 de forma consistente en la aplicación. El fin del lineamiento es asegurar que la aplicación en todo momento está utilizando HTTPS en las comunicaciones. El no cumplir con este lineamiento, el backend permitirá la degradación del protocolo de comunicación segura.

5.3.6.2. Lineamiento L6.2

Control definido: Las configuraciones del protocolo TLS siguen las mejores prácticas o tan cerca posible mientras que el sistema operativo del dispositivo lo permite y configura Ciphersuite que no estén considerados como débiles o vulnerables.
Pruebas de Validación: TLS y SSL son protocolos diseñados para brindar seguridad en la transferencia de información entre el cliente y el servidor. Los cipher suites son utilizados por los protocolos TLS y SSL para asegurar la transferencia de la información. Están conformado por una combinación de algoritmos de autenticación, cifrado y el código de autenticación de mensajes (MAC). Un ejemplo de un cipher suite es el siguiente: DHE-RSA-AES256-GCM-SHA384. Si alguno de estos algoritmos cuenta con debilidades de seguridad conocidas, un atacante podrá intentar degradar o romper el cifrado de la comunicación con el fin de acceder a la información [79].

5.3.6.3. Lineamiento L6.3

Control definido: 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.
Pruebas de Validación: El ataque conocido como Hombre en el Medio (MiTM) no solo tiene como fin intentar interceptar el tráfico de la víctima para alterar la información en

tránsito u obtener datos confidenciales, también es utilizado para analizar el comportamiento de la aplicación y encontrar nuevos vectores de ataque que permitan comprometer la aplicación o el backend. Por lo anterior el lineamiento tiene como objetivo evitar que un atacante intercepte las comunicaciones.

5.3.7. Interacción con la Plataforma

5.3.7.1. Lineamiento L7.1

Control definido: La aplicación requiere la mínima cantidad de permisos.
Pruebas de Validación: Cuando la aplicación expone componentes IPC, se debe realizar las respectivas configuraciones con los permisos necesarios para evitar que aplicaciones maliciosas puedan acceder a dichos componentes. La forma de gestionar los permisos en Android es diferente al de iOS a nivel conceptual pero bajo un mismo objetivo de proteger la data y los accesos a los recursos.

5.3.7.2. Lineamiento L7.2

Control definido: La aplicación no exporta funcionalidades sensibles vía esquemas de URL, salvo que dichos mecanismos estén debidamente protegidos.
Pruebas de Validación: Tanto en plataformas iOS como Android permiten la comunicación entre aplicaciones móviles por medio de esquemas URL personalizados. Los Identificadores de Recursos Uniforme (URI) definen que tipo de acción se va a realizar en la aplicación y que parámetros debe utilizar. Si los esquemas URL no son protegidos de forma adecuada, el usuario de la aplicación será susceptible a fuga de información o posibles afectaciones económicas.

5.3.7.3. Lineamiento L7.3

Control definido: La aplicación no exporta funcionalidades sensibles a través de mecanismos IPC salvo que los mecanismos estén debidamente protegidos.
Pruebas de Validación: Dependiendo del alcance de la aplicación, los desarrollares utilizan los mecanismo IPC con el fin de permitir que otras aplicaciones consuma ciertos recursos expuesto por la aplicación. Para permitir esta comunicación, los recursos como por ejemplo los activities, broadcast receivers, content providers y services deben ser exportados. De no aplicar una correcta configuración, los mecanismos pueden llegar a exponer información personal o sensible del usuario.

5.3.7.4. Lineamiento L7.4

Control definido: JavaScript se encuentra deshabilitado en los WebViews salvo que sea necesario.

Pruebas de Validación: Webview es un navegador web inmerso en una aplicación móvil pero con limitantes funcionales. Su entorno de ejecución es independiente al navegador nativo del dispositivo móvil y utiliza la sandbox propia de la aplicación para el almacenamiento en caché. Bajo esta definición, los webview son susceptible a ataques de inyección de código como cualquier navegador web.

5.3.7.5. Lineamiento L7.5

Control definido: Los WebViews se encuentran configurados para permitir el mínimo de los manejadores (idealmente, solo https). Manejadores peligrosos como file, tel y app-id deben estar deshabilitados.
Pruebas de Validación: Los webview tienen la propiedad de cargar contenido de forma remota o cargar contenido local ubicado en la sandbox de la aplicación, aprovechando manejadores (Handlers) considerados peligrosos, como es el caso de file:// o realizar llamadas telefónicas sin autorización como es el caso de tel://.

5.3.8. Calidad de Código y Configuración del Compilador

5.3.8.1. Lineamiento L8.1

Control definido: La aplicación es firmada bajo las versiones 1 y 2 y provista con un certificado válido.
Pruebas de Validación: El instalador de las aplicaciones móviles se firman antes de ser cargadas en las respectivas tiendas. Si la firma no cuenta con un algoritmo seguro (SHA1, MD5, etc), un atacante podrá inyectar código malicioso en la aplicación original y firmarla nuevamente obteniendo el mismo valor hash de los archivos originales logrando engañar a los controles antimalware de los dispositivos móviles, para evitar cualquier alertamiento. Para este lineamiento solo aplica validaciones de tipo estático.

5.3.8.2. Lineamiento L8.2

Control definido: La aplicación fue liberada en modo release y con las configuraciones apropiadas para el mismo. No debuggable y sin Backup activo.
Pruebas de Validación: En la etapa de desarrollo, los instaladores generados normalmente tienen la opción de debug activo para realizar una trazabilidad a los diferentes errores que se pueden presentar durante las pruebas de desarrollo y funcionales. Para un atacante o un malware, el debug le permite interactuar con la aplicación por medio de herramientas de desarrollo como es el caso de ADB (Android Debug Bridg), que permite acceder a la sandbox de la aplicación y extraer información. También se puede utilizar herramientas como andbug para realizar un seguimiento paso a paso de una funcionalidad con el fin de obtener información privada y sensible de la aplicación como es el caso de llaves de cifrado o data privada del usuario. Normalmente los instaladores tienen habilitada la opción de

backup, que genera un riesgo de exposición de información debido a que un atacante o un malware podrá extraer información privada de la sandbox de la aplicación, sin contar con permisos de root.

5.3.8.3. Lineamiento L8.3

Control definido: La aplicación captura y maneja debidamente las posibles excepciones.
Pruebas de Validación: En ocasiones las aplicaciones pueden entrar en un estado no esperado por diferentes causas, generando una excepción. La excepción puede ser originada por defectos o bugs propios en la aplicación o de forma intencional, al obligar a la aplicación a recibir datos no esperados o alterando su flujo funcional. En consecuencia, la aplicación responde con errores que pueden revelar información útil para un atacante, relacionada al componente afectado (Sistema operativo, plataforma web, base de datos entre otros) [80].

5.3.8.4. Lineamiento L8.4

Control definido: La lógica de manejo de errores en los controles de seguridad deniega el acceso por defecto.
Pruebas de Validación: A nivel de arquitectura de las aplicaciones, diferentes controles de seguridad son incorporados con el fin de proteger la aplicación de diferentes tipos de ataques que puedan comprometer la disponibilidad, integridad y confidencialidad de la información. En caso de generarse un error, por ejemplo a nivel de comunicaciones ya sea intencional o no, los controles de seguridad deben por defecto denegar el acceso al recurso protegido. Algunos ejemplos son, funciones captcha, análisis de riesgo, módulos de autenticación entre otros.

5.3.9. Impedir el Análisis Dinámico y la Manipulación

5.3.9.1. Lineamiento L9.1

Control definido: La aplicación detecta y responde a la presencia de un dispositivo rooted o con jailbreak, ya sea alertando al usuario o finalizando la ejecución de la aplicación.
Pruebas de Validación: Normalmente para el desarrollo de las pruebas de seguridad en aplicaciones móviles, los hackers éticos y ciberdelincuentes utilizan dispositivos móviles rooted para android y con jailbreak para iOS, con el fin de instalar una serie de herramientas para el análisis de la aplicación y lograr accesos al sistema operativo que normalmente no se puede obtener desde un dispositivo móvil sin root. Este control ayuda a evitar el análisis dinámico de la aplicación. Otro grupo de usuarios disponen de dispositivos móviles con root con el fin de poder instalar aplicaciones móviles de pago de forma gratuita y realizar personalizaciones en la configuración que son bloqueados de fábrica. Utilizar aplicaciones móviles en dispositivos con acceso a root, permiten que un malware logre una mayor afectación de la privacidad de la información gestionada por la aplicación móvil.

5.3.9.2. Lineamiento L9.2

Control definido: La aplicación previene el debugging o detecta y responde al debugging de la aplicación. Se deben cubrir todos los protocolos.
Pruebas de Validación: Un atacante utiliza la opción de debug en la aplicación con el fin de evaluar en tiempo de ejecución las diferentes funcionalidades y crear puntos de interrupción para capturar los valores de las diferentes variables y acceder de esta forma a información privada de la aplicación que permita saltar ciertos controles de seguridad, como por ejemplo las validaciones de root o el control de certificados SSL Pinning, entre otros.

5.3.9.3. Lineamiento L9.3

Control definido: La aplicación detecta y responde a modificaciones de ejecutables y datos críticos de la propia aplicación.
Pruebas de Validación: Una de las formas de evadir ciertos tipos de controles de seguridad implementados en la aplicación, consiste en alterar el instalador a nivel de configuración y modificación del código fuente. La modificación del binario permite en algunos casos evadir la validación de certificados, detección de equipos con root, activar el debug para realizar análisis en tiempo de ejecución, modificar los flujos funcionales de la aplicación cambiando la lógica desde el código fuente entre otros. Por lo anterior, la aplicación debe contar con controles eficaces que ayuden a evitar alteraciones en el binario generado.

5.3.9.4. Lineamiento L9.4

Control definido: La aplicación detecta la presencia de las herramientas de ingeniería reversa o frameworks mas utilizados.
Pruebas de Validación: Al momento de realizar pruebas de ingeniería inversa en las aplicaciones móviles, los atacantes suelen utilizar una serie de herramientas que facilitan dicha tarea y son instaladas en los dispositivos móviles. El fin del lineamiento es evitar que la aplicación se ejecute en caso de detectar este tipo de herramientas, dificultando el actuar del atacante.

5.3.9.5. Lineamiento L9.5

Control definido: La aplicación detecta y responde al ser ejecutada en un emulador.
Pruebas de Validación: Una de las formas mas sencillas para validar una aplicación móvil sin tener a disposición un dispositivo móvil físico y con acceso a root, es por medio de emuladores. Los emuladores permiten además del acceso como root al dispositivo, permite instalar programas que ayudan a realizar las pruebas de ingeniería inversa sobre la aplicación. Por lo anterior, el fin del lineamiento es dificultar el análisis en tiempo de ejecución de la aplicación al atacante, obligándolo a buscar técnicas que le permita saltar el control o de buscar un móvil y realizar el proceso de rooted.

5.3.9.6. Lineamiento L9.6

Control definido: La aplicación implementa múltiples mecanismos de detección (Resiliencia) para los puntos del 9.1 al 9.5. Es de aclarar que a mayor cantidad y diversidad de mecanismos usados, mayor es la resistencia.
Pruebas de Validación: El fin del lineamiento es validar que los controles aplicados en los puntos del 9.1 al 9.5 se implementaron de forma efectiva. Un atacante intentará buscar la forma de evadir los controles utilizando diferentes técnicas. Cada uno de estos lineamientos se soportan entre sí para brindar una mayor efectividad en el control definido.

5.3.9.7. Lineamiento L9.7

Control definido: La ofuscación es aplicada a las defensas del programa y código funcional, lo que a su vez impide la des-ofuscación mediante el análisis dinámico.
Pruebas de Validación: El proceso de ofuscación corresponde a la transformación del código y la data, con el fin de dificultar su compresión y evitar que un atacante entienda las funciones relacionada a la lógica de negocio y procesos críticos [81]. El entendimiento del código en funciones críticas de la aplicación, le permite a un atacante alterar el comportamiento de la aplicación, mediante la modificación del código java en el binario o en tiempo de ejecución. Aunque el lineamiento corresponde más a pruebas estáticas, es relevante a aclarar que la ofuscación del código es un método que dificulta o limita el uso de otros vectores de ataque del tipo dinámico.

5.4. Informe

El informe de seguridad es un documento que refleja el estado real de seguridad de la aplicación móvil previamente analizada. Debe contar con la información necesaria para la toma de decisiones que permitan establecer correcciones o la implementación de nuevos controles de seguridad, bajo una priorización definida por el negocio [82]. Para la presente tesis se hace uso del informe técnico, describiendo de una forma general una propuesta del alcance y posible contenido del informe. Los puntos a tener en cuenta son:

- Encabezados: En el encabezado puede contener el nombre de la aplicación analizada, el nombre del cliente, un código interno, la fecha y versión del documento.
- Tabla de contenido: Debe contar con una tabla de contenido para cada uno de los capítulos contenido en el informe.
- Introducción: En este punto se describe de forma general que aplicación se validó y el enfoque de las pruebas realizadas.
- Metodología: Se debe dar a conocer la metodología utilizada para ponderar y definir el nivel de riesgo de los hallazgos de seguridad identificados en las pruebas de seguridad.

- Alcance del proyecto: En este punto se describe el alcance de las pruebas definidas, por ejemplo se define si es solo en Android o iOS o en ambas, si las pruebas son del tipo Black Box o el cliente aportará usuarios de prueba, también se define si es sobre un entorno de pruebas o directamente en producción, el tiempo de duración de las pruebas entre otras.

- Resumen: El objetivo del resumen es describir el total de vulnerabilidades detectadas y una breve descripción bajo la clasificación del nivel de riesgo definido y dar a conocer el nivel de riesgo general de la aplicación.

- Resumen de vulnerabilidades: Se construye con una tabla que contiene el Id de la vulnerabilidad, un título, el nivel de riesgo, La clasificación según Top 10 OWASP Mobile y en que plataforma aplica (iOS/Android).

- Detalle de las vulnerabilidades: En el detalle se debe describir el Id de la vulnerabilidad, Titulo, Riesgo, Score del riesgo, Plataforma, Descripción detallada de la vulnerabilidad, Evidencias con imágenes de alta resolución y al menos 2 recomendaciones de solución.

- Conclusiones: En las conclusiones se describe el por que? el evaluador, define el nivel de riesgo general de la aplicación móvil analizada, adicionalmente se incluye la relevancia e impacto de las vulnerabilidades encontradas a nivel de confidencialidad, disponibilidad e integridad de la información.

- Toda figura, gráfica y tablas expuestas en el informe deben estar enumeradas y con una breve descripción.

6. Evaluación de la implementación de la guía en un caso de prueba

Con el fin de evaluar la guía metodológica propuesta en la tesis, se decide seleccionar una aplicación móvil relacionada al sector salud en Colombia, con las capacidades funcionales de gestionar datos de salud y personales de los pacientes. Para la prueba de concepto se procedió a seleccionar una aplicación móvil desde la tienda del Play Store de Google, la cual cumple con los requisitos necesarios para el desarrollo de las pruebas. La aplicación seleccionada tiene aproximadamente 4 años en la tienda Play Store de Google con más de 100.000 descargas.
Con la aplicación identificada, se procedió a realizar las pruebas de seguridad bajo las consideraciones dadas en la guía metodológica propuesta en el presente documento de tesis. Finalizadas las pruebas se procedió a reportar los hallazgos encontrados a la empresa dueña de la aplicación, quienes tendrán la responsabilidad de gestionar los hallazgos reportados con el fin de evitar incidentes de seguridad que ponga en riesgo la confidencialidad de los datos personales y de salud de los usuarios de la aplicación y posibles multas millonarias al no gestionar de forma segura los datos de los usuarios.

6.1. Análisis y pruebas realizadas

Como parte de la guía metodológica propuesta, se procede a realizar la adecuación del ambiente para iniciar las pruebas de seguridad con un dispositivo móvil físico con sistema operativo Android. Se realiza un reconocimiento de las diferentes funcionalidades de la aplicación con el fin de construir el modelado de amenazas y proyectar las respectivas pruebas de seguridad para validar el cumplimiento a los lineamientos de seguridad sugeridos.
En las tablas **6-1**, **6-2**, **6-3**, **6-4**, **6-5**, **6-6**, **6-7**, **6-8** y **6-9** se describe el resultado del modelado de amenazas establecido para la aplicación móvil seleccionada en la prueba de concepto y en este sentido en el capítulo 6.1.1, se describe el resultado de las validaciones de seguridad realizadas a la aplicación móvil, según cada uno de los lineamientos que debería cumplir, en las tablas **6-10**, **6-11**, **6-12**, **6-13**, **6-14**, **6-15**, **6-16**, **6-17** y **6-18**.

6.1.1. Planteamiento del Modelado de amenazas

Item	**Escenarios de amenaza - Backend**	**Validar**
1	Un atacante logra acceder a información sensible o privada del paciente al lograr obtener las cookies de sesión de la aplicación basado en un ataque de inyección de javascript XSS.	X
2	Un atacante logra acceder a información sensible o privada del paciente al capturar tráfico basado en interceptación de las comunicaciones (Man in The Middle).	X
3	Un atacante logra acceder a información sensible o privada del paciente, por medio de ataques del tipo clickjacking.	X
4	Un atacante logra acceder a información sensible o privada del paciente, al realizar un MIME-sniffing para enviar un ataque del tipo XSS.	X
5	Un atacante logra acceder a información sensible o privada del paciente, al inyectar código javascript en el contenido de la aplicación, para el cargue de páginas o contenido desde dominios de terceros con el fin de aprovechar por ejemplo ataques del tipo XSS o inyectar payloads que comprometan el servidor.	X
6	Un atacante logra determinar el tipo y versiones de componentes o web-servers que forman parte de la arquitectura de la aplicación, con el fin de enfocar un ataque mas efectivo y acceder a información sensible o privada del paciente.	X
7	Un atacante logra acceder a información sensible o privada del paciente, mediante la exploración de directorios y consumo directo de los servicios en el backend, como usuario anónimo o autenticado.	X
8	Un atacante logra acceder a información sensible o privada del paciente, al lograr inyectar en el Backend código javascript, sentencias sql, comandos de sistema operativo, LDAP o caracteres que no son permitidos según su lógica de negocio, definidas normalmente en el frontend.	X
9	Un atacante interno o externo logra obtener información sensible o privada del paciente, al acceder a los web logs registrados en el backend, el cual fue comprometido previamente.	-

Tabla 6-1.: Validar: Escenarios de amenaza - Backend.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Almacenamiento de datos y la Privacidad**	**Validar**
1	Un atacante logra acceder a información sensible o privada del paciente, al lograr capturar en código, o en archivos de configuración de la aplicación, claves de autenticación o llaves de cifrado y descifrado de información sensible.	X
2	Un atacante logra acceder a información sensible o privada del paciente o de la aplicación, al monitorear los logs registrados en el dispositivo móvil.	X
3	Un atacante logra acceder a información sensible o privada del paciente, al estar expuesta por métodos HTTP inseguros al momento de interceptar las comunicaciones.	X
4	Un atacante logra acceder a información sensible o privada del paciente, al ser compartida y transmitida por medio de librerías de terceros como parte funcional de la aplicación.	X
5	Un atacante logra acceder a información sensible o privada del paciente, mediante el análisis de la caché generada por la aplicación en la sandbox.	X
6	Una aplicación mal intencionada logra capturar información sensible o privada del paciente gestionada por la aplicación, mediante la captura de datos guardada en el portapapeles.	X
7	Un atacante o malware logra acceder a información sensible o privada del paciente, que es almacenada en base de datos ubicada en la sandbox de la aplicación.	X
8	Un atacante logra obtener información sensible o privada del paciente por medio de ataques del tipo shoulder surfing o por un malware que accede a capturas de pantallas de la aplicación.	X
9	Un atacante logra acceder a información sensible o privada del paciente, al extraer del dispositivo un backup de la aplicación o acceder a backups existentes en otras unidades de almacenamiento.	X
10	Un atacante o malware logra acceder a información sensible o privada del paciente, que es almacenada en memoria del dispositivo.	X
11	Un atacante logra acceder fácilmente a las aplicaciones instaladas en un dispositivo robado o extraviado, debido a la no configuración de un código de acceso en el móvil.	X
12	Un atacante o un malware logra acceder a información sensible o privada del paciente, al lograr extraer archivos o base datos de la sandbox de la aplicación, en móviles que no se encuentran con jailbreak o rooted, debido a una asignación errónea de permisos a los archivos.	X
13	Un atacante logra obtener información financiera del paciente, en flujos de la aplicación que permiten realizar pagos, en canales o servicios que proveen terceros.	-

Tabla 6-2.: Validar: Escenarios de amenaza - Almacenamiento de datos y la privacidad. Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Criptografía**	**Validar**
1	Un atacante logra acceder a información sensible o privada del paciente, en transito entre el móvil y el backend o almacenada en el dispositivo sin cifrar.	X
2	Un atacante logra acceder a información sensible o privada del paciente que se encuentra cifrada, utilizando las llaves de descifrado quemadas en el código, expuestas en la mensajería , o almacenadas de forma no segura.	X
3	Un atacante logra acceder a información sensible o privada del paciente que se encuentra cifrada, explotando vulnerabilidades en las funciones criptográficas.	X
4	Un atacante logra obtener información sensible o privada del paciente, debido a que la llave criptográfica comprometida es utilizada para varios propósitos o en común para todos los usuarios.	X

Tabla 6-3.: Validar: Escenarios de amenaza - Criptografía.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Autenticación**	**Validar**
1	Un atacante logra obtener información sensible o privada del paciente, al lograr consumir servicios de forma anónima.	X
2	Un atacante logra acceder a la aplicación y consumir los servicios del paciente, suplantando los identificadores del dispositivo (deviceID), el cual es utilizado como mecanismo de autenticación por parte del backend. El acceso también se puede dar por robo o pérdida del móvil.	X
3	Un atacante logra acceder a la aplicación y consumir los servicios del paciente, manipulando la respuesta del servidor en el flujo de autenticación, mediante la inyección de una respuesta válida de login, con el fin de engañar a la aplicación.	X
4	Un atacante logra acceder a la aplicación y consumir los servicios del paciente, al lograr adivinar fácilmente la contraseña de la víctima, utilizando un listado de contraseñas mas comunes. O generar una denegación de servicios masivo en el intento.	X
5	Un atacante por medio de diferentes técnicas, logra obtener un listado de usuarios existentes para el servicio, con el fin de potencializar otros ataques y materializar una suplantación de identidad.	X
6	Un atacante logra capturar las credenciales de acceso de la víctima, al realizar una interceptación de las comunicaciones o al encontrarse almacenadas de forma insegura en el dispositivo.	X
7	Un atacante con las credenciales de acceso de la víctima ya capturadas, logra realizar transacciones de riesgo desde la aplicación, obteniendo información sensible o privada del paciente.	X
8	Un atacante logra acceder a información sensible o privada del paciente, al acceder a servicios protegidos con un segundo factor de autenticación, reutilizando un OTP que tiene un tiempo de vida muy alto.	-
9	Un atacante logra acceder a información sensible o privada del paciente, al acceder a servicios protegidos con un segundo factor de autenticación, adivinando el OTP por medio de un ataque de fuerza bruta.	-
10	Un atacante logra obtener o cambiar las credenciales de acceso a la aplicación, aprovechando debilidades en el mecanismo de recordar usuario o contraseña.	-

Tabla 6-4.: Validar: Escenarios de amenaza - Autenticación.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Manejo de sesiones y autorización**	**Validar**
1	Un atacante logra acceder a información sensible o privada de otros pacientes, al lograr manipular las referencias a un objeto en la aplicación, para acceder a otros objetos sin autorización. Por ejemplo al realizar consultas, descargas de archivos o acceso a recursos privilegiados.	X
2	Un atacante logra acceder a información sensible o privada del paciente, al consumir servicios con sesiones entregadas por la aplicación sin autenticar.	X
3	Un atacante logra acceder a información sensible o privada del paciente, al lograr predecir el token de sesión y por ende acceder a los servicios de la aplicación.	X
4	Un atacante logra acceder a información sensible o privada del paciente, debido a que la aplicación no utiliza una sesión y envía las credenciales de acceso en cada petición.	X
5	Un atacante logra obtener información confidencial del paciente, almacenada en el Json Web Token.	-
6	Un atacante logra acceder a información sensible o privada del paciente, mediante la alteración del los valores contenidos en el Json Web Token, al utilizar algoritmos inseguros en la firma.	-
7	Un atacante logra acceder a información sensible o privada del paciente, mediante la alteración del los valores contenidos en el Json Web Token, al lograr obtener el "secret" de la firrma por medio de un ataque de fuerza bruta o diccionario.	-
8	Un atacante logra acceder al sistema y obtener información sensible o privada del paciente, al robar un token mediante la interceptación de las comunicaciones y lo utiliza para suplantar al usuario legítimo.	-
9	Un atacante logra acceder a información sensible o privada del paciente, al consumir servicios con una sesión previamente cerrada por la víctima desde la aplicación.	X
10	Un atacante logra acceder a información sensible o privada del paciente, al lograr acceder a la aplicación, mediante la extracción del token almacenado en base de datos sqlite.	X
11	Un atacante logra acceder a información confidencial o privada del paciente, al secuestrar la sesión o ingresar de forma simultanea a la aplicación.	X

Tabla 6-5.: Validar: Escenarios de amenaza - Manejo de sesiones y autorización.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Comunicación a través de la red**	**Validar**
1	Un atacante logra acceder a información sensible o privada del paciente, al capturar la información transmitida entre la aplicación y el servidor sin cifrar.	X
2	Un atacante logra acceder a información confidencial y privada capturando información transmitida entre la aplicación y el servidor por medio de la degradación de los ciphersuite o de los protocolos de comunicación seguros.	X
3	Un atacante logra acceder a información confidencial o privada del paciente, capturando información transmitida entre la aplicación y el servidor, utilizando certificados que no son confiables (autofirmados).	X
4	Un atacante logra acceder a información confidencial o privada del paciente, capturando información transmitida entre la aplicación y el servidor, utilizando certificados que son de confianza pero no corresponden al suministrado por el backend.	X
5	Un atacante logra acceder a la aplicación aprovechando funciones para recuperar el usuarios/contraseña o realizar operaciones críticas protegidas por un segundo factor de autenticación, al suplantar la SIMCARD de la víctima, para obtener el OTP y continuar con el flujo. (SIM SWAP Attack).	-
6	Un atacante logra acceder a información sensible o privada del paciente, accediendo a servicios protegidos con un segundo factor de autenticación, mediante la extracción del mensaje de voz que deja el mecanismo de llamada telefónica para la entrega del OTP, al momento de no responder la llamada.	-
7	Un atacante logra acceder a información sensible o privada del paciente, al explotar vulnerabilidades encontradas en librerías de terceros (Heartbleed, ShellShock).	X

Tabla 6-6.: Validar: Escenarios de amenaza - Comunicación a través de la red.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Interacción con la Plataforma**	**Validar**
1	Un atacante logra acceder a información sensible o privada del paciente, basado en una inadecuada o innecesaria asignación de permisos en la aplicación, hacia los recursos del sistema operativo.	X
2	Un atacante logra acceder a información sensible o privada del paciente, al momento de realizar una interceptación de las comunicaciones internas IPC de la app, para capturar información o inyectar valores para engañar a la víctima.	X
3	Un atacante logra acceder a información sensible o privada del paciente, al lograr acceder a información sensible almacenada en la sandbox de la aplicación, mediante la inyección de java script en los webview.	X

Tabla 6-7.: Validar: Escenarios de amenaza - Interacción con la plataforma.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Calidad de código y configuración del compilador**	**Validar**
1	Un atacante logra acceder a información sensible o privada del paciente, mediante la inyección de un payload malicioso en la aplicación, teniendo en cuenta que solo fue firmada con la versión 1 en Android. (janus).	X
2	Un atacante logra acceder a información sensible o privada del paciente o de la lógica del negocio, al firmar la aplicación con algoritmos considerados débiles o vulnerables.	X
3	Un atacante logra obtener información relacionada a la lógica funcional de la aplicación, al encontrar en el código fuente, funciones y métodos con nombres descriptivos, útiles para planear el bypass de un control de seguridad.	X
4	Un atacante logra acceder a información sensible o privada del paciente, mediante la modificación de valores en las variables utilizadas en ciertas funciones de la aplicación, en modo ejecución y sin modificar el binario.	X
5	Un atacante logra acceder a información sensible o privada del paciente, al tener acceso a los mensajes de debug expuestos por la aplicación, que no se desactivaron en el modo release.	X
6	Un atacante logra acceder a información sensible o privada del paciente o de la plataforma, al visualizar los mensajes de las diferentes excepciones no capturadas y tratadas por la aplicación.	X
7	Un atacante logra acceder a información sensible o privada del paciente, al lograr el acceso a funciones sin la debida autorización, al generarse un error en los controles de seguridad.	X
8	Un atacante logra acceder a información sensible o privada del paciente, al lograr inyecciones del tipo sentencias sql, Esquemas URL personalizados, códigos QR o llamadas IPC.	X

Tabla 6-8.: Validar: Escenarios de amenaza - Calidad de código y configuración del compilador.
Fuente: Elaboración propia.

Item	**Escenarios de amenaza - Impedir el análisis dinámico y la manipulación**	**Validar**
1	Un atacante logra acceder a información sensible o privada del paciente, evadiendo los controles de seguridad de acceso y autorización, mediante la instalación de herramientas de análisis dinámico de la aplicación y el acceso a recursos exclusivos para usuarios root.	X
2	Un atacante logra acceder a información sensible o privada del paciente, activando intencionalmente el modo debug de la aplicación, alterando su código fuente.	X
3	Un atacante logra acceder a información sensible o privada del paciente, evadiendo los controles de seguridad de acceso y autorización, mediante la manipulación del código fuente del binario.	X
4	Un atacante logra acceder a información sensible o privada del paciente, evadiendo los controles de seguridad basado en el análisis y manipulación de la lógica funcional, expuesta en el código fuente de la aplicación.	X

Tabla 6-9.: Validar: Escenarios de amenaza - Impedir el análisis dinámico y la manipulación.
Fuente: Elaboración propia.

6.1.2. Resultado validación lineamientos de seguridad.

Item	**Backend (Servicios)**	**Cumple**
1	Configurar de forma correcta las cabeceras en el servidor web.	No
2	Configurar controles que permitan asegurar que los usuarios solo pueden acceder a las URLs para las cuales poseen autorización.	No
3	Las validaciones realizadas en los diferentes puntos de entrada de la aplicación móvil, deben ser igualmente validados de lado del backend, para evitar inyecciones XSS, SQLi, XML o LDAP.	Si
4	Los logs registrados en el servidor no deben contener información considerada sensible o confidencial.	NA
5	Implementar controles que eviten el listado de directorios y archivos en los servidores con los que interactúa la aplicación (Hardening).	No

Tabla 6-10.: Resultado: Backend (Servicios).
Fuente: Elaboración propia.

Item	**Almacenamiento de datos y la privacidad**	Cumple
1	Las funcionalidades de almacenamiento de credenciales del sistema son utilizadas para almacenar la información sensible, como credenciales del usuario, Información de identificación personal PII y claves criptográficas.	No
2	No se escribe información sensible en los registros de la aplicación o en lugares de almacenamiento propios o externos del dispositivo móvil.	No
3	Para aplicaciones Híbridas o Web APP no se debe exponer información sensible por método GET, se debe enviar solo por Método POST.	No
4	No se comparte información sensible con servicios externos salvo que sea una necesidad de la arquitectura.	Si
5	Se debe desactivar el caché del teclado en los campos de texto donde se maneja información sensible.	Si
6	Se debe desactivar el portapapeles en los campos de texto donde se maneja información sensible.	No
7	No exponer información sensible mediante mecanismos entre procesos IPC.	Si
8	No exponer información sensible como contraseñas, datos personales o números de tarjetas de crédito a través de la interfaz o capturas de pantalla.	Si
9	No incluir información sensible en los respaldos generados por el sistema operativo, como es el caso de resultados de laboratorio almacenados en la sandbox de la aplicación en formato pdf.	Si
10	La aplicación remueve la información sensible de la vista cuando la aplicación pasa a un segundo plano.	No
11	La aplicación no conserva la información sensible en memoria más de lo necesario y la memoria es limpiada luego de su uso.	Si
12	La aplicación obliga a que exista una política mínima de seguridad en el dispositivo, como por ejemplo configurar un código de acceso.	No
13	Asignar permisos apropiados a los archivos y Base de datos gestionados por la aplicación.	Si
14	No utilizar credenciales o llaves de cifrado/descifrado quemadas en el código (Hard-coded).	Si
15	Cumplimiento de PCI en caso de gestionar números de tarjeta de crédito o débito en algún flujo de la aplicación.	NA

Tabla 6-11.: Resultado: Almacenamiento de datos y la privacidad.
Fuente: Elaboración propia.

Item	**Criptografía**	Cumple
1	Implementar un mecanismo para proteger la información sensible personal y la relacionada a la salud, con la utilización de mecanismos de cifrado y/o de seudonimización.	No
2	La aplicación no depende únicamente de criptografía simétrica, con claves quemadas en el código. Se recomienda el cifrado asimétrico.	No
3	La aplicación no utiliza protocolos o algoritmos criptográficos que son considerados deprecados para aspectos de seguridad.	No
4	La aplicación no reutiliza una misma clave criptográfica para varios propósitos.	NA

Tabla 6-12.: Resultado: Criptografía.
Fuente: Elaboración propia.

Item	**Autenticación**	Cumple
1	Si la aplicación provee acceso a un servicio remoto, un mecanismo aceptable de autenticación como usuario y contraseña es realizado en el servidor remoto.	Si
2	Existe una política de contraseñas y es aplicada en el servidor.	NA
3	El servidor implementa controles, cuando credenciales de autenticación son ingresadas una cantidad excesiva de veces. Automatización.	No
4	El servidor implementa controles, cuando se intenta realizar enumeración de usuarios en el login, recuperar contraseña, recuperar usuario o cualquier otro endpoint donde requiera el uso del usuario, número de cédula o identificador de fácil adivinación.	No
5	Establecer políticas para creación de usuarios que evite la fácil enumeración o adivinación de los mismos (Por ejemplo no utilizar números de cédula como usuario, nombre del usuario con números consecutivos pedro123 etc.).	No
6	La contraseña y el usuario deben ser sometidos a funciones criptográficas que incluyan un valor salt, antes de ser enviados al servidor.	No
7	La autenticación biométrica, si hay, no está atada a un evento del tipo “true” o “false” y almacena de forma segura las credenciales de acceso. keychain (iOS) o un keystore (Android).	NA
8	Existe un mecanismo de segundo factor de autenticación (2FA) en el servidor y es aplicado consistentemente.	No
9	Controlar riesgos relacionados a la recuperación de credenciales de acceso, a través de funcionalidades de recordar credenciales.	NA

Tabla 6-13.: Resultado: Autenticación.
Fuente: Elaboración propia.

Item	**Manejo de sesiones y autorización**	Cumple
1	Implementar procedimientos para verificar que una persona o entidad que busca acceso a la información de salud electrónica protegida, se encuentra debidamente autorizado. (Referencia insegura directa a objetos).	No
2	Informar al cliente, al inicio de cada sesión, la fecha y hora del último ingreso a este canal y la dirección IP. 2.3.4.9.5.	No
3	Una sesión solo puede ser entregada al cliente por parte del servidor previa autenticación válida. Si la aplicación maneja sesiones antes de autenticación, debe existir un mecanismo que autentique una nueva sesión antes de consumir los servicios.	No
4	Si se utiliza la gestión de sesión por estado, el servidor remoto usa tokens de acceso aleatorios para autenticar los pedidos del cliente sin requerir el envío de las credenciales del usuario en cada uno.	Si
5	Si se utiliza la autenticación basada en tokens sin estado, el servidor proporciona un token que se ha firmado utilizando un algoritmo seguro, no expone información sensible e implementa controles para evitar la suplantación.	NA
6	Cuando el usuario cierra la sesión en la aplicación, la sesión debe ser caducada en el servidor.	No
7	Las sesiones y los tokens de acceso expiran luego de un tiempo predefinido de inactividad.	No
8	Controlar las sesiones simultáneas sin importar los diferentes canales que provean el servicio.	NA
9	La aplicación informa al usuario acerca de los accesos a su cuenta desde dispositivos no registrados. El usuario es capaz de ver una lista de los dispositivos conectados y bloquear el acceso desde ciertos dispositivos.	No

Tabla 6-14.: Resultado: Manejo de sesiones y autorización.
Fuente: Elaboración propia.

Item	**Comunicación a través de la red**	Cumple
1	Implementar medidas de seguridad técnicas y canales seguros para protegerse contra el acceso no autorizado a información médica protegida electrónica que se transmite a través de una red de comunicaciones electrónicas.	Si
2	Las configuraciones del protocolo TLS siguen las mejores prácticas o tan cerca posible mientras que el sistema operativo del dispositivo lo permite y configura Ciphersuite que no estén considerados como débiles o vulnerables.	No
3	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.	No

Tabla 6-15.: Resultado: Comunicación a través de la red.
Fuente: Elaboración propia.

Item	**Interacción con la plataforma**	Cumple
1	La aplicación requiere la mínima cantidad de permisos.	Si
2	La aplicación no exporta funcionalidades sensibles vía esquemas de URL, salvo que dichos mecanismos estén debidamente protegidos.	Si
3	La aplicación no exporta funcionalidades sensibles a través de mecanismos IPC salvo que los mecanismos estén debidamente protegidos.	Si
4	JavaScript se encuentra deshabilitado en los WebViews salvo que sea necesario.	Si
5	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.	Si

Tabla 6-16.: Resultado: Interacción con la plataforma.
Fuente: Elaboración propia.

Item	**Calidad de código y configuración del compilador**	Cumple
1	La aplicación es firmada bajo las versiones 1 y 2 y provista con un certificado válido.	Si
2	La aplicación fue liberada en modo release y con las configuraciones apropiadas para el mismo. No debuggable y sin backup activo.	No
3	La aplicación captura y maneja debidamente las posibles excepciones.	Si
4	La lógica de manejo de errores en los controles de seguridad deniega el acceso por defecto.	Si

Tabla 6-17.: Resultado: Calidad de código y configuración del compilador.
Fuente: Elaboración propia.

Item	**Impedir el análisis dinámico y la manipulación**	Cumple
1	La aplicación detecta y responde a la presencia de un dispositivo rooted o con jailbreak, ya sea alertando al usuario o finalizando la ejecución de la aplicación.	No
2	La aplicación previene el debugging o detecta y responde al debugging de la aplicación. Se deben cubrir todos los protocolos.	No
3	La aplicación detecta y responde a modificaciones de ejecutables y datos críticos de la propia aplicación.	No
4	La aplicación detecta la presencia de las herramientas de ingeniería reversa o frameworks mas utilizados.	No
5	La aplicación detecta y responde al ser ejecutada en un emulador.	No
6	La aplicación implementa múltiples mecanismos de detección para los puntos del 9.1 al 9.6. Es de aclarar que a mayor cantidad y diversidad de mecanismos usados, mayor la resistencia.	No
7	La ofuscación es aplicada a las defensas del programa y código funcional, lo que a su vez impide la des-ofuscación mediante el análisis dinámico.	No

Tabla 6-18.: Resultado: Impedir el análisis dinámico y la manipulación.
Fuente: Elaboración propia.

6.1.3. Resumen resultado de las pruebas

Finalizada las pruebas de seguridad, se observó que la aplicación no cumple en un alto porcentaje los lineamientos de seguridad sugeridos para proteger la confidencialidad de la información de los pacientes. En la tabla **6-1** se puede observar el resumen del resultado de las pruebas.

En el lineamiento 1 Backend (Servicios), solo cumple 1/4 lineamientos sugeridos, con hallazgos del tipo listar directorios y archivos que contienen información personal de los pacientes. También se observó carencia de control de acceso a servicios restringidos y exposición de información de la plataforma utilizada. Los hallazgos reportados tienen un nivel de riesgo medio y bajo.

En el lineamientos 2: Almacenamiento de datos y la privacidad, solo cumple 8/14 lineamientos sugeridos, con hallazgos relacionados principalmente a la exposición de credenciales de acceso en texto claro en la url y la sandbox de la aplicación, junto con otros vectores que pueden exponer información personal y de salud del paciente. Los hallazgos reportados tienen un nivel de riesgo medio y bajo.

En el lineamientos 3: Criptografía, no cumple con ninguno de los 3 lineamientos sugeridos, los hallazgos están principalmente relacionado a la falta de cifrado en la información privada y de salud del paciente. Los hallazgos reportados tienen un nivel de riesgo medio y bajo.

En el lineamientos 4: Autenticación, solo cumple 1/5 lineamientos sugeridos, con hallazgos relacionados principalmente a la enumeración de usuarios, falta de controles contra la automatización y el uso de un segundo factor de autenticación. Los hallazgos reportados tienen un nivel de riesgo medio y bajo.

En el lineamientos 5: Manejo de sesiones y autorización, solo cumple 1/6 lineamientos sugeridos, con hallazgos relacionados principalmente a problemas de autorización como es el caso de la referencia insegura directa a objetos, al uso de tokens de sesión y cierre de sesión que permiten la exposición de información privada y de salud de los pacientes. Los hallazgos reportados tienen un nivel de riesgo alto y medio.

En el lineamientos 6: Comunicación a través de la red, solo cumple 1/3 lineamientos sugeridos, con hallazgos relacionados principalmente a la fortaleza en los protocolos de comunicaciones de seguridad utilizados y el uso de certificados. Este tipo de hallazgos permite que un atacante logre acceder a información privada y de salud de los pacientes en ataques del tipo hombre en el medio. Los hallazgos reportados tienen un nivel de riesgo medio y bajo.

En el lineamientos 7: Interacción con la plataforma, cumple 5/5 de los lineamientos sugeridos.

En el lineamientos 8: Calidad de código y configuración del compilador, cumple 3/4 de los lineamientos sugeridos, con un hallazgo relacionado al la opción de backup habilitada. Los hallazgos reportados tienen un nivel de riesgo bajo.

En el lineamientos 9: Impedir el análisis dinámico y la manipulación, no cumple ninguno de los 7 lineamientos sugeridos, con hallazgos relacionados principalmente a la manipulación del binario y el uso de dispositivo con root o en entornos emulados. Los hallazgos reportados tienen un nivel de riesgo medio.

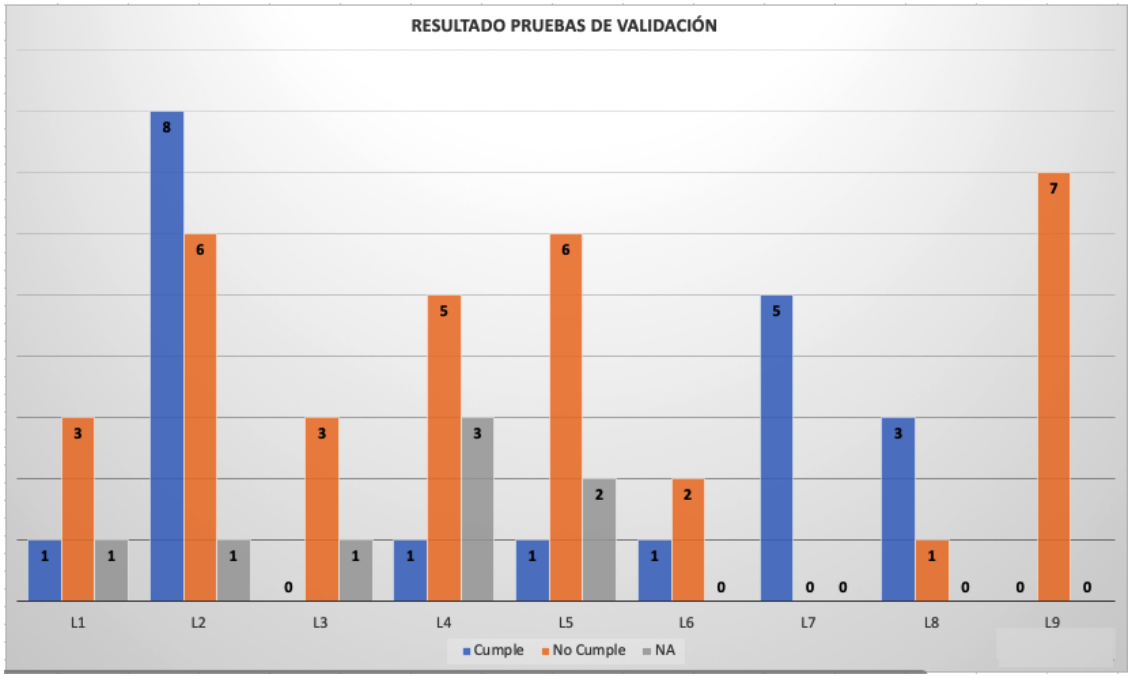

Figura 6-1.: Resumen resultado pruebas de seguridad.
Fuente: Elaboración propia.

7. Conclusiones y Recomendaciones

7.1. Conclusiones:

Lo expuesto a lo largo de este documento de tesis permitió llegar a las siguientes conclusiones:

- Bajo el análisis funcional realizado a diferentes aplicaciones móviles relacionadas al sector salud, se evidencia en el capítulo 4.2, el uso de datos de salud, personales y en algunos casos financieros del paciente, como parte del funcionamiento normal de las aplicaciones móviles al momento de consumir los servicios ofrecidos por una organización o empresa.

- Al revisar la normatividad Colombiana, se observó ítems muy generales relacionadas al cumplimiento de requisitos de seguridad de la información. Este comportamiento se puede observar en el Anexo F, que describe la relación entre las Leyes previamente identificadas y los lineamientos propuestos, dando como resultado que la mayor parte de las leyes requieren el cumplimiento total de los lineamientos.

- Las normativas o circulares como la expuesta por la Superintendencia Nacional de Salud, no cuenta con requerimientos de seguridad de la información claros, relacionados con la confidencialidad de la información de los pacientes que consumen servicios de empresas o entidades que gestionan información concerniente a la salud. En las circulares de la Superintendencia Nacional de Salud solo cuenta con algunos lineamientos para comunicaciones internas donde no existe la participación de los pacientes. Las normativas nacionales e internacionales que adoptan conceptos de seguridad de la información, son muy generales y relacionadas al que? se debe proteger y no el como? protegerlas o el como? validar si lo controles aplicados son eficientes y efectivos para proteger la información de los pacientes. Se llega a esta conclusión según el resultado de la definición de lineamientos de seguridad en el numeral 3.3.1 “Depuración de Lineamientos de seguridad consultados”.

- A nivel de estándares de seguridad para aplicaciones móviles, se identificó que OWASP Mobile Application Security es uno de los estándar mas completos, que mantiene actualizado constantemente sus repositorios y describe el que? se debe controlar y el como? validarlo. Este resultado es evidenciado en el numeral 3.2 “Estándares de seguridad para aplicaciones móviles”.

- En el desarrollo de la tesis se planteó inicialmente proponer los lineamientos de seguridad que deben ser validados en las aplicaciones móviles, antes de proponer el Modelado de Amenazas. Este planteamiento no fue el mas acertado debido a que generó un re proceso en el desarrollo de la tesis, que se puede reflejar en el mundo real al momento de certificar la seguridad en una aplicación móvil. El modelado de amenazas es la base para lograr identificar los posibles agentes y escenarios de amenaza, el tipo de información que puede llegar a comprometerse, y abarcar de una forma mas acertada los lineamientos de seguridad y las pruebas a realizar. Se llegó a esta conclusión al momento de construir el modelado de amenazas en el numeral 4.3 "Construcción del modelado de amenazas".

- Al aplicar la guía metodológica a la aplicación móvil seleccionada para la prueba de concepto, se detectaron vulnerabilidades críticas que comprometen la confidencialidad de la información de salud y personal de los pacientes. Los hallazgos de seguridad encontrados debieron ser identificados durante la etapa de desarrollo, como parte del ciclo de vida del software seguro o en la realización de pruebas de Ethical Hacking en ambientes productivos.

7.2. Recomendaciones:

- Definir y plantear un modelado de amenazas, antes de proponer los lineamientos de seguridad a cumplir durante el desarrollo de una nueva aplicación móvil o la creación de nuevas funcionalidades.

- Al momento de identificar vulnerabilidades de seguridad en la aplicación móvil como parte de pruebas previamente programadas o identificadas por un incidente de seguridad, se debe incluir como parte del informe las leyes o normas que se están incumpliendo, según la matriz ley vs lineamientos, con el fin de dar a conocer a nivel ejecutivo el impacto de no implementar o mejorar los controles de seguridad.

- Realizar periódicamente revisiones al modelado de amenazas, lineamientos de seguridad y pruebas de validación, para evitar obsolescencia en los procesos planteados inicialmente y realizar mejoras continuas.

- Para trabajos futuros se propone el planteamiento de una serie de pruebas de validación para cada uno de los lineamientos que cubre los dominios R1 - Gestión de la Seguridad y R2 - Arquitectura, Diseño y Modelado de Amenazas. El trabajo tendría como alcance la elaboración de un check list que permita corroborar a nivel de procesos, procedimientos o guías, el cumplimiento de cada uno de estos lineamientos.

- Mejorar la descripción de como realizar las pruebas de seguridad en aplicaciones móviles bajo el sistema operativo Android y iOS, es otro frente a mejorar para cada uno de los

lineamientos de seguridad propuestos en la tesis.

7.3. Resumen de logros por objetivo:

En la Tabla **7-1** se puede observar cada uno de los objetivos específicos planteado en la tesis para el cumplimiento del objetivo general, el resultado esperado según planteamiento aprobado en la propuesta de tesis y la ubicación en el documento de la evidencia que soporta el cumplimiento de cada objetivo.

Objetivo	Resultado esperado	Evidencia
Establecer los requerimientos mínimos de seguridad en el tratamiento de la información en las aplicaciones móviles del sector salud, que permiten dar cumplimiento a los requisitos de ley relacionados con la protección de datos personales en Colombia.	Un documento con un diagrama en bloques, que permita visualizar las exigencias de ley establecidas en Colombia y como son reforzados por medio de requerimientos de seguridad relacionado a la confidencialidad de la información, que se generarán a partir de otras leyes o estándares nacionales e internacionales, obteniendo al final del bloque un nuevo listado de requerimientos de seguridad más detallados o específicos.	Anexo A Anexo B Anexo C Anexo D Anexo E Anexo F Anexo G
Realizar un modelado de amenazas general que permita relacionar los requerimientos establecidos y los riesgos de seguridad informática asociados a cada uno de ellos.	Un documento que contiene el modelado de amenazas genérico a tener en cuenta en dispositivos móviles.	Anexo I
Definir pruebas dinámicas especializadas en seguridad informática, que permitan verificar el cumplimiento de los requerimientos, de acuerdo con el modelado de amenazas planteado.	Documento con el detalle de las pruebas técnicas especializadas, según los grupos previamente definidos.	Anexo J
Documentar una guía metodológica que describa el proceso para la verificación de seguridad en aplicaciones móviles, haciendo uso del listado de requerimientos y pruebas técnicas desarrolladas.	Documento final de la guía metodológica planteada en la tesis.	Se crea un documento definido como Guía Metodológica
Evaluar la implementación de la guía propuesta en una aplicación móvil relacionada al sector salud, para verificar su nivel de exposición a riesgos por perdida de confidencialidad de la información.	Documento que describirá los hallazgos encontrados y la información previamente definida en el formato del informe.	Se crea informe técnico clasificado de los hallazgos y se entrega a CSIETE como reporte al servicio de Alerta Temprana para notificar al dueño de la aplicación de los riesgos. En el capítulo 6 se puede observar de forma general el resultado de las pruebas.

Tabla 7-1.: Resumen de logros por objetivo.
Fuente: Elaboración propia.

A. Anexo: Legislación del Sector Salud vs Riesgos y Lineamientos

En la siguiente tabla se puede observar los riesgos de seguridad y lineamientos que se deben cumplir según las Leyes del Sector Salud en el ámbito de la confidencialidad de la información.

Leyes sector Salud enfocadas en seguridad de la información				Riesgo	Lineamientos
Resolución N. 1995 de 1999	**Capítulo III:** Organización y manejo del archivo de historias clínicas	**Artículo 16:** Seguridad del archivo de historias clínicas	El prestador de servicios de salud, debe archivar la historia clínica en un área restringida, con acceso limitado al personal de salud autorizado, conservando las historias clínicas en condiciones que garanticen la integridad física y técnica, sin adulteración o alteración de la información.	Integridad de la información	Fuera del alcance de la tesis
		Artículo 18: De los medios técnicos de registro y conservación de la historia clínica	Los programas automatizados que se diseñen y utilicen para el manejo de las Historias Clínicas, así como sus equipos y soportes documentales, deben estar provistos de mecanismos de seguridad, que imposibiliten la incorporación de modificaciones a la Historia Clínica una vez se registren y guarden los datos.	Integridad de la información	Fuera del alcance de la tesis
			En todo caso debe protegerse la reserva de la historia clínica mediante mecanismos que impidan el acceso de personal no autorizado para conocerla y adoptar las medidas tendientes a evitar la destrucción de los registros en forma accidental o provocada.	Integridad de la información Confidencialidad	- Backend - Criptografía - Autenticación - Manejo de Sesiones y autorización - Comunicación a través de la red - Interacción con la Plataforma - Calidad de Código y Configuración del Compilador - Impedir el Análisis Dinámico y la Manipulación
Ley 1266 2008 (Hábeas Data)	**Título I**	**Artículo 4.** Principios de la administración de datos	**c) Principio de circulación restringida:** Los datos personales, salvo la información pública, no podrán ser accesibles por Internet o por otros medios de divulgación o comunicación masiva, salvo que el acceso sea técnicamente controlable para brindar un conocimiento restringido solo a los titulares o los usuarios autorizados conforme a la presente ley;	Confidencialidad	- Backend - Almacenamiento de datos y la Privacidad - Criptografía - Autenticación - Manejo de Sesiones y autorización - Comunicación a través de la red - Interacción con la Plataforma - Calidad de Código y Configuración del Compilador - Impedir el Análisis Dinámico y la Manipulación
			f) Principio de seguridad: La información que conforma los registros individuales constitutivos de los Bancos de Datos a que se refiere la ley, así como la resultante de las consultas que de ella hagan sus usuarios, se deberá manejar con las medidas técnicas que sean necesarias para garantizar la seguridad de los registros evitando su adulteración, pérdida, consulta o uso no autorizado;	Integridad de la información Disponibilidad Confidencialidad	- Backend - Almacenamiento de datos y la Privacidad - Criptografía - Autenticación - Manejo de Sesiones y autorización - Comunicación a través de la red - Interacción con la Plataforma - Calidad de Código y Configuración del Compilador - Impedir el Análisis Dinámico y la Manipulación
			g) Principio de confidencialidad: Todas las personas naturales o jurídicas que intervengan en la administración de datos personales que no tengan la naturaleza de públicos están obligadas en todo tiempo a garantizar la reserva de la información, inclusive después de finalizada su relación con alguna de las labores que comprende la administración de datos, pudiendo solo realizar suministro o comunicación de datos cuando ello corresponda al desarrollo de las actividades autorizadas en la presente ley y en los términos de la misma.	Confidencialidad	Aplica para controles internos

Tabla A-1.: Leyes sector salud parte 1.
Fuente: Elaboración propia.

Ley 1581 del 2012	**Título III**	**Artículo 5.** Datos sensibles.	Para los propósitos de la presente ley, se entiende por datos sensibles aquellos que afectan la intimidad del Titular o cuyo uso indebido puede generar su discriminación, tales como aquellos que revelen el origen racial o étnico, la orientación política, las convicciones religiosas o filosóficas, la pertenencia a sindicatos, organizaciones sociales, de derechos humanos o que promueva intereses de cualquier partido político o que garanticen los derechos y garantías de partidos políticos de oposición **así como los datos relativos a la salud**, a la vida sexual y los datos biométricos.	Integridad de la información Disponibilidad Confidencialidad	Define como dato sensible los relacionados a la salud y por ende requiere un tratamiento especial, relacionado a las medidas de seguridad.
	Título VI	**Artículo 17.** Deberes de los Responsables del Tratamiento	**d)** Conservar la información bajo las condiciones de seguridad necesarias para impedir su adulteración, pérdida, consulta, uso o acceso no autorizado o fraudulento;	Integridad de la información Disponibilidad Confidencialidad	- Backend - Almacenamiento de datos y la Privacidad - Criptografía - Autenticación - Manejo de Sesiones y autorización - Comunicación a través de la red - Interacción con la Plataforma - Calidad de Código y Configuración del Compilador - Impedir el Análisis Dinámico y la Manipulación
			i) Exigir al Encargado del Tratamiento en todo momento, el respeto a las condiciones de seguridad y privacidad de la información del Titular;	Integridad de la información Disponibilidad Confidencialidad	Aplica para controles internos
		Artículo 18. Deberes de los Encargados del Tratamiento.	**b)** Conservar la información bajo las condiciones de seguridad necesarias para impedir su adulteración, pérdida, consulta, uso o acceso no autorizado o fraudulento;	Integridad de la información Disponibilidad Confidencialidad	- Backend - Almacenamiento de datos y la Privacidad - Criptografía - Autenticación - Manejo de Sesiones y autorización - Comunicación a través de la red - Interacción con la Plataforma - Calidad de Código y Configuración del Compilador - Impedir el Análisis Dinámico y la Manipulación
			j) Permitir el acceso a la información únicamente a las personas que pueden tener acceso a ella;	Confidencialidad	- Backend - Almacenamiento de datos y la Privacidad - Criptografía - Autenticación - Manejo de Sesiones y autorización - Comunicación a través de la red - Interacción con la Plataforma - Calidad de Código y Configuración del Compilador - Impedir el Análisis Dinámico y la Manipulación
Resolución 1531 de 2014 Modifica el artículo 10 de la Resolución 3374 de 2000	**Artículo 3.** Modifica el artículo 10 de la Resolución 3374 de 2000	**Artículo Decimo:** Procesos informáticos para la generación de datos de la prestación de servicios de salud	Las entidades administradoras de planes de beneficios están obligadas a garantizar la confiabilidad, seguridad y calidad de los datos sobre la prestación individual de servicios de salud; la entrega oportuna al Ministerio de Salud y Protección Social y la conformación de su propia base de datos sobre los servicios prestados. de manera individualizada.	Integridad de la información Disponibilidad Confidencialidad	- Backend - Almacenamiento de datos y la Privacidad - Criptografía - Autenticación - Manejo de Sesiones y autorización - Comunicación a través de la red - Interacción con la Plataforma - Calidad de Código y Configuración del Compilador - Impedir el Análisis Dinámico y la Manipulación
	Artículo 8. Seguridad de la información.	Seguridad de la información.	Todas las entidades obligadas a reportar deben garantizar el cumplimiento de los principios rectores para el tratamiento de datos, consagrados en la Ley 1581 de 2012, su decreto reglamentario y las normas que los modifiquen o sustituyan, en virtud de lo cual se hacen responsables de la privacidad, seguridad, confidencialidad y veracidad de la información suministrada y sobre los datos a los cuales tienen acceso	Integridad de la información Disponibilidad Confidencialidad	- Backend - Almacenamiento de datos y la Privacidad - Criptografía - Autenticación - Manejo de Sesiones y autorización - Comunicación a través de la red - Interacción con la Plataforma - Calidad de Código y Configuración del Compilador - Impedir el Análisis Dinámico y la Manipulación
Título V Circular Única SIC Protección de datos personales (Mar 2018)	**Capítulo Tercero:** Transferencia y transmisión de datos perosnales a terceros países	3.1. Estándares de un nivel adecuado de protección del país receptor de la información personal	**b)** Consagración normativa de principios aplicables al Tratamiento de datos, entre otros: legalidad, finalidad, libertad, veracidad o calidad, transparencia, acceso y circulación restringida, seguridad y confidencialidad.	Integridad de la información Disponibilidad Confidencialidad	Aplica para controles internos

Tabla A-2.: Leyes sector salud parte 2.
Fuente: Elaboración propia.

B. Anexo: Disposición SuperSalud enfocadas en seguridad de la información

En la siguiente tabla se puede observar los requisitos de seguridad dispuestos por la SuperSalud, enfocadas en seguridad de la información.

Disposición SuperSalud enfocadas en seguridad de la información				Resultado
Titulo I	Capitulo Primero	Seguridad técnica y jurídica para las comunicaciones electrónicas de la Superintendencia Nacional de Salud que requiere firma digital	garantizar un intercambio seguro y eficiente de los datos entre los vigilados y la entidad, y garantizar los atributos de autenticidad, integridad y no repudio de la información,	No aplica para el alcance de la tesis debido a que los requisitos en las comunicaciones son entre la supersalud y las entidades del sector salud y no entre el usuario final y las entidades del sector salud
Titulo II	Capitulo Primero	1.2.1.2. Plataforma tecnológica mínima requerida 1.2.1.2.5. Otros requerimientos técnicos.	b. Las entidades deberán poseer planes de contingencia y sistemas de respaldo y seguridad que le permitan a la entidad ante un daño grave, destrucción o robo de sus equipos de cómputo, regresar a su normal funcionamiento en un tiempo prudencial.	
			c. Establecer procedimientos de auditoría de sistemas que garanticen el cumplimiento de los parámetros aquí definidos, así como el íntegro y correcto manejo de la información por parte del ente vigilado y de su supervisor.	

Tabla B-1.: Disposición SuperSalud.
Fuente: Elaboración propia.

C. Anexo: Disposición SuperFinanciera enfocadas en seguridad de la información

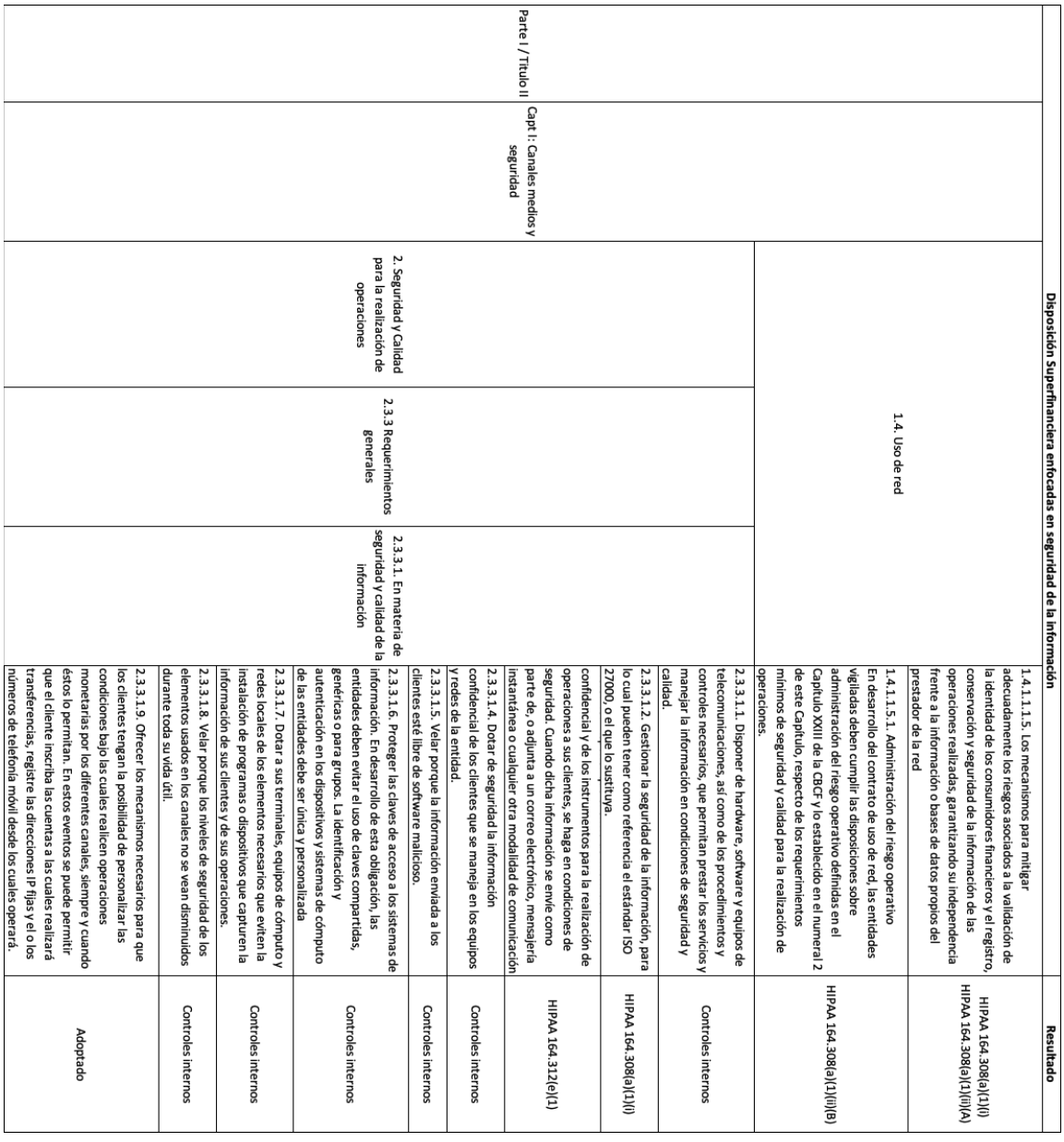

Tabla C-1.: Disposición SuperFinanciera parte 1.
Fuente: Elaboración propia.

					2.3.3.1.10. Ofrecer la posibilidad de manejar contraseñas diferentes para los instrumentos o canales, en caso de que éstos lo requieran y/o lo permitan.	Funcionalmente no es viable para este tipo de servicios
					2.3.3.1.11. Establecer los mecanismos necesarios para que el mantenimiento y la instalación o desinstalación de programas o dispositivos en las terminales o equipos de cómputo sólo pueda ser realizado por personal debidamente autorizado.	Controles internos
					2.3.3.1.12. Establecer procedimientos expeditos para el bloqueo de canales o de instrumentos para la realización de operaciones, cuando lo solicite el cliente, cuando existan situaciones o hechos que lo ameriten o después de un número de intentos de accesos fallidos, así como las medidas operativas y de seguridad para la reactivación de los mismos.	L4.5 OWASP
					2.3.3.1.13. Elaborar el perfil de las costumbres transaccionales de cada uno de sus clientes y definir procedimientos para la confirmación oportuna de las operaciones monetarias que no correspondan a sus hábitos.	Adoptado
					2.3.3.1.14. Realizar una adecuada segregación de funciones del personal que administre, opere, mantenga y, en general, tenga la posibilidad de acceder a los dispositivos y sistemas usados en los distintos canales e instrumentos para la realización de operaciones. En desarrollo de lo anterior, las entidades deben establecer los procedimientos y controles para el alistamiento, transporte, instalación y mantenimiento de los dispositivos usados en los canales de distribución de servicios.	Controles internos
					2.3.3.1.15. Definir los procedimientos y medidas que se deben ejecutar cuando se encuentre evidencia de la alteración de los dispositivos usados en los canales de distribución de servicios financieros	164.308(a)(1)(i)
					2.3.3.1.16. Sincronizar todos los relojes de los sistemas de información de la entidad involucrados en los canales de distribución. Se debe tener como referencia la hora oficial suministrada por el Instituto Nacional de Metrología de Colombia.	Controles internos
					2.3.3.1.17. Tener en operación sólo los protocolos, servicios, aplicaciones, usuarios, equipos, entre otros, necesarios para el desarrollo de su actividad.	Controles internos
					2.3.3.1.18. Contar con controles y alarmas que informen sobre el estado de los canales, y además permitan identificar y corregir las fallas oportunamente.	164.312(b)
					2.3.3.1.20. Considerar en sus políticas y procedimientos relativos a los canales de distribución, la atención a personas con discapacidades físicas, con el fin de que no se vea menoscabada la seguridad de su información.	Fuera del alcance
					2.3.3.1.21. Los establecimientos de crédito deben adoptar mecanismos que le permitan atender las operaciones de los consumidores financieros, por los canales que resulten necesarios y por las cuantías que determine razonables, para garantizar un nivel mínimo de prestación de sus servicios a los consumidores financieros, cuando la entidad opere fuera de línea.	Fuera del alcance
Parte I / Titulo II	Capt I: Canales medios y seguridad	2. Seguridad y Calidad para la realización de operaciones	2.3.3 Requerimientos generales	2.3.3.1. En materia de seguridad y calidad de la información		

Tabla C-2.: Disposición SuperFinanciera parte 2.
Fuente: Elaboración propia.

						2.3.4.9.1. Implementar los algoritmos y protocolos necesarios para brindar una comunicación segura.	L6.2 Qualys SSL Labs L6.3 y L6.4 OWASP
						2.3.4.9.2. Realizar como mínimo 2 veces al año una prueba de vulnerabilidad y penetración a los equipos, dispositivos y medios de comunicación usados en la realización de operaciones monetarias por este canal. Sin embargo, cuando se realicen cambios en la plataforma que afecten la seguridad del canal, debe realizarse una prueba adicional.	Adoptado
						2.3.4.9.3. Promover y poner a disposición de sus clientes mecanismos que reduzcan la posibilidad de que la información de sus operaciones monetarias pueda ser capturada por terceros no autorizados durante cada sesión.	HIPAA 164.312(e)(1)
					2.3.4.9. Internet	2.3.4.9.4. Establecer el tiempo máximo de inactividad, después del cual se debe dar por cancelada la sesión, exigiendo un nuevo proceso de autenticación para realizar otras operaciones.	L5.7 OWASP
						2.3.4.9.5. Informar al cliente, al inicio de cada sesión, la fecha y hora del último ingreso a este canal.	Adoptado
						2.3.4.9.6. Implementar mecanismos que permitan a la entidad financiera verificar constantemente que no sean modificados los enlaces (links) de su sitio web, ni suplantados sus certificados digitales, ni modificada indebidamente la resolución de sus DNS.	Fuera del alcance
						2.3.4.9.7. Contar con mecanismos para incrementar la seguridad de los portales, protegiéndolos de ataques de negación de servicio, inyección de código malicioso u objetos maliciosos, que afecten la seguridad de la operación o su conclusión exitosa.	Dengación - Fuera del alcance L1.2 OWASP L7.2-L7.5-L7.6-L7.7-L7.8 OWASP
						2.3.4.9.8. Las entidades que permitan realizar operaciones monetarias por este canal deben ofrecer a sus clientes mecanismos fuertes de autenticación.	L4.9 OWASP
					2.3.4. Requerimientos especiales por tipo de canal	2.3.4.11.1. Contar con mecanismos de autenticación de 2 factores para la realización de operaciones monetarias y no monetarias.	L4.9 OWASP
						2.3.4.11.2. Para operaciones monetarias individuales o que acumuladas mensualmente por cliente superen 2 SMMLV, implementar mecanismos de cifrado fuerte de extremo a extremo para el envío y recepción de información confidencial de las operaciones realizadas, tal como: clave, número de cuenta, número de tarjeta, etc. Esta información, en ningún caso, puede ser conocida por los proveedores de redes y servicios de telecomunicaciones ni por cualquier otra entidad diferente a la entidad financiera que preste el servicio a través de este canal.	L2.16 SF PCI DSS
						2.3.4.11.3. Cualquier comunicación que se envíe al teléfono móvil como parte del servicio de alertas o notificación de operaciones no requiere ser cifrada, salvo que incluya información confidencial.	L6.7 OWASP
					2.3.4.11. Banca Móvil	2.3.4.11.4. Para las operaciones monetarias individuales o que acumuladas mensualmente por cliente sean inferiores a 2 SMMLV y que no cifren la información de extremo a extremo, la entidad debe adoptar las medidas necesarias para mitigar el riesgo asociado a esta forma de operar, el cual debe considerar los mecanismos de seguridad en donde la información no se encuentre cifrada. La SFC puede suspender el uso del canal cuando se advierta que existen fallas que afecten la seguridad de la información.	L2.16 SF PCI DSS
						2.3.4.11.5. Contar con medidas que garanticen la atomicidad de las operaciones y eviten su duplicidad debido a fallas en la comunicación ocasionadas por la calidad de la señal, el traslado entre celdas, entre otras.	Controles internos
						2.3.4.11.6. Los servicios que se presten para la realización de operaciones a través de Internet, en sesiones originadas desde el dispositivo móvil, deben cumplir con los requerimientos establecidos en el subnumeral 2.3.4.9. de Internet.	Analizado en los puntos anteriores 2.3.4.9
				2.3.5 Requerimientos en materia de actualización de Software		2.3.5.1. Mantener tres ambientes independientes: uno para el desarrollo de software, otro para la realización de pruebas y un tercer ambiente para los sistemas en producción. En todo caso, el desempeño y la seguridad de un ambiente no pueden influir en los demás.	Adoptado
						2.3.5.2. Implementar procedimientos que permitan verificar que las versiones de los programas del ambiente de producción correspondan a las versiones de programas fuentes catalogadas.	Controles internos
						2.3.5.3. Cuando las entidades necesiten tomar copias de la información de sus clientes para la realización de pruebas, se deben establecer los controles necesarios para garantizar su destrucción, una vez concluidas las mismas.	Controles internos
						2.3.5.4. Contar con procedimientos y controles para el paso de programas a producción. El software en operación debe estar catalogado.	Controles internos
						2.3.5.5. Contar con interfaces para los clientes o usuarios que cumplan con los criterios de seguridad y calidad, de tal manera que puedan hacer uso de ellas de una forma simple e intuitiva.	Fuera del alcance
						2.3.5.6. Mantener documentada y actualizada, al menos, la siguiente información: parámetros de los sistemas donde operan las aplicaciones en producción, incluido el ambiente de comunicaciones; versión de los programas y aplicativos en uso; soportes de las pruebas realizadas a los sistemas de información; y procedimientos de instalación del software.	Controles internos

Tabla C-3.: Disposición SuperFinanciera parte 3.
Fuente: Elaboración propia.

D. Anexo: Disposición HIPAA enfocadas en seguridad de la información

Leyes definidas por HIPAA enfocadas en seguridad de la información			Resultado
Administrative Safeguards	164.308(a)(1)(i)	**Estándar: Security Management Process:** Implement policies and procedures to prevent, detect, contain, and correct security violations.	Adoptado
	164.308(a)(1)(ii)(A	**Risk analysis (Required).** Conduct an accurate and thorough assessment of the potential risks and vulnerabilities to the confidentiality, integrity, and availability of electronic protected health information held by the covered entity or business associate.	Adoptado
	164.308(a)(1)(ii)(B	**Risk management (Required).** Implement security measures sufficient to reduce risks and vulnerabilities to a reasonable and appropriate level to comply with §164.306(a).	Adoptado
	164.308(a)(1)(ii)(C	**Sanction policy (Required).** Apply appropriate sanctions against workforce members who fail to comply with the security policies and procedures of the covered entity or business associate.	Controles Internos
	164.308(a)(1)(ii)(D	**Information system activity review (Required).** Implement procedures to regularly review records of information system activity, such as audit logs, access reports, and security incident tracking reports.	Controles Internos
	164.308(a)(2)	**Standard: Assigned security responsibility.** Identify the security official who is responsible for the development and implementation of the policies and procedures required by this subpart for the covered entity or business associate.	Controles Internos
	164.308(a)(4)(i)	**Information Access Management**: Implement policies and procedures for authorizing access to EPHI that are consistent with the applicable requirements of subpart E of this part.	164.308(a)(4)(ii)(B)
	164.308(a)(4)(ii)(A	**Isolating health care clearinghouse functions (Required)**. If a health care clearinghouse is part of a larger organization, the clearinghouse must implement policies and procedures that protect the electronic protected health information of the clearinghouse from unauthorized access by the larger organization.	Controles Internos
	164.308(a)(4)(ii)(B	**Access authorization (Addressable).** Implement policies and procedures for granting access to electronic protected health information, for example, through access to a workstation, transaction, program, process, or other mechanism.	Adoptado
	164.308(a)(4)(ii)(C	**Access establishment and modification (Addressable).** Implement policies and procedures that, based upon the covered entity's or the business associate's access authorization policies, establish, document, review, and modify a user's right of access to a workstation, transaction, program, or process.	Controles Internos
	164.308(a)(5)(ii)(D	**Log-in monitoring (Addressable).** Procedures for monitoring log-in attempts and reporting discrepancies.	Adoptado
	164.308(a)(8)	**Standard: Evaluation.** Perform a periodic technical and nontechnical evaluation, based initially upon the standards implemented under this rule and, subsequently, in response to environmental or operational changes affecting the security of electronic protected health information, that establishes the extent to which a covered entity's or business associate's security policies and procedures meet the requirements of this subpart.	SF 2.3.4.9.2

Tabla D-1.: Disposición HIPAA parte 1.
Fuente: Elaboración propia.

Technical Safeguards	164.312(a)(1)	**Standard: Access control.** Implement technical policies and procedures for electronic information systems that maintain electronic protected health information to allow access only to those persons or software programs that have been granted access rights as specified in §164.308(a)(4).	Adoptado
	164.312(a)(2)(i)	**Unique user identification (Required).** Assign a unique name and/or number for identifying and tracking user identity.	Adoptado
	164.312(a)(2)(iii)	**Automatic logoff (Addressable).** Implement electronic procedures that terminate an electronic session after a predetermined time of inactivity.	L5.7 OWASP
	164.312(a)(2)(iv)	**Encryption and decryption (Addressable).** Implement a mechanism to encrypt and decrypt electronic protected health information.	Adoptado
	164.312(b)	**Standard: Audit controls.** Implement hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information.	Adoptado
	164.312(c)(1)	**Standard: Integrity.** Implement policies and procedures to protect electronic protected health information from improper alteration or destruction.	Adoptado
	164.312(c)(2)	**Implementation specification: Mechanism to authenticate electronic protected health information (Addressable).** Implement electronic mechanisms to corroborate that electronic protected health information has not been altered or destroyed in an unauthorized manner.	Fuera del alcance
	164.312(d)	**Standard: Person or entity authentication.** Implement procedures to verify	Adoptado
	164.312(e)(1)	**Standard: Transmission security.** Implement technical security measures to	Adoptado
	164.312(e)(2)(i)	**Integrity controls (Addressable).** Implement security measures to ensure that electronically transmitted electronic protected health information is not improperly modified without detection until disposed of.	Fuera del alcance
	164.312(e)(2)(ii)	**Encryption (Addressable).** Implement a mechanism to encrypt electronic protected health information whenever deemed appropriate.	164.312(a)(2)(iv)

Tabla D-2.: Disposición HIPAA parte 2.
Fuente: Elaboración propia.

E. Anexo: Disposición GDPR enfocadas en seguridad de la información

Leyes definidas por GDPR enfocadas en seguridad de la información				Resultado
Reglamentos	Numeral 35	información sobre su estado de salud física o mental pasado, presente o futuro. Se incluye la información sobre la persona física recogida con ocasión de su inscripción a efectos de asistencia sanitaria, o con ocasión de la prestación de tal asistencia, de conformidad con la Directiva 2011/24/UE del Parlamento Europeo y del Consejo (1); todo número, símbolo o dato asignado a una persona física que la identifique de manera unívoca a efectos sanitarios; la información obtenida de pruebas o exámenes de una parte del cuerpo o de una sustancia corporal, incluida la procedente de datos genéticos y muestras biológicas, y cualquier información relativa, a título de ejemplo, a una enfermedad, una discapacidad, el riesgo de padecer enfermedades, el historial médico, el tratamiento clínico o el estado fisiológico o biomédico del interesado, independientemente de su fuente, por ejemplo un médico u otro profesional sanitario, un hospital, un dispositivo médico, o una prueba diagnóstica in vitro.		R2.4 OWASP
	Numeral 49	Constituye un interés legítimo del responsable del tratamiento interesado el tratamiento de datos personales en la medida estrictamente necesaria y proporcionada para garantizar la seguridad de la red y de la información, es decir la capacidad de una red o de un sistema información de resistir, en un nivel determinado de confianza, a acontecimientos accidentales o acciones ilícitas o malintencionadas que comprometan la disponibilidad, autenticidad, integridad y confidencialidad de los datos personales conservados o transmitidos, y la seguridad de los servicios conexos ofrecidos por, o accesibles a través de, estos sistemas y redes, por parte de autoridades públicas, equipos de respuesta a emergencias informáticas (CERT), equipos de respuesta a incidentes de seguridad informática (CSIRT), proveedores de redes y servicios de comunicaciones electrónicas y proveedores de tecnologías y servicios de seguridad. En lo anterior cabría incluir, por ejemplo, impedir el acceso no autorizado a las redes de comunicaciones electrónicas y la distribución malintencionada de códigos, y frenar ataques de «denegación de servicio» y daños a los sistemas informáticos y de comunicaciones electrónicas.		L6 OWASP Comunicación a través de la red
	Numeral 83	A fin de mantener la seguridad y evitar que el tratamiento infrinja lo dispuesto en el presente Reglamento, el responsable o el encargado deben evaluar los riesgos inherentes al tratamiento y aplicar medidas para mitigarlos, como el cifrado. Estas medidas deben garantizar un nivel de seguridad adecuado, incluida la confidencialidad, teniendo en cuenta el estado de la técnica y el coste de su aplicación con respecto a los riesgos y la naturaleza de los datos personales que deban protegerse. Al evaluar el riesgo en relación con la seguridad de los datos, se deben tener en cuenta los riesgos que se derivan del tratamiento de los datos personales, como la destrucción, pérdida o alteración accidental o ilícita de datos personales transmitidos, conservados o tratados de otra forma, o la comunicación o acceso no autorizados a dichos datos, susceptibles en particular de ocasionar daños y perjuicios físicos, materiales o inmateriales.		R1.1 HIPAA 164.308(a)(1)(ii)(A) R1.3 'HIPAA 164.308(a)(1)(ii)(B)
Artículo 32	Seguridad del tratamiento	1. Teniendo en cuenta el estado de la técnica, los costes de aplicación, y la naturaleza, el alcance, el contexto y los fines del tratamiento, así como riesgos de probabilidad y gravedad variables para los derechos y libertades de las personas físicas, el responsable y el encargado del tratamiento aplicarán medidas técnicas y organizativas apropiadas para garantizar un nivel de seguridad adecuado al riesgo, que en su caso incluya, entre otros:	a) la seudonimización y el cifrado de datos personales;	HIPAA 164.312(a)(2)(iv)
			b) la capacidad de garantizar la confidencialidad, integridad, disponibilidad y resiliencia permanentes de los sistemas y servicios de tratamiento;	Adoptada parcialmente 'HIPAA 164.308(a)(1)(ii)(A)
			d) un proceso de verificación, evaluación y valoración regulares de la eficacia de las medidas técnicas y organizativas para garantizar la seguridad del tratamiento.	Adoptada parcialmente SF 2.3.4.9.2
		2. Al evaluar la adecuación del nivel de seguridad se tendrán particularmente en cuenta los riesgos que presente el tratamiento de datos, en particular como consecuencia de la destrucción, pérdida o alteración accidental o ilícita de datos personales transmitidos, conservados o tratados de otra forma, o la comunicación o acceso no autorizados a dichos datos.		HIPAA 164.308(a)(1)(ii)(A)

Tabla E-1.: Disposición GDPR.
Fuente: Elaboración propia.

F. Anexo: Leyes nacionales vs Lineamientos de seguridad

Normativa Legal	Backend					Almacenamiento de datos y la Privacidad														Criptografía			
	L1.1	L1.2	L1.3	L1.4	L1.5	L2.1	L2.2	L2.3	L2.4	L2.5	L2.6	L2.7	L2.8	L2.9	L2.10	L2.11	L2.12	L2.13	L2.14	L2.15	L3.1	L3.2	L3.3	L3.4
Resolución N. 1995 /Capítulo III/Articulo 18		X			X																X	X	X	X
Ley 1266 /Titulo I/Artículo 4/c)	X	X	X		X	X	X	X	X	X	X		X			X	X	X	X		X	X	X	X
Ley 1266 /Titulo I/Artículo 4/f)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo III/Artículo 7/3.	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo III/Artículo 7/6.		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo IV/Artículo 11/3.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo II/Artículo 4/g.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo VI/Artículo 17/d)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo VI/Artículo 18/b)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo VI/Artículo 18/j)	X	X	X		X	X	X	X	X	X	X		X			X	X	X	X		X	X	X	X
Resolución 1531/Articulo 3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Resolución 1531/Articulo 8	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Normativa Legal	Autenticación									Manejo de Sesiones y autorización									Comunicación a través de la red		
	L4.1	L4.2	L4.3	L4.4	L4.5	L4.6	L4.7	L4.8	L4.9	L5.1	L5.2	L5.3	L5.4	L5.5	L5.6	L5.7	L5.8	L5.9	L6.1	L6.2	L6.3
Resolución N. 1995 /Capítulo III/Articulo 18	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X		X		
Ley 1266 /Titulo I/Artículo 4/c)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo I/Artículo 4/f)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo III/Artículo 7/3.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo III/Artículo 7/6.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo IV/Artículo 11/3.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo II/Artículo 4/g.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo VI/Artículo 17/d)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo VI/Artículo 18/b)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo VI/Artículo 18/j)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Resolución 1531/Articulo 3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Resolución 1531/Articulo 8	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Tabla F-1.: Leyes vs Lineamientos parte 1.
Fuente: Elaboración propia.

Normativa Legal	Interacción con la Plataforma					Calidad de Código y Configuración				Impedir el Análisis Dinámico y la Manipulación						
	L7.1	L7.2	L7.3	L7.4	L7.5	L8.1	L8.2	L8.3	L8.4	L9.1	L9.2	L9.3	L9.4	L9.5	L9.6	L9.7
Resolución N. 1995 /Capítulo III/Articulo 18		X	X			X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo I/Artículo 4/c)		X	X			X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo I/Artículo 4/f)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo III/Artículo 7/3.						X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo III/Artículo 7/6.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1266 /Titulo IV/Artículo 11/3.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo II/Artículo 4/g.	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo VI/Artículo 17/d)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo VI/Artículo 18/b)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Ley 1581/Titulo VI/Artículo 18/j)		X	X			X	X	X	X	X	X	X	X	X	X	X
Resolución 1531/Articulo 3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Resolución 1531/Articulo 8	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Ley 1266 /Titulo I/Artículo 4/g)	Controles internos
Ley 1266 /Titulo VI/Artículo 17/3.	Controles internos
Ley 1581/Titulo II/Artículo 4/h.	Controles internos
Ley 1581/Titulo III/Artículo 5	Controles internos
Ley 1581/Titulo VI/Artículo 17/i)	Controles internos
Titulo V/Capítulo Tercero/3.1.	Controles internos

Tabla F-2.: Leyes vs Lineamientos parte 2.
Fuente: Elaboración propia.

G. Anexo: Requisitos y Lineamientos de seguridad

R	Requisito	Se recomienda su implemnentación para velar por el correcto cumplimiento del Lineamiento
L	Linemaiento	Recomendaciones a validar en la aplicación móvil

		Verificación detallada de requerimientos
R1		Gestión de la seguridad
R1.1	HIPAA 164.308(a)(1)(i)	Cuenta con un proceso de administración de seguridad que Implemente políticas y procedimientos para prevenir, detectar, contener y corregir infracciones de seguridad.
R1.2	HIPAA 164.308(a)(1)(ii)(A) GDPR Art32-1-b)	Cuenta con un análisis de riesgo. Llevar a cabo una evaluación precisa y exhaustiva de los posibles riesgos y vulnerabilidades a la confidencialidad, integridad y disponibilidad de la información de salud protegida electrónica en poder de la entidad o asociado comercial cubierto, incluyendo resilencia de los sistemas y servicios de tratamiento.
R1.3	HIPAA 164.308(a)(1)(ii)(B)	Cuenta con una Gestión de Riesgos mediante la implementar medidas de seguridad suficientes para reducir riesgos y vulnerabilidades a un nivel razonable y apropiado
R1.4	HIPAA 164.308(a)(5)(ii)(D)	Cuenta con procedimientos para monitorear intentos de ingreso y reportar discrepancias.
R1.5	SF 2.3.4.9.2	Realizar como mínimo 2 veces al año una prueba de vulnerabilidad y penetración a los equipos, dispositivos y medios de comunicación usados en la realización de operaciones por este canal. Sin embargo, cuando se realicen cambios en la plataforma que afecten la seguridad del canal, debe realizarse una prueba adicional.
R1.6	GDPR Art32-1-d)	Incluir un proceso de verificación, evaluación y valoración regulares de la eficacia de las medidas organizativas para garantizar la seguridad del tratamiento.
R1.7	HIPAA 164.312(a)(1)	Implementa políticas y procedimientos técnicos para los sistemas de información electrónicos que mantienen información de salud de los pacientes, para permitir el acceso solo a aquellas personas o programas de software a los que se les ha otorgado acceso.
R1.8	HIPAA 164.312(a)(2)(i)	Cuenta con una Identificación única del usuario, que permite asignar un nombre y / o número único para identificar y rastrear la identidad del usuario.
R1.9	HIPAA 164.312(b)	Implementa hardware, software y / o mecanismos de procedimientos que registran y examinan la actividad en los sistemas de información que contienen o usan información de salud protegida electrónicamente.
R1.10	HIPAA 164.312(c)(1)	Implementa políticas y procedimientos para proteger la información de salud electrónica de los pacientes contra alteraciones o destrucción inadecuadas.
R1.11	HIPAA 164.308(a)(4)(ii)(B)	Implementar políticas y procedimientos para otorgar acceso a información de salud electrónica protegida

Tabla G-1.: Lineamientos parte 1.
Fuente: Elaboración propia.

R2		Arquitectura, Diseño y Modelado de Amenazas
R2.1	OWASP	Todos los componentes se encuentran identificados y asegurar que son necesarios.
R2.2	OWASP	Los controles de seguridad nunca se aplican sólo en el lado del cliente, sino que también en los respectivos servidores remotos.
R2.3	OWASP	Se definió una arquitectura de alto nivel para la aplicación y los servicios y se incluyeron controles de seguridad en la misma.
R2.4	OWASP	Se identificó claramente la información considerada sensible en el contexto de la aplicación móvil.
R2.5	OWASP	Todos los componentes de la aplicación están definidos en términos de la lógica de negocio o las funciones de seguridad que proveen.
R2.6	OWASP	Se realizó un modelado de amenazas para la aplicación móvil y los servicios en el que se definieron las mismas y sus contramedidas.
R2.7	OWASP	La implementación de los controles de seguridad se encuentra centralizada.
R2.8	OWASP	Existe una política explícita 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.
R2.9	Jonathan Aldrich	Se debe realizar un diseño claro, contemplando un diseño general de la aplicación (e.j. Estructura arquitectónica) y definir opciones de diseño en código (e.j. protocolos, algoritmos, formatos de datos)
R2.10	Jonathan Aldrich	Expresar de forma clara las restricciones de seguridad, contemplando la definición de requisitos para el llamado de una interfaz y contar con las propiedades de confidencialida e integridad.
R2.11	Jonathan Aldrich	Verificar el diseño y seguridad en código mediante la: **unificación** del diseño y la implementación (por medio de lenguajes y librerías) y la **verificación** de la implememementación con el diseño (a través de análisis, tipos, modelo de comprobación, revisión)
R2.12	OWASP	Existe un mecanismo para imponer las actualizaciones de la aplicación móvil.
R2.13	OWASP	Se realizan tareas de seguridad en todo el ciclo de vida de la aplicación.
R2.14	Otros	Diseñar una arquietctura que permita al servidor identificar los flujos y el estado actual de la aplicación para evitar la manipulación de flujos que puede resultar en la evasión de contorles de seguridad
R2.15	SF 2.3.3.1.9	Ofrecer los mecanismos necesarios para que los clientes tengan la posibilidad de personalizar las condiciones bajo las cuales realicen operaciones por los diferentes canales, siempre y cuando éstos lo permitan. En estos eventos se puede permitir que el paciente registre el o los números de telefonía móvil desde los cuales operará 2.3.3.1.9.
R2.16	SF 2.3.5.1	Mantener tres ambientes independientes: uno para el desarrollo de software, otro para la realización de pruebas y un tercer ambiente para los sistemas en producción. En todo caso, el desempeño y la seguridad de un ambiente no pueden influir en los demás.
R2.17	OWASP	La aplicación implementa un "enlace al dispositivo" utilizando una huella del dispositivo derivado de varias propiedades únicas al mismo.
R2.18	SF 2.3.4.9.6	Implementar mecanismos que permitan verificar constantemente que no sean modificados los enlaces (links) de su sitio web, ni suplantados sus certificados digitales, ni modificada indebidamente la resolución de sus DNS.
L1		Backend (Servicios)
L1.1	OWASP	Configurar de forma correcta las cabeceras en el servidor web
L1.2	OWASP	Configurar controles que permitan asegurar que los usuarios solo pueden acceder a las URL´s para las cuales poseen autorización.
L1.3	OWASP	Las validaciones realizadas en los diferentes puntos de entrada de la aplicación móvil, deben ser igualmente validados de lado del Backend, para evitar inyecciones XSS, SQLi, XML o LDAP
L1.4	OWASP	Los logs registrados en el servidor no deben contener inofrmación considerada sensible o confidencial.
L1.5	OWASP	Implementar controles que eviten el listado de directorios y archivos en los servidores con los que interactua la aplicación (Hardening)

Tabla G-2.: Lineamientos parte 2.
Fuente: Elaboración propia.

L2		Almacenamiento de datos y la Privacidad
L2.1	OWASP	Las funcionalidades de almacenamiento de credenciales del sistema son utilizadas para almacenar la información sensible, como credenciales del usuario, Información de identificación personal (PII) y claves criptográficas.
L2.2	OWASP	No se escribe información sensible en los registros de la aplicación o en lugares de almacenamiento propios o externos del dispositivo móvil
L2.3	OWASP	Para aplciaciones Hibridas o Web APP no se debe exponer información sensible por método GET, se debe enviar solo por Método POST
L2.4	OWASP	No se comparte información sensible con servicios externos salvo que sea una necesidad de la arquitectura.
L2.5	OWASP	Se debe desactivar el caché del teclado en los campos de texto donde se maneja información sensible.
L2.6	OWASP	Se debe desactivar el portapapeles en los campos de texto donde se maneja información sensible.
L2.7	OWASP	No exponer información sensible mediante mecanismos entre procesos (IPC).
L2.8	OWASP	No exponer información sensible como contraseñas, datos personales o números de tarjetas de crédito a través de la interfaz o capturas de pantalla.
L2.9	OWASP	No incluir información sensible en los respaldos generados por el sistema operativo, como es el caso de resultados de laboratorio almacenados en la sandbox de la aplciación en formato pdf
L2.10	OWASP	La aplicación remueve la información sensible de la vista cuando la aplicación pasa a un segundo plano.
L2.11	OWASP	La aplicación no conserva la información sensible en memoria más de lo necesario y la memoria es limpiada luego de su uso.
L2.12	OWASP	La aplicación obliga a que exista una política mínima de seguridad en el dispositivo, como por ejemplo configurar un código de acceso.
L2.13	OASAM	Asignar permisos apropiados a los archivos y Base de datos gestionados por la aplicación
L2.14	OASAM	No utilizar credenciales quemadas en el código (Hard-coded)
L2.15	PCI DSS	Cumplimiento de PCI en caso de gestionar números de tarjeta de crédito o débito en algún flujo de la aplicación
L3		Criptografía
L3.1	HIPAA 164.312(a)(2)(iv) / GDPR	Implementar un mecanismo para proteger la información sensible personal y la relacionada a la salud, con la utilización de mecanismos de cifrado y de seudonimización.
L3.2	OWASP	La aplicación no depende únicamente de criptografía simétrica, con claves quemadas en el código. Se recomienda el de cifrado asimétrico
L3.3	OWASP	La aplicación no utiliza protocolos o algoritmos criptográficos que son considerados deprecados para aspectos de seguridad.
L3.4	OWASP	La aplicación no reutiliza una misma clave criptográfica para varios propósitos.
L4		Autenticación
L4.1	OWASP	Si la aplicación provee acceso a un servicio remoto, un mecanismo aceptable de autenticación como usuario y contraseña es realizado en el servidor remoto.
L4.2	OWASP	Existe una política de contraseñas y es aplicada en el servidor.
L4.3	OWASP	El servidor implementa controles, cuando credenciales de autenticación son ingresadas una cantidad excesiva de veces. Automatizacion
L4.4	Otros	El servidor implementa controles, cuando se intenta realizar enumeración de usuarios en el login, recueperar contraseña, recuperar usuario o cualquier otro endpoint donde requiera el uso del usuario, número de cédula o identificador de fácl adivinación.
L4.5	Otros	Establecer políticas para creación de usuarios que evite la fácil enumeración o adivinación de los mismos (Por ejemplo no utilizar números de cédula como usuario, nombre del usuario con números consecutivos pedro123 etc.)
L4.6	Otros	La contraseña y el usuario deben ser sometidos a funciones criptográficas que incluyan un valor salt, antes de ser enviados al servidor
L4.7	OWASP	La autenticación biométrica, si hay, no está atada a un evento del tipo "true" o "false" y almacena de forma segura las credenciales de acceso. keychain (iOS) o un keystore (Android).
L4.8	OWASP	Existe un mecanismo de segundo factor de autenticación (2FA) en el servidor y es aplicado consistentemente.
L4.9	OASAM	Controlar riesgos relacionados a la recperación de credenciales de acceso, atravez de funcionalidades de recordar credenciales.

Tabla G-3.: Lineamientos parte 3.
Fuente: Elaboración propia.

L5		**Manejo de Sesiones y autorización**
L5.1	**HIPAA 164.312(d)**	Implementar procedimientos para verificar que una persona o entidad que busca acceso a la información de salud electrónica protegida, se encuentra debidamente autorizado. (Referencia insegura directa a objetos)
L5.2	**SF 2.3.4.9.5.**	Informar al cliente, al inicio de cada sesión, la fecha y hora del último ingreso a este canal y la dirección IP. 2.3.4.9.5.
L5.3	**Otros**	Una sesión solo puede ser entregada al cliente por parte del servidor previa autenticación válida. Si la aplicación maneja sesiones antes de autenticación, debe existir un mecanismo que autentique una nueva sesión antes de consumir los servicios
L5.4	**OWASP**	Si se utiliza la gestión de sesión por estado, el servidor remoto usa tokens de acceso aleatorios para autenticar los pedidos del cliente sin requerir el envío de las credenciales del usuario en cada uno.
L5.5	**OWASP**	Si se utiliza la autenticación basada en tokens sin estado, el servidor proporciona un token que se ha firmado utilizando un algoritmo seguro, no expone información sensible e implementa controles para evitar la suplantación.
L5.6	**OWASP**	Cuando el usuario cierra la sesión en la aplicación, la sesión debe ser caducada en el servidor.
L5.7	**OWASP**	Las sesiones y los tokens de acceso expiran luego de un tiempo predefinido de inactividad.
L5.8	**Otros**	Controlar las sesiones simultáneas sin importar los diferentes canales que provean el servicio
L5.9	**OWASP**	La aplicación informa al usuario acerca de los accesos a su cuenta desde dispositivos no registrados. El usuario es capaz de ver una lista de los dispositivos conectados y bloquear el acceso desde ciertos dispositivos.
L6		**Comunicación a través de la red**
L6.1	**HIPAA 164.312(e)(1)**	Implementar medidas de seguridad técnicas para protegerse contra el acceso no autorizado a información médica protegida electrónica que se transmite a través de una red de comunicaciones electrónicas.
L6.2	**Qualys SSL Labs**	Las configuraciones del protocolo TLS siguen las mejores prácticas o tan cerca posible mientras que el sistema operativo del dispositivo lo permite y configura Ciphersuite que no estén considerados como débiles o vulnerables.
L6.3	**OWASP**	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.
L7		**Interacción con la Plataforma**
L7.1	**OWASP**	La aplicación requiere la mínima cantidad de permisos.
L7.2	**OWASP**	La aplicación no exporta funcionalidades sensibles vía esquemas de URL, salvo que dichos mecanismos estén debidamente protegidos.
L7.3	**OWASP**	La aplicación no exporta funcionalidades sensibles a través de mecanismos IPC salvo que los mecanismos estén debidamente protegidos.
L7.4	**OWASP**	JavaScript se encuentra deshabilitado en los WebViews salvo que sea necesario.
L7.5	**OWASP**	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.
L8		**Calidad de Código y Configuración del Compilador**
L8.1	**OWASP**	La aplicación es firmada bajo las versiones 1 y 2 y provista con un certificado válido.
L8.2	**OWASP**	La aplicación fue liberada en modo release y con las configuraciones apropiadas para el mismo (ej. non-debuggable).
L8.3	**OWASP**	La aplicación captura y maneja debidamente las posibles excepciones.
L8.4	**OWASP**	La lógica de manejo de errores en los controles de seguridad deniega el acceso por defecto.
L9		**Impedir el Análisis Dinámico y la Manipulación**
L9.1	**OWASP**	La aplicación detecta y responde a la presencia de un dispositivo rooted o con jailbreak, ya sea alertando al usuario o finalizando la ejecución de la aplicación
L9.2	**OWASP**	La aplicación previene el debugging o detecta y responde al debugging de la aplicación. Se deben cubrir todos los protocolos.
L9.3	**OWASP**	La aplicación detecta y responde a modificaciones de ejecutables y datos críticos de la propia aplicación.
L9.4	**OWASP**	La aplicación detecta la presencia de las herramientas de ingeniería reversa o frameworks mas utilizados.
L9.5	**OWASP**	La aplicación detecta y responde ante modificaciones de código o datos en su propio espacio de memoria.
L9.6	**OWASP**	La aplicación implementa múltiples mecanismos de detección para los puntos del 9.1 al 9.6. Nótese que a mayor cantidad y diversidad de mecanismos usados, mayor la resistencia.
L9.7	**OWASP**	La ofuscación es aplicada a las defensas del programa y código funcional, lo que a su vez impide la des-ofuscación mediante el análisis dinámico.

Tabla G-4.: Lineamientos parte 4.
Fuente: Elaboración propia.

H. Anexo: Modelado de Amenazas

Ref No	Elementos	Interación	Escenario de amenaza	S	T	R	I	D	E	Contramedidas	Owasp Top Ten
BK-01	Backend (Servicios)	Paciente	Un atacante logra acceder a información sensible o privada del paciente al lograr obtener las cookies de sesión de la aplicación basado en un ataque de inyección de javascript (XSS)	X			X			L1.1	A6 A7
Bk-02			Un atacante logra acceder a información sensible o privada del paciente al capturar tráfico basado en interceptación de las comunicaciones (Man in The Middle)		X		X		X		A6
BK-03			Un atacante logra acceder a información sensible o privada del paciente, por medio de ataques del tipo clickjacking	X			X				A6
BK-04			Un atacante logra acceder a información sensible o privada del paciente, al realizar un MIME-sniffing para enviar un ataque del tipo XSS	X			X				A6 A7
BK-05			Un atacante logra acceder a información sensible o privada del paciente, al inyectar código javascript en el contenido de la aplciación, para el cargue de páginas o contenido desde dominios de terceros con el fin de aprovechar por ejemplo ataques del tipo XSS o inyectar payloads que comprometan el servidor.	X	X		X				A6 A7
BK-06			Un atacante logra determinar el tipo y versiones de componentes o webservers que forman parte de la arquitectura de la aplicación, con el fin de enfocar un ataque mas efectivo y acceder a información sensible o privada del paciente				X				A3 A6
BK-07			Un atacante logra acceder a información sensible o privada del paciente, mediante la exploración de directorios y consumo directo de los servicios en el Backend, como usuario anónimo o autenticado.				X			L1.2 L1.5	A6
BK-08			Un atacante logra acceder a información sensible o privada del paciente, al lograr inyectar en el Backend código javascript, setencias sql, comandos de sistema operativo, LDAP o caracteres que no son permitidos según su lógica de negocio, definidas normalmente en el frontend	X	X		X	X	X	L1.3	A1
BK-09			Un atacante interno o externo logra obtener información sensible o privada del paciente, al acceder a los web logs registrados en el Backend, el cual fue comprometido previamente.				X			L1.4	A3

Tabla H-1.: Modelado de amenazas parte 1.
Fuente: Elaboración propia.

DP-01	Almacenamiento de datos y la Privacidad	Paciente	Un atacante logra acceder a información sensible o privada del paciente, al lograr capturar en código, o en archivos de configuración de la aplicación, claves de autenticación o llaves de cifrado y descifrado de información sensible	X			X			L2.1 L2.14	M2
DP-02			Un atacante logra acceder a información sensible o privada del paciente o de la aplicación, al monitorear los logs registrados en el dispositivo móvil	X			X			L2.2	M2
DP-03			Un atacante logra acceder a información sensible o privada del paciente, al estar expuesta por métodos HTTP inseguros al momento de incerceptar las comunicaciones.	X			X			L2.3	M3
DP-04			Un atacante logra acceder a información sensible o privada del paciente, al ser comparatida y transmitida por medio de librerías de terceros como parte funcional de la aplicación.	X			X			L2.4	M2 M3
DP-05			Un atacante logra acceder a información sensible o privada del paciente, mediante el análisis de la caché generada por la aplicación en la sandbox.	X			X			L2.5	M2
DP-06			Una aplicación mal intencionada logra capturar información sensible o privada del paciente gestionada por la aplicación, mediante la captura de datos guardada en el portapapeles	X			X			L2.6	M2
DP-07			Un atacante o malware logra acceder a información sensible o privada del paciente, que es almacenada en base de datos ubicada en la sandbox de la aplciación.	X	X		X			L2.7	M2
DP-08			Un atacante logra obtener información sensible o privada del paciente por medio de ataques del tipo shoulder surfing o por un malware que accede a capturas de pantallas de la aplicación	X			X			L2.8 L2.10	M2
DP-09			Un atacante logra acceder a información sensible o privada del paciente, al extraer del dispositivo un backup de la aplicación o acceder a backups existentes en otras unidades de almacenamiento.	X			X			L2.9	M2
DP-10			Un atacante o malware logra acceder a información sensible o privada del paciente, que es almacenada en memoria del dispositivo	X			X			L2.11	M2
DP-11			Un atacante logra acceder facilmente a las aplicaciones instaladas en un dispositivo robado o extraviado, debido a la no configuración de un código de acceso en el móvil.	X			X			L2.12	M1
DP-12			Un atacante o un malware logra acceder a información sensible o privada del paciente, al lograr extraer archivos o base datos de la sandbox de la aplicación, en móviles que no se encuenrtra con jailbreak o rooted, debido a una asignación errónea de permisos a los archivos.	X			X			L2.13	M1
DP-14			Un atacante logra obtener información finanaciera del paciente, en flujos de la aplicación que permiten realizar pagos, en canales o servicios que proveen terceros.	X			X			L2.15	M2 M3

Tabla H-2.: Modelado de amenazas parte 2.
Fuente: Elaboración propia.

CT-01	Criptografía	Paciente	Un atacante logra acceder a información sensible o privada del paciente, en transito entre el móvil y el Backend o almacenada en el dispositivo sin cifrar	X	X		X			L3.1	M3 M5
CT-02			Un atacante logra acceder a información sensible o privada del paciente que se encuentra cifrada, utilizando las llaves de descifrado quemadas en el código, expuestas en la mensajería , o almacenadas de forma no segura.	X	X		X			L2.14 L3.2	M3 M5 M9
CT-03			Un atacante logra acceder a información sensible o privada del paciente que se encuentra cifrada, explotando vulnerabilidades en las funciones criptográficas.	X	X		X			L3.3	M5 M9
CT-06			Un tacante logra obtener información sensible o privada del paciente, debido a que la llave criptográfica comprometida es utilizada para varios propósitos o en común para todos los usuarios	X	X		X			L3.4	M5
AT-01	Autenticación	Paciente	Un tacante logra obtener información sensible o privada del paciente, al lograr consumir servicios de forma anónima				X			L4.1	M4
AT-02			Un atacante logra acceder a la apliación y consumir los servicios del paciente, suplantando los identificadores del dispositivo (deviceID), el cual es utilizado como mecanismo de autenticación por parte del Backend. El acceso también se puede dar por robo o pérdida del móvil	X	X		X			L4.1 L4.7	M4
AT-03			Un atacante logra acceder a la apliación y consumir los servicios del paciente, manipulando la respuesta del servidor en el flujo de autenticación, mediante la inyección de una respuesta válida de login, con el fin de engañar a la aplicación.	X	X		X			L4.1 L4.2 L4.7	M4
AT-04			Un atacante logra acceder a la aplicación y consumir los servicios del paciente, al lograr adivinar facilmente la contraseña de la vícitma, utilizando un listado de contraseñas mas comunes. O generar una denegación de servicios masivo en el intento.	X			X	X		L4.2 L4.3	M4
AT-05			Un atacante por medio de diferentes técnicas, logra obtener un listado de usuarios existentes para el servicio, con el fin de potencializar otros atques y materializar una suplantación de identidad.	X			X			L4.4 L4.5	M4
AT-06			Un atacante logra capturar las credenciales de acceso de la vícitma, al realizar una interceptaicón de las comunicaciones o al encontrarse almacenadas de forma insegura en el dispositivo	X	X		X			L4.6	M4 M3 M2
AT-07			Un atacante con las credenciales de acceso de la víctima ya capturadas, logra realizar transacciones de riesgo desde la aplicación, obteniendo información sensible o privada del paciente.	X			X			L4.8	M4
AT-08			Un atacante logra acceder a información sensible o privada del paciente, al acceder a servicios protegidos con un segundo factor de autenticación, reutilizando un OTP que tiene un tiempo de vida muy alto.	X	X		X			L4.8	M4
AT-09			Un atacante logra acceder a información sensible o privada del paciente, al acceder a servicios protegidos con un segundo factor de autenticación, adivinando el OTP por medio de un ataque de fuerza bruta.	X	X		X			L4.8	M4
AT-10			Un atacante logra obtener o cambiar las credenciales de acceso a la aplicación, aprovechando debilidades en el mecanismo de recordar usuario o contraseña	X			X	X		L4.9	M4

Tabla H-3.: Modelado de amenazas parte 3.
Fuente: Elaboración propia.

SE-01	Manejo de Sesiones y autorización	Paciente	Un atacante logra acceder a información sensible o privada de otros pacientes, al lograr manipular las referencias a un objeto en la aplicación, para acceder a otros objetos sin autorización. Por ejemplo al realizar consultas, descargas de archivos o acceso a recursos privilegiados.	X	X		X		X	L5.1 L2.1 L2.11	M6
SE-02			Un atacante logra acceder a información sensible o privada del paciente, al consumir servicios con sesiones entregadas por la aplicación sin autenticar.	X			X			L5.3	M4
SE-03			Un atacante logra acceder a información sensible o privada del paciente, al lograr predecir el token de sesión y por ende acceder a los servicios de la aplicación	X			X			L5.4	M4
SE-04			Un atacante logra acceder a información sensible o privada del paciente, debido a que la aplicacón no utiliza una sesión y envía las credenciales de acceso en cada petición	X			X			L5.4	M4
SE-05			Un atacante logra obtener información confidencial del paciente, almacenada en el Json Web Token.				X			L5.5	M2 M4
SE-06			Un atacante logra acceder a información sensible o privada del paciente, mediante la alteración del los valores contenidos en el Json Web Token, al utilizar algoritmos inseguros en la firma.	X	X		X		X	L5.5	M4 M6
SE-07			Un atacante logra acceder a información sensible o privada del paciente, mediante la alteración del los valores contenidos en el Json Web Token, al lograr obtener el "secret" de la firrma por medio de un ataque de fuerza bruta o diccionario.	X	X		X		X	L5.5	M4 M6
SE-08			Un atacante logra acceder al sistema y obtener información sensible o privada del paciente, al robar un token mediante la interceptación de las comunicaciones y lo utiliza para suplantar al usuario legítimo	X	X		X			L5.5	M4
SE-09			Un atacante logra acceder a información sensible o privada del paciente, al consumir servicios con una sesión previamente cerrada por la vícitma desde la aplicación.	X			X			L5.6 L5.7	M4
SE-10			Un atacante logra acceder a información sensible o privada del paciente, al lograr acceder a la aplicación, mediante la extracción del token almacenado en base de datos sqlite.	X	X		X			L.2.1 L2.2	M4 M2
SE-11			Un atacante logra acceder a información confidnecial o privada del paciente, al secuestrar la sesión o ingresar de forma simultanea a la aplicación.	X			X			L5.8 L5.9 L5.2	M4

Tabla H-4.: Modelado de amenazas parte 4.
Fuente: Elaboración propia.

CR-01	Comunicación a través de la red	Paciente	Un atacante logra acceder a información sensible o privada del paciente, al capturar la información transmitida entre la aplicación y el servidor sin cifrar.		X		X			L6.1	M3
CR-02			Un atacante logra acceder a información condidencial y privada capturando información transmitida entre la aplicación y el servidor por medio de la degracación de los ciphersuite o de los protocolos de comunicación seguros.		X		X			L6.2	M3
CR-03			Un atacante logra acceder a información condidencial o privada del paciente, capturando información transmitida entre la aplicación y el servidor, utilizando certificados que no son confiables (autofirmados).		X		X			L6.3	M3
CR-04			Un atacante logra acceder a información condidencial o privada del paciente, capturando información transmitida entre la aplicación y el servidor, utilizando certificados que son de confianza pero no corresponden al sumistrado por el backend.		X		X			L6.3	M3
CR-05			Un atacante logra acceder a la aplicación aprovechando funciones para recuperar el usuarios/contraseña o realizar operaciones críticas protegidas por un segundo factor de autenticación, al suplantar la SIMCARD de la víctima, para obtener el OTP y continuar con el flujo. (SIM SWAP Attack)	X			X			L4.8 L6.1	M3
CR-06			Un atacante logra acceder a información sensible o privada del paciente, accediendo a servicios protegidos con un segundo factor de autenticación, mediante la extracción del mensaje de voz que deja el mecanismo de llamada telefónica para la entrega del OTP, al momento de no responder la llamada.	X			X			L4.8 L6.1	M3
CR-07			Un atacante logra acceder a información sensible o privada del paciente, al explotar vulnerabilidades encontradas en librerias de terceros (Heartbleed, ShellShock).	X	X		X			L6.2	M3
IP-01	Interacción con la Plataforma	Paciente	Un atacante logra acceder a información sensible o privada del paciente, basado en una inadecuada o inecesesaria asignación de permisos en la aplicación, hacia los recursos del sistema operativo		X		X			L7.1	M1
IP-02			Un atacante logra acceder a información sensible o privada del paciente, al momento de realizar una interceptación de las comunicaciones internas (IPC) de la app, para capturar información o inyectar valores para engañar a la víctima.	X	X		X			L7.2 L7.3 L2.7	M1
IP-03			Un atacante logra acceder a información sensible o privada del paciente, al lograr acceder a información sensible almacenada en la sandbox de la aplicación, mediante la inyección de java script en los webview		X		X			L7.4 L7.5	M1 M2

Tabla H-5.: Modelado de amenazas parte 5.
Fuente: Elaboración propia.

CC-01	Calidad de Código y Configuración del Compilador	Paciente	Un atacante logra acceder a información sensible o privada del paciente, mediante la inyección de un payload malicioso en la aplicación, teniendo en cuenta que solo fue firmada con la version 1 en android. (janus)	X	X		X			L8.1	M7
CC-02			Un atacante logra acceder a información sensible o privada del paciente o de la lógica del negocio, al firmar la aplicación con algoritmos considerados débiles o vulnerables.	X	X		X			L8.1	M7
CC-03			Un atacante logra obtener información relacionada a la lógica funcional de la aplciación, al encontrar en el código fuente, funciones y métodos con nombres descriptivos, útiles para planear el bypass de un control de seguridad				X			L8.2	M7 M9
CC-04			Un atacante logra acceder a información sensible o privada del paciente, mediante la modificación de valores en las variables utilizadas en ciertas funciones de la aplicación, en modo ejecución y sin modificar el binario.		X		X		X	L8.2	M7 M8
CC-06			Un atacante logra acceder a información sensible o privada del paciente, al tener acceso a los mensajes de debug expuestos por la aplicación, que no se desactivaron en el modo release.		X		X			L8.2 L2.2	M7
CC-08			Un atacante logra acceder a información sensible o privada del paciente o de la plataforma, al visualizar los mensajes de las diferentes excepciones no capturadas y tratadas por la aplicación.		X		X			L8.3	M7
CC-09			Un atacante logra acceder a información sensible o privada del paciente, al lograr el acceso a funciones sin la debida autorización, al generarse un error en los controles de seguridad.		X		X			L8.4	M7
CC-10			Un atacante logra acceder a información sensible o privada del paciente, al lograr inyecciones del tipo sentencias sql, Esquemas URL personalizados, códigos QR o llamadas IPC.		X		X			L1.3	M7
AR-01	Impedir el Análisis Dinámico y la Manipulación	Paciente	Un atacante logra acceder a información sensible o privada del paciente, evadiendo los controles de seguridad de acceso y autorización, mediante la instalación de herramientas de análisis dinámico de la aplicación y el acceso a recursos exclusivos para usuarios root.	X	X		X		X	L9.1 L9.4 L9.5	M9
AR-02			Un atacante logra acceder a información sensible o privada del paciente, activando intencionalmente el modo debug de la aplicación, alterando su código fuente	X	X		X			L9.2 L9.4 L9.6	M9 M8
AR-03			Un atacante logra acceder a información sensible o privada del paciente, evadiendo los controles de seguridad de acceso y autorización, mediante la manipulación del código fuente del binario.	X	X		X		X	L9.3 L9.4 L9.6	M8 M9
AR-05			Un atacante logra acceder a información sensible o privada del paciente, evadiendo los controles de seguridad basado en el análisis y manipulación de la lógica funcional, expuesta en el código fuente de la aplicación.	X	X		X		X	L9.1 L9.3 L9.4 L9.5 L9.6 L9.7	M8 M9

Tabla H-6.: Modelado de amenazas parte 6.
Fuente: Elaboración propia.

I. Anexo: Adecuar el ambiente de pruebas con dispositivo móvil físico y Máquina virtual Android

La descripción de la arquitectura de conexión para las pruebas de seguridad con dispositivo móvil físico se detallan a continuación:

I.0.1. Dispositivo móvil físico

- **Punto 1 - Dispositivo móvil:** El flujo de la información inicia con las peticiones realizadas desde la aplicación móvil. Para el buen desarrollo de las pruebas, se requiere que el dispositivo móvil cuente con los siguientes requisitos:

  Dispositivos Android:

  - El dispositivo debe estar rooted: Este tipo de permisos de gran privilegio sobre el sistema operativo de Android, son requeridos con el fin de poder explorar las diferentes rutas en el sistema operativo en busca de información sensible que comprometa la lógica de negocio o los datos personales de los usuarios gestionados por las aplicaciones y la posibilidad de utilizar herramientas especializadas que requieren de este tipo de privilegios para su ejecución. En la imagen **I-1** se observa el resultado de la validación realizada por la aplicación conocida como App Root Checker sobre un dispositivo móvil que ya está con privilegios de root.

  - El dispositivo debe estar *en modo modo depuración (Debug)* con el fin de conectar el dispositivo móvil al equipo de pruebas y poder utilizar la herramienta ADB que permite interactuar de diversas formas con el sistema operativo y poder realizar debug a las aplicaciones para un análisis más profundo en el comportamiento de la aplicación que se está analizando. En la imagen **I-2** se observa la activación del modo depuración en un Android. La ruta para la configuración es ***Ajustes/Opciones de desarrollador/Depuración de USB***.

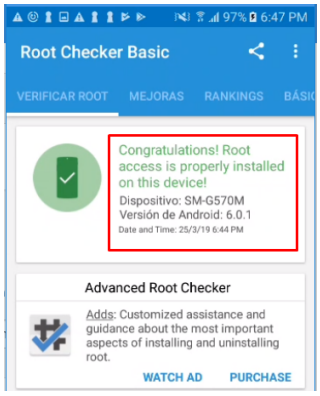

Figura I-1.: APP - Root Checker.
Fuente: Elaboración propia.

Figura I-2.: Activación - Depuración USB.
Fuente: Elaboración propia.

- Se debe *configurar la red wifi* expuesta por el Hotspot, con el fin de incluir la dirección IP y puerto del Proxy que también será posteriormente configurado en el OWASP Zap o Burpsuite en el equipo de pruebas. De esta forma se en ruta el tráfico que genere el dispositivo móvil para su análisis. En la imagen **I-3** se observa la configuración manual del proxy en la red identificada como Darkred que es generada por el Hotspot.
- Se debe instalar en el dispositivo móvil *el certificado de seguridad generado por el web proxy* (Zap, Burpsuite), con el fin de ver el tráfico sin el cifrado del protocolo HTTPS y evadir el control de algunos servidores que implementan en sus cabeceras como es el caso del Strict Transpor Security (HSTS). El certificado generado por el web proxy debe guardarse una de las carpetas de almacenamiento interno o externo del dispositivo móvil, luego se procede a realizar el llamado de dicho certificado desde la ruta ***Ajustes/Bloqueo y seguridad/Otros ajustes de seguridad/Instalar desde almacenamiento***. Con el fin de validar la instalación del certificado, se debe acceder a la opción de ***Ver certificados de***

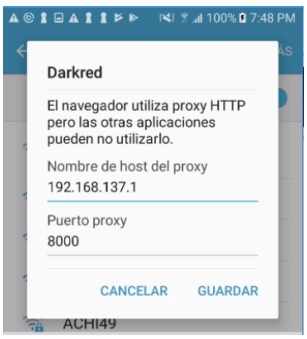

Figura I-3.: Configuración del proxy en red conectada al Hotspot.
Fuente: Elaboración propia.

seguridad y seleccionar la opción de ***Usuario***. En la imagen **I-4** se observa los certificados de OWASP Zap y de Burpsuite correctamente instalados.

Figura I-4.: Certificados de seguridad instalados por el usuario.
Fuente: Elaboración propia.

Dispositivos iOS:

- El dispositivo debe estar con jailbreak: Este tipo de permisos de gran privilegio sobre el sistema operativo iOS, son requeridos con el fin de poder explorar las diferentes rutas en el sistema operativo en busca de información sensible que comprometa la lógica de negocio o los datos personales de los usuarios gestionados por las aplicaciones y la posibilidad de utilizar herramientas especializadas que requieren de este tipo de privilegios para su ejecución.
- El dispositivo debe contar con un *servicio ssh)* en el dispositivo con el fin de poder conectar directamente al sistema operativo del iphone desde el equipo de pruebas. Esta conexión permite interactuar de diferentes formas con el sistema operativo como por ejemplo acceder a las rutas internas exclusivas del usuario root y acceder a la sandbox de las diferentes aplicaciones. En la imagen **I-5** se observa la conexión al servicio ssh desde un cliente ssh instalado en el iphone.

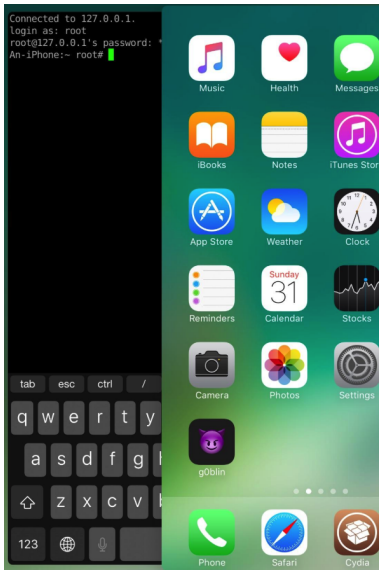

Figura I-5.: Conexión SSH al dispositivo móvil con jailbreak.
Fuente: Elaboración propia.

- Se debe *configurar la red wifi* expuesta por el Hotspot, con el fin de incluir la dirección IP y puerto del Proxy que también será posteriormente configurado en el OWASP Zap o Burpsuite en el equipo de pruebas. De esta forma se en ruta el tráfico que genere el dispositivo móvil para su análisis. La ruta para la configuración del proxy es ***/settings/Wifi/Darkred/Configure Proxy/***. En la imagen **I-6** se observa la configuración manual del proxy en la red identificada como Darkred que es generada por el Hotspot.

- Se debe instalar en el dispositivo móvil *el certificado de seguridad generado por el web proxy* (Zap o Burpsuite), con el fin de ver el tráfico sin el cifrado del protocolo HTTPS y evadir el control de algunos servidores que implementan en sus cabeceras, como es el caso del Strict Transpor Security (HSTS). El certificado generado por el web proxy debe enviarse preferiblemente por correo electrónico y abrirlo desde el dispositivo móvil. Luego de abrir el archivo se debe realizar la instalación desde el perfil ubicado en la siguiente ruta ***/Settings/General/Profiles/***. Para finalizar se debe habilitar el certificado como Certificado de Confianza en la ruta ***/settings/Wifi/Darkred/Configure Proxy/***. En la imagen **I-7** se

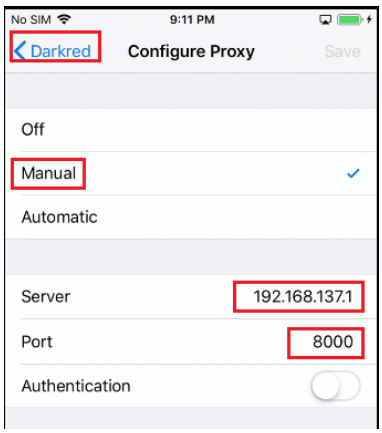

Figura I-6.: Configuración proxy en la red del Hotspot.
Fuente: Elaboración propia.

observa la instalación del certificado en el dispositivo móvil y en la imagen **I-8** la habilitación del certificado de OWASP Zap como certificado de confianza.

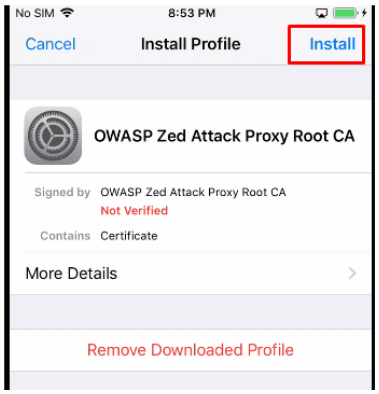

Figura I-7.: Instalación certificado OWASP Zap.
Fuente: Elaboración propia.

- **Punto 2 - Hotspot:** Cuando la información sale de la aplicación móvil, es dirigida al Hotspot expuesto en el equipo de pruebas. El Hotspot es una funcionalidad permite exponer un punto de acceso inalámbrico utilizando la conexión a internet de la red Wifi o red Lan del equipo de pruebas. De esta forma se logra redireccionar el tráfico generado por los dispositivos móviles conectados a dicho hotspot al webproxy instalado en el equipo de pruebas y lograr de esta forma la interceptación y modificación de la información. La técnica para la interceptación de las comunicaciones es conocida como ataque de Hombre en el Medio (Man in the Middle). Cada sistema operativo cuenta de forma nativa o por medio de software de terceros, la posibilidad de exponer este tipo de servicios. En la imagen **I-9** se observa el hotspot generado desde el sistema operativos Windows.

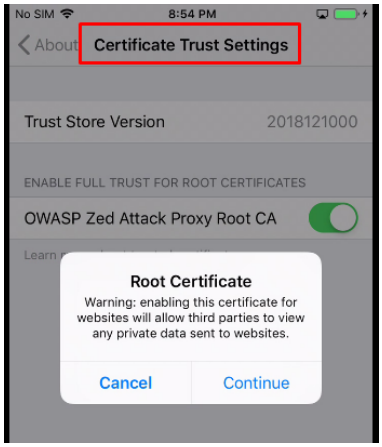

Figura I-8.: Habilitar confianza del certificado.
Fuente: Elaboración propia.

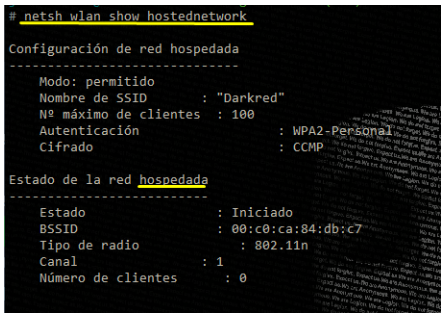

Figura I-9.: Hotspot nativo de Windows 10.
Fuente: Elaboración propia.

Punto 3 - Web proxy: En este punto el flujo de la información es en rutada hacia el Web proxy configurado en el equipo de pruebas. El web proxy permite interceptar las comunicaciones a través de protocolos Http, Https o websockets, brindando la posibilidad de modificar la mensajería intercambiada entra la aplicación móvil y el backend. Para lograr establecer un ataque de hombre en el medio y evadir controles relacionados a la validación de certificados de confianza, es necesario exportar el certificado de seguridad del web proxy (Owasp Zap o Burpsuite) e instalarlo en el dispositivo móvil tal y como se describe en el Punto 1. Para el caso de OWASP Zap, el certificado se puede exportar desde ***Tools / Options / Dynamic SSL Certificates / Save***. La configuración en el web proxy requerida para lograr la interceptación de las comunicaciones, se observa en la imagen **I-10**.

- **Punto 4 - Conexión a Internet:** . En este punto el flujo de la información debe ser dirigido a internet desde las conexiones habilitadas en el equipo de pruebas, utilizando la red física (Ethernet) o la red Inalámbrica (Wifi). Esta parte de la arquitectura no sufre ningún tipo de modificaciones debido a que la información ya fue alterada

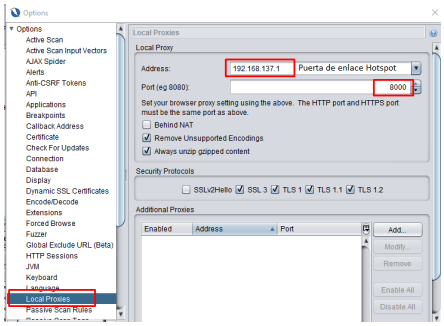

Figura I-10.: Configuración del proxy en OWASP Zap.
Fuente: Elaboración propia.

y queda en espera de ser procesada por el Servidor o Backend ubicado en internet. Este mismo proceso se presenta en sentido contrario, debido a que también podemos modificar la mensajería de respuesta por parte del Backend, con el fin de evadir posibles controles de seguridad configuradas en la aplicación.

I.0.2. Con máquina virtual

La descripción de la arquitectura de conexión para las pruebas de seguridad con máquina virtual Android, está conformada por una *Máquina nativa* que contiene dos (2) máquinas virtuales ver imagen **I-11**, una de ellas corresponde a la *Máquina Análisis* y la otra corresponde a la *Máquina Android*. A continuación se describe de forma general cada uno de los componentes:

- **Máquina Nativa:** La máquina nativa o anfitriona, puede ser un sistema operativo Windows, Linux o Mac, con un software que permita gestionar sistemas operativos virtualizados como es el caso de VirtualBox, VMWare entr otros. Es importante contar con capacidad de disco duro recomendada de 500 Gigas y 16 Gigas de RAM. Se debe tener en cuenta que las maquinas virtual van a requerir de buenos recursos para su buen funcionamiento.

- **Máquina Análisis:** La máquina de análisis es la encargada de recibir el tráfico generado por parte de la Máquina Android, con el fin de poder interceptar, analizar y modificar el tráfico entrante y saliente. En la máquina de análisis se debe configurar un servicio DHCP que se encargará de asignar una IP a la máquina Android, también se debe instalar un analizador de tráfico web como es eñ caso de un web proxy y de red como por ejemplo Wireshark. Es importante contar con el Android Studio para tener disponible las herramientas propias del framework y otras herramientas adicionales necesarias para realizar las diferentes pruebas de seguridad enfocada a las aplicaciones Android.

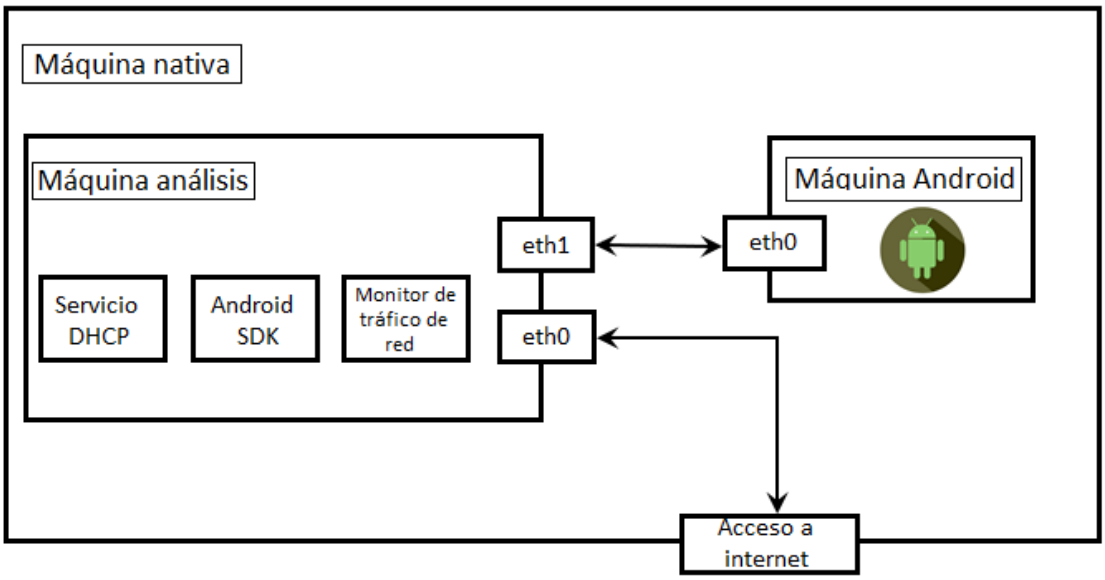

Figura I-11.: Esquema de conexión.
Fuente: Elaboración propia.

- **Máquina Android:** Esta máquina virtual tiene instalado el sistema operativo Andorid x86, que permite el acceso como root, el cual contendrá la aplicación o aplicaciones móviles a validar y la instalación de algunas aplicaciones no disponibles en el play store que apoyarán en la realización de las diferentes pruebas de seguridad. La máquina virtual es configurada de la tal manera que no tenga acceso directo a internet, obligando su paso por la máquina de pruebas para lograr consumir servicios expuestos fuera de su entorno.

J. Anexo: Detalle técnico de las pruebas de seguridad a realizar

A continuación se realiza una descripción general y detallada de las pruebas a realizar, con las herramientas de apoyo a utilizar

El alcance contempla los lineamientos definidos como:

- L1. Backend - Servicios.
- L2. Almacenamiento de datos y la Privacidad.
- L3. Criptografía.
- L4. Autenticación.
- L5. Manejo de sesiones y autorización.
- L6. Comunicación a través de la red.
- L7. Interacción con la Plataforma.
- L8. Calidad de Código y Configuración del Compilador.
- L9. Impedir el Análisis Dinámico y la Manipulación.

J.0.1. Backend - Servicios

J.0.1.1. Lineamiento L1.1

Control definido: Configurar de forma correcta las cabeceras en el servidor web.
Pruebas de Validación: La validación de este lineamiento consiste en identificar si el servidor responde con la cabeceras requeridas para evitar diferentes tipos de ataques que puede llegar a comprometer la seguridad del cliente final. La verificación de las cabeceras se debe realizar de la siguiente manera:

- Se debe interceptar las comunicaciones realizadas por la aplicación por medio de un webproxy, con el fin de capturar el response header en la mensajería y analizar los parámetros configurados.

- Validar que tenga configurada las siguientes cabeceras:
  - Set-Cookie: secure; HttpOnly.
  - Expires:Thu, 19 Feb 2019 08:52:00 GMT
  - Cache-Control: no-store, no-cache, must-revalidate.
  - Strict-Transport-Security: max-age=31536000; includeSubDomains; preload.
  - X-Frame-Options: SAMEORIGIN.
  - X-Content-Type-Options: nosniff.
  - X-XSS-Protection: 1; mode=block.
  - Referrer-Policy: strict-origin-when-cross-origin
  - Content-Security-Policy: "default-src'self'; script-src 'self'; style-src 'self'; img-src 'self'; frame-ancestors none.
  - X-Permitted-Cross-Domain-Policies: Use 'master-only'.
  - Access-Control-Allow-Origin: https://DominioPermitido.
  - Verificar si el valor "Server"no expone información de la plataforma, al igual que otro valores como "X-Powered-By"que exponen información sobre las tecnologías utilizadas.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite. http://sqlmap.org/ y https://portswigger.net/burp.
- Escaner online: https://securityheaders.com.

Lineamiento L1.2

Control definido: Configurar controles que permitan asegurar que los usuarios solo pueden acceder a las URLs para las cuales poseen autorización.
Pruebas de Validación: Con la enumeración de los servicios realizada en la fase de reconocimiento, se debe validar si alguno de los servicios puede ser accedido de forma directa sin ningún tipo de autenticación. El fin de esta validación es lograr consumir servicios en el backend sin utilizar la aplicación móvil. La verificación de este lineamiento se debe realizar de la siguiente manera:

- Listar cada uno de los servicios consumidos por la aplicación, contemplando todas las funcionalidades disponibles incluyendo los servicios de terceros que son llamados desde la aplicación y que forman parte del core del negocio.

- Si la aplicación no es nativa, se debe abrir cada una de las urls de los servicios consumidos en un navegador web desde un Computador, con el fin de validar su acceso y exposición de información.
- Si la aplicación es nativa, se debe reutilizar los request capturados y reenviarlos sin el uso de sesiones.
- Se debe interceptar la mensajería generada desde el navegador, para observar la información que se expone en el response que en ocasiones no sue visualiza en front.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite. http://sqlmap.org/ y https://portswigger.net/burp.
- Plugin Chrome/Firefox *"UserAgent Switcher"* que permite modificar el Agent enviado en el request, por uno comunmente utilizado por Android o iOS, para evitar las validaciones básicas del tipo de navegador en el Backend.

Lineamiento L1.3

Control definido: Las validaciones realizadas en los diferentes puntos de entrada de la aplicación móvil, deben ser igualmente validados de lado del Backend, para evitar inyecciones XSS, SQLi, XML, XXE o LDAP.
Pruebas de Validación: Las aplicaciones suelen tener controles de seguridad a nivel de Front para evitar posibles inyecciones de código en los diferentes puntos de entrada en la aplicación móvil. En algunos casos, estos controles no son aplicados en la capa de servicios. La verificación de este lineamiento se debe realizar de la siguiente manera:

- Se debe identificar los puntos de entrada en la aplicación, que pueden ser un formulario o variables que no se visualizan en el front pero viajan en el request.
- Con la ayuda de un web proxy se debe interceptar las comunicaciones e inyectar caracteres especiales con el fin de evaluar la respuesta del backend. Se puede iniciar con la inyección de unos simples tags html y seguir con una serie de payloads que tienen como fin encontrar inyecciones del tipo XSS, Sqli, XXE o Command Injection. Para esta rea se puede utilizar herramientas que automaticen el proceso.
- Es relevante validar si el Backend está aceptando caracteres que por reglas de negocio no están permitidos. En este caso no es primordial lograr la materialización de un ataque, lo importante es validar el cumplimiento del uso de listas Blancas definidos en el Front, apliquen de igual forma en el backend.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para interceptar y modificar la mensajería y tambien tienen la capacidad de realizar escaneos de vulnerabilidades del tipo inyecciones de código.
- SQLmap para inyecciones sql Link: http://sqlmap.org/ y https://portswigger.net/burp.
- XSStrike v3.1.4 para ataques avanzados de Cross Site Scripting. https://github.com/s0md3v/XSStrike.
- Commix herramienta utilizada para OS command. https://github.com/commixproject/commix injection.

Lineamiento L1.4

Control definido: Los logs registrados en el servidor no deben contener información considerada sensible o confidencial.
Pruebas de Validación: El foco de la validación consiste en identificar si los servidores que gestionan la información de los usuarios de la aplicación, almacenan datos sensibles en los registros de logs. Este tipo de validaciones se realizan principalmente como parte de las pruebas de seguridad del tipo caja blanca pero también pueden aplicar como pruebas de caja negra teniendo en cuenta que el servidor puede ser comprometido y accedido de forma remota. La verificación de este lineamiento se debe realizar de la siguiente manera:

- Se debe obtener el mayor número de registros de logs y someterlos a búsquedas específicas de información de salud o de datos personales. Las búsquedas pueden ser realizadas por medio de herramientas que automatizan el proceso o con scripts personalizados.
- Principalmente las pruebas deben ser realizada en ambientes de QA o pruebas que manejan información ficticia. Las pruebas en producción deben tener un procedimiento de entrega que evite la exposición accidental de información.

Herramientas de apoyo:

- grepWin es una herramienta entre otras que permite el uso de búsquedas por palabras claves o por medio de expresiones regulares.
- Lenguajes de programación como Python o Batch permite desarrollar scripts que ayudan a personalizar el proceso de búsqueda.

Lineamiento L1.5

Control definido: Implementar controles que eviten el listado de directorios y archivos en los servidores con los que interactúa la aplicación (Hardening).

Pruebas de Validación: Las pruebas consisten en identificar problemas de hardening en el backend, relacionada a los diferentes servicios web expuestos para el consumo de la aplicación. La verificación del cumplimiento del control se describe a continuación:

- Listar cada una de las urls de los servicios consumidos por la aplicación, contemplando todas las funcionalidades disponibles incluyendo los servicios de terceros que son llamados desde la aplicación y que forman parte del core del negocio.
- Abrir cada una de las urls de los servicios consumidos en un navegador web desde un Computador.
- Realizar un recorte de la ruta de las diferentes urls y observar la respuesta del servidor. El objetivo es validar la posibilidad de listar archivos y directorios en el servidor. Se debe revisar cada uno de los directorios y archivos listados con el fin de encontrar información personal o de salud de los usuarios de la aplicación o información que permita identificar nuevos vectores de ataque.

Herramientas de apoyo:

- Plugin Chrome/Firefox *"UserAgent Switcher"* que permite modificar el Agent enviado en el request, por uno comunmente utilizado por Android o iOS, para evitar las validaciones básicas del tipo de navegador en el Backend.

Almacenamiento de datos y la Privacidad

Lineamiento L2.1

Control definido: Las funcionalidades de almacenamiento de credenciales del sistema son utilizadas para almacenar la información sensible, como credenciales del usuario, Información de identificación personal PII, datos de salud o claves criptográficas.
Pruebas de Validación: Algunos desarrolladores almacenan información de forma persistente en el dispositivo móvil y no utilizan las APIs de seguridad proporcionadas por el sistema operativo para proteger la información, que puede estar relacionada a los usuarios, claves de cifrado y descifrado de datos entre otras.

- Se debe listar la información identificada y enumerada en la etapa de reconocimiento de funciones de la aplicación (Datos personales, de salud y financieros).
- Validar que la información listada anteriormente no se encuentre almacenada en texto claro en los siguientes puntos:
  - **Core Data (iOS):** Es un framework encargado de administrar la capa modelo de objetos en una aplicación. Para un almacenamiento persiste, se apoya de base de datos SQLite.

- **NSUserDefaults (iOS):** Esta clase permite almacenar configuraciones y propiedades que hacen referencia a los datos del usuario o de la aplicación.
- **Shared Preferences (Android):** Es usada normalmente para guardar información en pares de clave y valor.
- **Databases:** Es una base de datos que puede almacenar todo tipo de información propia de la aplicación. Normalmente suelen ser del tipo Sqlite o Realm y pueden estar o no cifradas.
- **Internal Storage:** Lugar asignado de forma exclusiva por el sistema operativo a una aplicación para su instalación y funcionamiento. Su acceso es restringido.
- **External Storage:** Lugar de almacenamiento externo al sistema operativo y no cuenta con restricciones de acceso.

- Se debe contar con un dispositivo Android Rooted y un dispositivo iOS con Jailbreak, con el fin de poder acceder al sistema operativo de los dispositivos móviles.

Herramientas de apoyo:

- Android Debug Bridg (adb) para acceder al sistema operativo de Android y estraer los archivos con el comando adb pull.
  link: http://adbshell.com/commands/adb-pull
- Cliente Shell Secure (SSH) para acceder al sistema operativo iOS.
- Cliente SQLite para validar las bases de datos extraídas del dispositivo móvil.
  Link: https://github.com/sqlitebrowser/sqlitebrowser/wiki
- Cliente SQLite3 para validar las bases de datos desde el mismo dispositivo.
  Link: https://play.google.com/store/apps/details?id=com.kanolato.sqlitehl=en

Lineamiento L2.2

Control definido: No se escribe información sensible en los registros de la aplicación en el sistema operativo.
Pruebas de Validación: Normalmente los desarrolladores activan el registro de logs en la aplicación, con el fin de realizar seguimiento a posibles fallos o errores generados durante el desarrollo. En ocasiones las aplicaciones salen a producción con los logs activos, generando el registro de información datos personales o de salud del paciente, que puede ser accedida por cualquier aplicación.

- Desde el Equipo de pruebas o máquina nativa, se debe lograr visualizar el Dispositivo Móvil o Máquina Android desde la aplicación ADB, por medio del comando:
  #adb devices.

- En Android se debe ejecutar el siguiente comando con el fin de visualizar el registro de logs generado por la aplicación que se está analizando:
  Linux: #adb -s idDispositivo shell logcat |grep NombreApp.
  Windows: #adb -s idDispositivo shell logcat |FindStr NombreApp.
  Desde interfaz gráfica se puede utilizar el Android Device Monitor, herramienta incluida en el Android Studio.
- En iOS se debe conectar por ssh al dispositivo móvil y ejecutar el comando:
  # tail -f /var/log/syslog.
- Con el comando ejecutado, se debe navegar en todas las funcionalidades de la aplicación e identificar que tipo de información se está registrando en los logs del dispositivo.

Herramientas de apoyo:

- Android Debug Bridg (adb) para la ejecución de comandos en el sistema operativo Android.
- Cliente Shell Secure (SSH) para la ejecución de comandos en el sistema operativo iOS.
- Logcat es una herramienta de linea de comandos incluida en el Android Studio y se encarga de mostrar los mensajes generados por el sistema operativo Android.
  link: https://developer.android.com/studio/command-line/logcat
- Android Device Monitor. Permite utilizar la herramienta de Logcat por medio de una interfaz gráfica.
  Link: https://developer.android.com/studio/profile/monitor

Lineamiento L2.3

Control definido: Para aplicaciones Híbridas o Web APP no se debe exponer información sensible por método GET, se debe enviar solo por Método POST.
Pruebas de Validación: En aplicaciones híbridas, móviles web o web móvil embebida, suelen enviar información privada del usuario en la url, registros de inscripción a la plataforma como es el caso de las credenciales de acceso u otro tipo de información como identificadores de sesión entre otros. Esta información puede ser capturada por un atacante que esté realizando un sniffing de la red, o en históricos de los servidores de la aplicación.

- Se debe interceptar las comunicaciones con un webproxy para capturar todas las peticiones realizadas por la aplicación.
- Navegar por todas las funcionalidades de la aplicación, incluyendo el registro y todos los flujos posibles que pueda tener la aplicación móvil.

- Analizar en el webproxy todas las urls con información enviada por método GET.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas por la aplicación móvil.

Lineamiento L2.4

Control definido: No se comparte información sensible con servicios externos salvo que sea una necesidad de la arquitectura.
Pruebas de Validación: Con el fin de mejorar la experiencia de usuario, algunas aplicaciones cuentan con librerías o sdk de terceros para realizar monitoreo de navegación o de uso de la aplicación, agregar nuevos servicios como por ejemplo la georeferenciación, incluir controles de seguridad, entre otros, que en ocasiones suelen compartir información privada de los usuarios sin ninguna protección.

- Se debe interceptar las comunicaciones realizadas entre la aplicación y cualquier servicio expuesto en internet, por medio de un webproxy bajo un esquema de ataque de hombre en el medio (MiTM).

- Se debe analizar e identificar posible información privada de los usuarios de la aplicación, que pueda ser compartida con terceros en cada una de las mensajerías capturadas previamente.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas por la aplicación móvil.

Lineamiento L2.5

Control definido: Se debe desactivar el caché del teclado en los campos de texto donde se maneja información sensible.
Pruebas de Validación: Algunas aplicaciones permiten la sugerencia de ingreso de datos en algunos formularios, como una forma de facilitar la interacción entre el usuario final y la aplicación. En ocasiones se permite el uso del cache del teclado en campos que contienen información privada, que puede ser accedida por un atacante con acceso físico al dispositivo o por un malware.

- Se debe identificar los formularios o punto de ingreso de información por parte del usuario en la aplicación, que gestione información privada como por ejemplo el usuario y la contraseña.

- El siguiente paso consiste en ingresar a la aplicación y validar si cada uno de los formularios o puntos de ingreso identificados previamente, realiza sugerencia para el ingreso de información.

Herramientas de apoyo:

- Esta es una actividad que debe ser realizada de forma manual en la aplicación.

Lineamiento L2.6

Control definido: Se debe desactivar el portapapeles en los campos de texto donde se maneja información sensible.
Pruebas de Validación: La opción copiar y pegar es muy utilizada por los usuarios para evitar diligenciar información en algunos campos. Permitir esta opción en campos de entrada con data sensible, da la posibilidad a que información privada sea capturada por malware desde el portapapeles. Un ejemplo es la aplicación Clipper en Android.

- Validar Se debe identificar los formularios o punto de ingreso de información por parte del usuario en la aplicación, que gestione información privada como por ejemplo el usuario y la contraseña.

- Con los campos identificados, se debe validar si la opción pegar se encuentra permitida en cada uno de ellos.

Herramientas de apoyo:

- Esta es una actividad que debe ser realizada de forma manual en la aplicación.

Lineamiento L2.7

Control definido: No exponer información sensible mediante mecanismos IPC.
Pruebas de Validación: El Content Providers tienen como propósito compartir entre aplicaciones, el acceso a diferentes tipos de datos estructurados. Algunas aplicaciones utilizan este servicio de Content Providers para almacenar diferentes tipos de información como cookies, llaves, información personal entre otras bajo el formato de base de datos como sqlite, que pueden llegar a exponer información privada a otras aplicaciones al no estar protegida correctamente.

En Android:

- Desde la Máquina de pruebas se debe validar la conexión con el dispositivo móvil o Máquina Android por medio de la herramienta ADB. La guía para realizar la conexión por medio de ADB se puede consultar en el siguiente Link: https://developer.android.com/
studio/command-line/adb?hl=es-419.

- Para realizar la validación del IPC relacionado al Contetn Providers se utilizará la herramienta Drozer, que permite identificar posibles vulnerabilidades en las comunicaciones IPC gestionadas por la aplicación bajo análisis. La siguiente guía del proyecto contiene lo necesario para la instalación y configuración tanto en la máquina de pruebas como en el dispositivo móvil o máquina Android. Link: https://github.com/mwrlabs/drozer.

- Con la comunicación debidamente establecida con Drozer, se debe realizar las siguientes actividades:
  - Identificar el paquete relacionado a la aplicación que se está analizando por medio del módulo app.package.info.
  - Identificar la superficie de ataque en la aplicación por medio del módulo app.package.attacksurface, validando si existen *Content Providers* potencialmente vulnerables. Se consideran vulnerables al ser configurados como: android:exported="true.[en] el Manifest.
  - Obtener mayor información sobre los Content Providers exportados por la aplicación por medio del módulo app.provider.info. Lo anterior es con el fin de poder identificar cuales no requieren ningún tipo de permisos para interactuar con ellos.
  - Enumerar los identificadores de recursos que permiten consumir los Content Providers, por medio del módulo scanner.provider.finduris. Normalmente los DBContentProvider inician con çontent://ƶ son esenciales para poder extraer la información de las bases de datos.
  - Extraer o modificar la información de las bases de datos por medio del módulo app.provider.query, tomando como base los identificadores de recursos enumerados en el paso anterior.
  - Validar la posibilidad de realziar inyecciones del tipo sql por medio del módulo scanner.provider.injection. El lograr explotar un sql injection en la aplicación, abre la posibilidad de evadir ciertos controles de seguridad, por ejemplo, el lograr acceder a información a tablas que están protegidas.

  Herramientas de apoyo:

  - El detalle de la ejecución de los comandos en Drozer se pueden consultar en el siguiente link: https://mobiletools.mwrinfosecurity.com/Using-Drozer-for-application-security-assessments/.
  - Android Debug Bridg (adb) para la ejecución de comandos en el sistema operativo Android.

En iOS:

- En el momento no se identificó herramientas que permitan interactuar con los IPC en sistemas operativos iOS. Este tipo de validaciones debe realizarse a nivel de código.

Lineamiento L2.8

Control definido: No exponer información sensible como contraseñas, datos personales o números de tarjetas de crédito a través de la interfaz o capturas de pantalla.
Pruebas de Validación: Las aplicaciones normalmente solicitan información a sus usuarios que puede ser considera privada. La información puede ser expuesta mediante el ingreso de datos en formularios gestionados por la aplicación, como por ejemplo las credenciales de acceso, información financiera por pagos de facturas, entre otras. También puede ser expuesta como resultado a las diferentes consultas que son realizadas contra el backend.

- Consultar la tabla 4.6 y 4.7 relacionada a la clasificación de los datos vs funcionalidades, con el fin de identificar en que parte de la aplicación, el dato no debe ser expuesto en su totalidad.
- Recorrer cada una de las funcionales con información privada previamente identificada y verificar si se puede observar el dato completo en pantalla. Por ejemplo validar si se encuentra sin enmascarar la contraseña, los números de Tarjeta de Crédito, flujos de registro, segundo factor de autenticación entre otros.
- Verificar si es posible de realizar una captura de pantalla o grabar vídeo en las pantallas que exponen la información.

Herramientas de apoyo:

- Esta es una actividad que debe ser realizada de forma manual en la aplicación.

Lineamiento L2.9

Control definido: No incluir información sensible en los respaldos generados por el sistema operativo, como es el caso de resultados de laboratorio almacenados en la sandbox de la aplicación en formato pdf.
Pruebas de Validación: Los dispositivos móviles Android y iOS permiten realizar copias de respaldo automáticas de la información y configuraciones de las aplicaciones. Ésta funcionalidad podría almacenar información sensible que podría ser extraída por un atacante desde el equipo de computo que la almacena, o puede ser utilizada como una opción para acceder a la información que se encuentra almacenada en la sandbox de las aplicaciones, en caso de no lograr el acceso desde el dispositivo móvil.

- La forma mas sencilla de validar si el backup está habilitado en una aplicación de Android, es por medio la validación del archivo Manifest android:allowBackup=”true”. La validación se puede realizar ejecutando el sigueinte comando *apktool d -s app.apk*.
- En Android, la validación dinámica propuesta por OWASP consiste en crear un backup por medio de ADB *adb backup -apk -nosystem ¡package-name¿* para luego convertirlo en un archivo comprimido .tar por medio

del siguiente comando *dd if=backup.ab bs=1 skip=24 — python -c ïmport zlib,sys;sys.stdout.write(zlib.decompress(sys.stdin.read()))» backup.tar*. Por último se descomprime el archivo .tar y se analiza que tipo de información se está almacenando.

- En iOS, la validación dinámica propuesta por OWASP consiste en validar el tipo de información que se está almacenando en los siguientes tres directorios: Documents/, Library/Application Support/, Library/Preferences/.

Se recomienda validar los siguientes repositorios de OWASP:

- Android - https://github.com/OWASP/owasp-mstg/blob/1.1.0/Document/0x05d-Testing-Data-Storage.mdtesting-backups-for-sensitive-data.
- iOS - https://github.com/OWASP/owasp-mstg/blob/1.1.0/Document/0x06d-Testing-Data-Storage.mdtesting-backups-for-sensitive-data.

Herramientas de apoyo:

- Android Debug Bridg (adb) para acceder al sistema operativo de Android y estraer losarchivos con el comando adb pull.
  link: http://adbshell.com/commands/adb-pull
- Apktool: Herramienta de reversing para aplciaciones Android.
  link: https://github.com/iBotPeaches/Apktool

Lineamiento L2.10

Control definido: La aplicación remueve la información sensible de la vista cuando la aplicación pasa a un segundo plano.
Pruebas de Validación: Una funcionalidad particular de los dispositivos móviles consiste en realizar una captura de pantalla cuando las aplicaciones entran en segundo plano. Estas capturas suelen ser almacenadas de forma local en el dispositivo y puede contener información sensible como resultados de laboratorio entre otros, que tienen el riesgo de ser extraídas por aplicaciones maliciosas o por técnicas forenses. Para validar el cumplimiento del control, se debe:

- Consultar la tabla 4.6 y 4.7 relacionada a la clasificación de los datos vs funcionalidades, con el fin de identificar en que parte de la aplicación el dato no debe ser capturado como imagen al momento de ingresar en segundo plano.
- El siguiente paso consiste en abrir varias aplicaciones incluyendo la aplicación objetivo y elegir la opción que permite visualizar las diferentes aplicaciones abiertas en segundo plano, con el fin de observar si existe una exposición de información sensible o privada.

- Una opción adicional consiste en encontrar la imagen que contiene la captura de pantalla visualizada en segundo plano. La ubicación puede variar dependiendo del sistema operativo. Por ejemplo en iOS la ubicación del archivo está en la ruta: /User/Containers /Data/Application/xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx/Library/Caches/Snapshots/ appication.com/.

Herramientas de apoyo:

- Esta es una actividad que debe ser realizada de forma manual en la aplicación.
- Android Debug Bridg (adb) para la ejecución de comandos en el sistema operativo Android, con el fin de buscar la captura de pantalla.
- Cliente Shell Secure (SSH) para la ejecución de comandos en el sistema operativo iOS.

Lineamiento L2.11

Control definido: La aplicación no conserva la información sensible en memoria más de lo necesario y la memoria es limpiada luego de su uso.
Pruebas de Validación: Para ciertos tipos de transacciones, las aplicaciones móviles utilizan la memoria del dispositivo para la ejecución de procesos particulares que suelen dejar información sensible que no es eliminada de forma correcta. El inadecuado uso de este recurso abre la puerta a que un atacante por medio de diferentes técnicas logre obtener datos en memoria con información que puede ser de carácter privado del usuario o información sensible de la aplicación como es el caso de las llaves de cifrado.

En Android:

- **La primera opción** consiste en realizar un dump de memoria relacionado a la aplicación analizada por medio del comando adb. El archivo generado .hprof se debe convertir en un formato específico para ser analizado con la herramienta Memory Analyzer Tool (MAT) de Eclipse. Este tipo de análisis es algo complejo por la canitdad de información obtenida. Los pasos a seguir son:
  - Se debe realizar una navegación en la aplicación con el fin de generar transacciones con información privada o confidencial.
  - Para realizar el dump de memoria de una aplicación en particular, se debe obtener el proceso con el que se está ejecutando la aplicación, por medio del comando: adb shell ps — grep "paquete".
  - Con el proceso identificado se peocede a obtener el dump de momria con extensión .hprof: adb shell am dumpheap PID /data/local/tmp/result.hprof.

  - El archivo obtenido en el paso anterior se debe converitr a un formato que pueda ser leido por la herramienta Memory Analyzer Tool (MAT) por medio del comando: hprof-conv result.hprof MAT.hprof.
  - Por último se procede a analizar el archivo MAT.hprof por medio de la herramienta MAT, en búsqueda de información sensible.

- **La segunda opción** consiste en utilizar la herramienta Fridump, que se encarga de realizar una volcado de memoria a un archivo para ser analizado posteriormente. El análisis tiene como fin la búsqueda de información sensible como es el caso de llaves de cifrado, contraseñas entre otros por medio de la lectura de cadenas de texto. Esta opción para sistemas operativos Android y iOS.
  - Se requiere tener instalado la herramienta de Frida. Con la herramienta instalada se procede a ejecutar el archivo de frida-server que debe estar ubicado en la ruta temporal /tmp/ según las indicaciones de instalación.
  - Se procede a validar le funcionamiento de frida mediante la ejecución del comando Frida-ps -U encargado de listar las aplicaciones abiertas con su respectivo ID de proceso.
  - En este punto se ejecuta fridump para la captura de información en memoria por medio del siguiente comando para android: python3 fridump.py -U -s -o RutaArchivo –max-size 2097152 NombrePaquete. Para iOS un ejemplo es el sigueinte python3 fridump.py -u NombrePaquete -s.
  - Una vez definido el tiempo de captura, se procede a extraer los strings del archivo por medio del comando strings -a file ¿strings.txt.
  - Por último se debe abrir el archivo strings.txt y validar su contenido en búsqueda de información privada de los usuarios o exclusiva de la aplicación.

Herramientas de apoyo:

- Fridump: https://github.com/Nightbringer21/fridump
- Frida: https://www.frida.re/docs/android/.
- ADB: Android Debug Bridg (adb), herramienta utilizada para la ejecución de comandos en el sistema operativo Android.
- HPROF Converter: https://stuff.mit.edu/afs/sipb/project/android/docs/tools/help/hprof-conv.html.
- Memory Analyzer Tool (MAT) : https://www.eclipse.org/mat/.

Lineamiento L2.12

Control definido: La aplicación obliga a que exista una política mínima de seguridad en el dispositivo, como por ejemplo configurar un código de acceso.
Pruebas de Validación: Se debe definir un política de seguridad o condiciones mínimas que debe tener el dispositivo móvil para permitir el acceso a la aplicación. Este tipo de controles permite resguardar en cierta forma al usuario final por pérdida del dispositivo y por una posible acción de atacante o malware. Se recomienda:

- Validar que el dispositivo cuente con un PIN o password de acceso al dispositivo.
- Definir la versión mínima permitida del Sistema Operativo Android para instalar o ejecutar la aplicación, con el fin de proteger al usuario de sistemas operativos obsoletos o con fallos de seguridad de alto impacto.
- Validar que la aplicación no se ejecuta si la opción de USB Debugging se encuentra activa en el dispositivo móvil.
- Verificar que la aplicación no se ejecuta si el dispositivo se encuentra Rooted.

Herramientas de apoyo:

- Esta es una actividad que debe ser realizada de forma manual en la aplicación.

Lineamiento L2.13

Control definido: Asignar permisos apropiados a los archivos y Base de datos gestionados por la aplicación.
Pruebas de Validación: La importancia de contar con los permisos apropiados en los archivos que se encuentran alojados en la sandbox de la aplicación, es evitar que aplicaciones externas logren extraerlos y por ende obtener información privada del usuario o parámetros de configuración propios de la aplicación que se puede traducir en un posible incidente de seguridad.

- Identificar en la sandbox de la aplicación, que archivos pueden contener información privada del usuario o de la plataforma y obtener el permiso asignado.
- Validar que los archivos identificados previamente en la sandbox de la aplicación, cuenten con los permisos adecuados que eviten su exfiltración por aplicaciones maliciosas. Los permisos de los archivos ubicados por ejemplo en las carpetas shared_prefs y databases, deben ser del tipo rw-rw- - - - o -rw- - - - - - -. Los permisos totales rwxrwxrwx o que incluyan permisos públicos son potencialmente susceptibles a ser extraídos o modificados por aplicaciones maliciosas.

Herramientas de apoyo:

- ADB: Android Debug Bridg (adb), herramienta utilizada para la ejecución de comandos en el sistema operativo Android.
- La página https://chmod-calculator.com/ cuenta con una calculadora que ayudará al entendimiento de los diferentes permisos asignados a los archivos.

Lineamiento L2.14

Control definido: No utilizar credenciales quemadas en el código (Hard-coded).
Pruebas de Validación: Considerando que este lineamiento se encuentra bajo la clasificación de pruebas del tipo estáticas, es relevante mencionar el gran beneficio que representa para las pruebas dinámicas, la verificación de código con un alcance específico. El enfoque no es el encontrar posibles XSS o SQli, lo principal es lograr encontrar las llaves de cifrado y descifrado de datos, el password para abrir una base de datos sqlite cifrada, credenciales de acceso para el consumo de servicios de terceros entre otras, que permiten generar nuevos vectores de ataque en las pruebas de seguridad tipo dinámicas en la aplicación.

- Identificar si la aplicación está utilizando cifrado simétrico para cifra y descifrar la mensajería (Request - Response) o datos sensibles como por ejemplo información relacionada a tarjetas de crédito, o documentos con información de salud que puedan ser intercambiados con el cliente.
- Identificar si existen bases de datos SQLite cifradas que requieren de una contraseña para su desbloqueo. en ocasiones estas claves son quemadas en el código y llamadas por las funciones al momento de almacenar información en dichas bases de datos.
- Validar sin en la mensajería capturada, la aplicación envía credenciales de acceso en texto claro al momento de autenticar con servicios de terceros expuestos en internet.
- Para validar le código fuente de la aplicación, se debe convertir el archivo con extensión .dex a un archivo .jar. Los archivos .jar pueden ser leídos fácilmente por medio de un cliente que interpreta el código fuente java y permite realizar búsquedas específicas en el código.
- Se debe validar los archivos .js que se encuentran alojados en el binario. Para esta validación se debe descomprimir la aplicación .apk que es realmente un .zip y realizar la búsqueda de datos de interés en los diferentes archivos que la contienen.

Herramientas de apoyo:

- DEX2JAR: Es una herramienta que permite convertir archivos .dex a archivos .jar Página de la herramienta: https://github.com/pxb1988/dex2jar

- JADX: Es una herramienta que permite decompilar de Dex a Java y sirve de cliente para visualizar el código fuente decompilado. Se puede cargar directamente el instalador .apk o archivos con extensión .jar. Página de la herramienta: https://github.com/skylot/jadx
- GREP: Herramienta que permite realizar búsquedas de palabras en diferentes archivos alojados en diferentes directorios.

Lineamiento L2.15

Control definido: Cumplimiento de PCI en caso de gestionar números de tarjeta de crédito o débito en algún flujo de la aplicación.
Pruebas de Validación: En algunas de las aplicaciones evaluadas con el fin de identificar que tipo de información intercambia con el usuario, se observó funcionalidades relacionadas a a pagos. Estas funcionalidades cuentan con un registro de Tarjetas de crédito y como requisito de seguridad, el usuario debe ingresar manualmente el código CVC. Por lo anterior, se debe garantizar el cumplimiento a las definiciones de PCI para evitar la exfiltración de datos financieros.

- Validar que el número de la tarjeta de crédito esté enmascarado al ser expuesta en pantalla. A nivel de mensajería y logs el número de tarjeta debe estar ilegible, es decir Cifrado, enmascarado o tokenizado.
- Validar que el código CVC y Fecha de vencimiento se encuentren enmascarados al momento de digitarlos en el campo de entrada, cifrados a nivel de mensajería y que no se encuentren almacenados en logs, caché o bases de datos SQLite.

Herramientas de apoyo:

- ADB: Android Debug Bridg (adb), herramienta utilizada para la ejecución de comandos en el sistema operativo Android. Se requiere para validar la información almacenada en la sandbox de la aplicación. (BD SQlite, caché, entre otros).
- Cliente Shell Secure (SSH) para la ejecución de comandos en el sistema operativo iOS. Se requiere para validar la información almacenada en la sandbox de la aplicación. (BD SQlite, caché, entre otros).
- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.

J.0.2. Criptografía

Lineamiento L3.1

Control definido: Implementar un mecanismo para proteger la información sensible personal y la relacionada a la salud, con la utilización de mecanismos de cifrado o de seudonimización.
Pruebas de Validación: Tomando como base la figura **4-6**, se define como información privada relacionada a datos personales y de salud a: Dictámenes médicos, reserva médica, así como los datos relativos a la salud y datos personales sensibles que afectan la intimidad del titular. Este tipo de información debe ser identificada y protegida adecuadamente, con el fin de evitar que personas no autorizadas logren acceder a dicha información e identificar a quien pertenecen. Los pasos a seguir son:

- Identificar los datos personales y de salud que gestiona la aplicación como son, Dictámenes médicos, reserva médica y los datos relativos a la salud. Para dar mayor claridad sobre el término "datos relativos a la salud", se toma como referencia la definición entregada por Lucia Nicole Cristea Uivaru en su tesis doctoral LA PROTECCIÓN DE DATOS DE CARÁCTER SENSIBLE EN EL ÁMBITO EUROPEO [83]. Los datos relativos a la salud consisten en:
  - Información relacionada a la salud pasada, presente o futura.
  - en personas sanas o enfermas.
  - Con enfermedades de carácter físico o psicológico.
  - Incluye la adicción al alcohol o las drogas.
  - Datos genéticos y muestras biológicas.
  - Datos que permiten conocer las dolencias o enfermedades que padecieron, padecen o podrán padecer.
  - Antecedentes de salud familiar, hábitos de vida, de alimentación y consumo.
- Validar si los datos personales o de salud cuentan con una protección adecuada (cifrado o mecanismos de seudonimización) al estar en tránsito o cuando son almacenados por alguna razón en el dispositivo móvil.

Herramientas de apoyo:

- ADB: herramienta utilizada para la ejecución de comandos en el sistema operativo Android. Se requiere para validar la información almacenada en la sandbox de la aplicación. (BD SQlite, caché, entre otros).
- Cliente Shell Secure (SSH) para la ejecución de comandos en el sistema operativo iOS. Se requiere para validar la información almacenada en la sandbox de la aplicación. (BD SQlite, caché, entre otros).

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.

Lineamiento L3.2

Control definido: La aplicación no depende únicamente de criptografía simétrica, con claves quemadas en el código. Se recomienda el de cifrado asimétrico.
Pruebas de Validación: Algunas aplicaciones utiliza funciones criptográficas con el fin de proteger la información sensible gestionada por la aplicación de posibles accesos no autorizados. Por ejemplo, el cifrado simétrico es utilizado normalmente para cifrar los request y response intercambiados entre la aplicación y el servidor para evitar la exposición de información o alteración de la misma. También es utilizada para cifrar data sensible como usuarios, contraseñas, tokens de sesión entre otros. Las validaciones a realizar son las siguientes cuando la aplicación utiliza criptografía simétrica:

- Se debe analizar la mensajería capturada con la herramienta webproxy e identificar si la data sensible o la misma mensajería se encuentra cifrada.

- Con el fin fortalecer las pruebas de seguridad dinámicas, se requiere en gran parte de realizar validaciones estáticas en el código fuente de la aplicación móvil, para identificar el uso de posibles funciones criptográficas. Para validar el código fuente de la aplicación en Android, se debe convertir el archivo con extensión .dex a un archivo .jar y analizarlo por medio de un cliente que interprete el código fuente java y realizar búsquedas específicas en el código relacionado a las clases java.security.* o javax.crypto.* en Android. Otra forma es realizar una búsqueda con la palabra "AES". que nos puede ayudar a identificar clases o funciones que la utilizan o por las palabras "IV". que el vector de inicialización y la palabra "Key" correspondiente a la clave. En iOS en el código fuente del binario, de deb buscar el archivo Commoncryptor.h que contiene los parámetros para las operaciones criptográficas simétricas.

- Seguir la definiciones descritas en los Lineamientos L2.11 y L2.14 con el fin de identificar la exposición de llaves o claves que están hardcoded en el código.

- Identificar si las variables IV y Key son diferentes con el fin de evitar una mayor exposición de información sensible. Se debe validar que el valor del vector de inicialización "IV" es dinámico en cada uso o por sesión.

Herramientas de apoyo:

- DEX2JAR: Es una herramienta que permite convertir archivos .dex a archivos .jar Página de la herramienta: https://github.com/pxb1988/dex2jar

- JADX: Es una herramienta que permite decompilar de Dex a Java y sirve de cliente para visualizar el código fuente decompilado. Se puede cargar directamente el instalador .apk o archivos con extensión .jar. Página de la herramienta: https://github.com/skylot/jadx
- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.

Lineamiento L3.3

Control definido: La aplicación no utiliza protocolos o algoritmos criptográficos que son considerados deprecados para aspectos de seguridad.
Pruebas de Validación: Se debe tener presente que algunos algoritmos definidos como seguros en el pasado, a hoy ya no lo son. Con el pasar del tiempo los algoritmos tienden a ser inseguros. Como parte de las buenas prácticas descritas por OWASP, se debe validar que la aplicación no utiliza algoritmos que son considerados vulnerables o cuentan debilidades que pueden exponer la información que se desea proteger. La forma de validar el cumplimiento del lineamiento es el siguiente:

- Teniendo en cuenta que las validaciones se deben realizar a nivel de código fuente, se requiere convertir el archivo con extensión .dex ubicado dentro del binario, a un archivo .jar y analizarlo por medio de un cliente que interprete el código fuente java.
- Validar que la aplicación no utiliza algoritmos criptográficos considerados vulnerables o débiles a la fecha. La búsqueda puede ser realizada desde el cliente de java, colocando como palabras clave los siguientes algoritmos: DES, 3DES, RC2, RC4, MD4, MD5, BLOWFISH o SHA1.

Herramientas de apoyo:

- DEX2JAR: Es una herramienta que permite convertir archivos .dex a archivos .jar Página de la herramienta: https://github.com/pxb1988/dex2jar
- JADX: Es una herramienta que permite decompilar de Dex a Java y sirve de cliente para visualizar el código fuente decompilado. Se puede cargar directamente el instalador .apk o archivos con extensión .jar. Página de la herramienta: https://github.com/skylot/jadx
- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.

Lineamiento L3.4

Control definido: La aplicación no reutiliza una misma clave criptográfica para varios propósitos.
Pruebas de Validación: En ocasiones por temas de simplicidad, a nivel de arquitectura utilizan las mismas llaves criptográficas para diferentes propósitos. Esta mala práctica extiende el campo de compromiso en caso de comprometerse las llaves de cifrado. Por ejemplo, si un organización cuenta con mas de dos aplicaciones móviles que proveen servicios diferentes, utilizan las mismas llaves de cifrado para proteger la data y la mensajería intercambiada entre la aplicación móvil y el backend. Por una vulnerabilidad en la aplicación A que expuso las llaves de cifrado, se compromete también el cifrado utilizado en la aplicación B. Otro escenario ocurre cuando se utilizan las mismas llaves de cifrado tanto en ambiente de Desarrollo, QA y Producción. Normalmente bajo lo escenarios de desarrollo y QA, las llaves de cifrado son compartidas y conocidas para un grupo mayor de personas y los ambientes no cuentan con la misma rigurosidad a nivel de implementaciones de seguridad. A continuación se exponen algunas validaciones a tener en cuenta.

- Las pruebas del tipo caja negra pueden ser complicadas para validar el cumplimiento del lineamiento, tiendo en cuenta que requiere información que no podrá ser obtenida por personal externo a la organización. Una de las pruebas consiste en validar si las llaves de cifrado utilizadas en los entornos de pruebas (Desarrollo, QA) son utilizadas en ambiente de producción.

- En caso de validar mas de una aplicación móvil de un mismo cliente, se debe validar si las llaves de cifrado son compartidas entre las aplicaciones móviles o incluso en los portales web.

J.0.3. Autenticación

Lineamiento L4.1

Control definido: Si la aplicación provee acceso a un servicio remoto, un mecanismo aceptable de autenticación como usuario y contraseña es realizado en el servidor remoto.
Pruebas de Validación: El proceso de autenticación en primer nivel debe requerir al menos el ingreso de un usuario y una contraseña que cumpla con las característica de Primer factor de autenticación, es decir algo que solo el usuario conoce y deben ser validadas de lado del servidor. En una de las aplicaciones analizadas utilizaban únicamente el número de cédula y la fecha de nacimiento para poder acceder a la aplicación de salud, datos que no son solo de conocimiento del usario de la aplicación. Los putos a validar son los siguientes:

- Con el fin de poder validar si el usuario y contraseña de acceso son creadas por el usuario, se debe analizar el proceso de registro y autenticación en la aplicación. No se recomienda utilizar el número de cédula o datos que estén relacionados al usuario o

que lo puedan identificar como fecha de nacimiento, fecha de expedición de la cédula entre otros.

- Validar si las credenciales de acceso ingresadas son enviadas al servidor para su verificación antes de permitir el acceso. Para este análisis se debe capturar el tráfico con un webproxy durante el flujo de autenticación. El fin de la prueba es validar que la aplicación no realiza autenticación con validación de credenciales de acceso de forma local para acceder a la aplicación.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.

Lineamiento L4.2

Control definido: Existe una política de contraseñas y es aplicada en el servidor.
Pruebas de Validación: El cumplimiento de una política de contraseñas, permite definir la complejidad con la que el usuario debe creer la contraseña para evitar ser adivinada por un ciber-delincuente. A continuación se describirá un conjunto de características que debe contener la política para evitar la adivinación de contraseñas.

- Es importante que las validaciones a realizar no solo estén definidas desde el frontend, cada regla de negocio definida para la creación de contraseñas también debe ser validada de lado del Backend.

- La prueba consiste en intentar registrar una contraseña con una longitud menor a 10 dígitos.

- Para validar su complejidad se debe identificar si la aplicación obliga al usuario a incluir mínimo tres de las siguientes cuatro reglas: Contener al menos una letra mayúscula, una minúscula, un dígito y un carácter especial.

- Intentar realizar un cambio de contraseña ingresando la misma o las últimas 5 contraseñas utilizadas. El fin de la prueba es validar hasta después de cuantas contraseñas o tiempo está configurada la aplicación, para su reuso.

- Intentar registrar una contraseña que contenga mas de 2 caracteres repetidos.

- Para validar si la contraseña tienen una fecha de expiración, se debe consultar con usuarios que lleven tiempo utilizando la aplicación o consultando con el negocio. La prueba dinámica no es viable realizarla por el corto tiempo que tiene las pruebas de seguridad.

- Validar si permite ingresar una contraseña que contenga el nickname de usuario, es decir si mi usuario es george22 mi contraseña sería george22 o george123 o george321 entre otras combinaciones, de igual forma sucede si el usuario corresponde al número de cédula. Un atacante que cuenta con una lista de usuarios válidos para la aplicación, la primeras contraseñas a utilizar para el ataque de diccionario es incluir el usuario con algunas combinaciones adicionales como 123, 321, el año entre un grupo mas de combinaciones que son comúnmente utilizada por los usuarios.

Herramientas de apoyo:

- La mayor parte de las validaciones se deben realizar consumiendo funcionalmente los servicios desde la aplicación.
- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.

Lineamiento L4.3

Control definido: El servidor implementa controles, cuando credenciales de autenticación son ingresadas una cantidad excesiva de veces. Automatización.
Pruebas de Validación: Cuando un atacante dispone de un conjunto de nombres de usuarios (nicknames) válidos en la aplicación, el paso a seguir es intentar adivinar la contraseña por medio del envío de múltiples request de forma automatizada. En este punto la aplicación o controles externos a ella deberán tener la capacidad de detectar y contener este tipo de ataques, conocidos como ataques de fuerza bruta o de diccionario y los controles normalmente utilizados son captchas y la implementación de soluciones WAF que por medio de reglas pueden identificar este tipo de peticiones por IP, número de peticiones vs tiempo entre otras. Las validaciones a realizar son:

- Para adivinar la contraseña de un usuario legítimo de forma automatizada, se debe conocer inicialmente cuantos intentos consecutivos permite la apelación antes de bloquear al usuario. Si se logra mas de 3 intentos consecutivos, la posibilidad de adivinar la contraseña aumenta. La prueba se puede realizar directamente en la aplicación sin ningún tipo de herramienta.
- El siguiente paso consiste en automatizar el ataque, utilizando herramientas de apoyo que permitan reenviar un request capturado con anterioridad por un webproxy y con la capacidad de cambiar durante cada petición el valor de la contraseña. El diccionario a utilizar en la automatización debe contemplar dos posibles vectores de ataque dependiendo de los controles detectados en la aplicación:
    - Ataque de diccionario tradicional: Este vector de ataque tiene como objetivo aprovechar la carencia de controles que eviten el ingreso excesivo de contraseñas

fallidas en la aplicación. El diccionario a utilizar corresponde a un usuario vs lista de posibles contraseñas. Un ejemplo del diccionario se observa en la tabla **J-1** :

Usuario	Contraseña
jorge	1234
jorge	4321
jorge	abc123
jorge	jorge1124

Tabla J-1.: Diccionario usuario vs contraseñas.
Fuente: Elaboración propia.

- Ataque Password spraying: Este vector de ataque es ideal para implementaciones que controlan el número de intentos de ingresos fallidos a la aplicación, mediante el bloqueo de la cuenta. El diccionario a utilizar corresponde a una lista de usuarios vs total de contraseñas permitidas antes de bloqueo. Este proceso es mas lento teniendo en cuenta que se debe esperar a que el usuario ingrese a la aplicación para realizar un reset del contador y continuar con el proceso. La viabilidad del ataque se da siempre y cuando se cuente con un buen listado de usuarios válidos en la aplicación. Un ejemplo del diccionario para una aplicación que bloquea el usuario al cuarto intento se observa en la tabla **J-2** :

Usuario	Contraseña
jorge	1234
jorge	4321
jorge	abcd123
maria	1234
maria	4321
maria	abcd123
johana	1234
johana	4321
johana	abcd123

Tabla J-2.: Diccionario usuario vs contraaeñas.
Fuente: Elaboración propia.

Herramientas de apoyo:

- Webproxy Burp Suite, utilizando la opción en la pestaña Intruder. En el siguiente enlace se puede observar un ejemplo de su uso.

https://support.portswigger.net/customer/portal/ articles/1964020-using-burp-to-brute-force-a-login-page. https://www.hackingarticles.in/ brute-force-website-login-page-using-burpsuite/

- Webproxy Owasp Zap, utilizando la función Fuzzing. En el siguiente video se puede observar un ejemplo de su uso. https://www.youtube.com/watch?v=tBXX_GAK7BU

- Lenguaje de programación en python, permite crear programar este tipo de ataques dirigidos a protocolos http y https. El programar el payload permite tener mayor flexibilidad para obtener un mejor resultado. Un ejemplo es el siguiente: https://github.com/Sanix-Darker/Brute-Force-Login.

Lineamiento L4.4

Control definido: El servidor implementa controles cuando se intenta realizar enumeración de usuarios en el login, recuperar contraseña, recuperar usuario o cualquier otro endpoint donde requiera el uso del usuario, número de cédula o identificador de fácil adivinación.
Pruebas de Validación: La enumeración de usuarios tiende a ser viable debido a que la prueba se enfoca en encontrar un usuario válido para la aplicación sin importar la contraseña utilizada. Normalmente los controles son establecidos en números de intentos fallidos de ingreso de contraseña por usuario y no por la cantidad de usuarios que solicitan el acceso en un determinado tiempo desde un mismo punto de origen. El contar con un grupo de usuarios válidos permite a un atacante fortalecer no solo la validación de posibles vulnerabilidades, también permite exponer la información de un grupo mayor de usuarios al explotar una vulnerabilidad en particular. Se debe tener en cuenta que la enumeración de usuarios no depende directamente de la automatización, la automatización aumenta el impacto en caso de lograr explotar una vulnerabilidad. Las validaciones a realizar son las siguientes:

- La primera prueba consiste en validar la respuesta entregada por el servidor, al momento de ingresar un usuario válido vs usuario no válido en la aplicación. En ocasiones el servidor responde indicando que el usuario no existe o está inactivo por medio de un mensaje en pantalla. Con base a la respuesta, se puede realizar validaciones automatizadas con el fin de lograr identificar posibles usuarios existentes en la aplicación. Las validaciones iniciales se realizan de forma funcional directamente en la aplicación.

- La segunda prueba consiste en validar el tamaño de la respuesta por parte del servidor, al momento de ingresar un usuario válido vs usuario no válido en la aplicación. En ocasiones las aplicaciones muestran un error genérico que no permite identificar si el usuario existe o no, pero si se puede presentar información adicional en el body del mensaje, permitiendo identificar a un usario por el tamaño que presenta el Response Body. Para realizar la validación se requiere revisar la mensajería intercambiada entre la aplicación y el servidor por medio de un webproxy.

- La tercera prueba consiste en validar la cabeceras entregadas en la respuesta por parte del servidor, al momento de ingresar un usuario válido vs usuario no válido en la aplicación. En ocasiones el servidor cambia algunos valores en la cabecera o agrega nuevos, cuando el usuario es válido o no. Por lo anterior es relevante realizar un comparativo a nivel de mensajería en las cabeceras de respuesta para identificar un posible cambio que ayude a establecer un diferencial que permita realizar la enumeración de usuarios. Para realizar la validación se requiere revisar la mensajería intercambiada entre la aplicación y el servidor por medio de un webproxy.

- La cuarta prueba consiste en validar el Round-trip time (RTT), que corresponde al tiempo en milisegundos que dura el mensaje en enviarse desde el punto de origen al destino y retornar, al momento de ingresar un usuario válido vs usuario no válido en la aplicación. Algunas aplicaciones cuentan con configuraciones y controles de seguridad que evitan enumerar usuarios basado en las validaciones expuestas anteriormente y no controlan los tiempos de respuesta por parte del servidor, teniendo en cuenta que al verificar un usuario válido el servidor invierte mas tiempo en responder que al ingresar un usuario no existente. Para realizar la validación se requiere revisar la mensajería intercambiada entre la aplicación y el servidor por medio de un webproxy.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entrela aplicación móvil y el backed.

Lineamiento L4.5

Control definido: Establecer políticas para creación de usuarios que evite la fácil enumeración o adivinación de los mismos (Por ejemplo no utilizar números de cédula como usuario, nombre del usuario con números consecutivos del tipo pedro123, entre otros.)
Pruebas de Validación: Para ataques de enumeración de usuarios se debe definir un formato que ayude al usuario a crear nicknames que no sean de fácil adivinación para un atacante. Las siguientes validaciones ayudan a identificar si la aplicación cuenta con una política para la creación de usuarios.

- Validar si la aplicación utiliza números de cédula o correos electrónicos como nickname. Al revisar algunas aplicaciones del sector salud, se observó que el nickname corresponde al número de cédula del paciente, permitiendo a un atacante conocer de forma fácil el primer dato para la autenticación de la víctima. Se debe recordar que el primer factor de autenticación corresponde a algo que yo sé (nickname y contraseña), el cual se ve degradado en su definición debido a que el número de cédula o correo electrónico es un dato muy fácil de obtener, por ejemplo en encuestas físicas o digitales, o en diferentes bases de datos que son de conocimiento por un grupo considerado de personas.

- Validar si la aplicación permite la creación de usuarios con las siguientes características:
  - Permite incluir caracteres especiales.
  - Permite crear usuarios solo numéricos o solo con alfabeto.
  - Permite crear usuarios con un tamaño menor a 5 caracteres.

Herramientas de apoyo:

- Las validaciones se deben realizar consumiendo funcionalmente los servicios desde la aplicación.

Lineamiento L4.6

Control definido: La contraseña y el usuario deben ser sometidos a funciones criptográficas que incluyan un valor salt, antes de ser enviados al servidor.
Pruebas de Validación: Una forma de evitar la exposición de la contraseña en la mensajería y los ataques de enumeración dirigidos a las contraseñas y los usuarios de la aplicación, consiste en aplicar funciones criptográficas robustas con valores aleatorios que ayuden a proteger los datos tanto en reposo como en tránsito. Los puntos a validar son:

- Se debe analizar el tráfico intercambiado entre la aplicación y el backend al momento de autenticarse, con el fin de observar si la contraseña y usuario viajan en texto claro o cifrada.

- La siguiente prueba consiste en capturar la mensajería al momento de realizar varias autenticaciones en la aplicación, con el fin de validar si el cifrado está utilizando un valor salt para generar un string aleatorio en cada autenticación. El no aplicar el salt en la función de cifrado, el atacante que obtenga la contraseña cifrada podrá reutilizar sin conocer su valor real.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entrela aplicación móvil y el backed.

Lineamiento L4.7

Control definido: La autenticación biométrica, si hay, no está atada a un evento del tipo "true" o "false" y almacena de forma segura las credenciales de acceso. keychain (iOS) o un keystore (Android).
Pruebas de Validación: Algunas aplicaciones aprovechan las funcionalidades nativas de biometría (Touch Id - Face Id) utilizadas por los dispositivos móviles, para facilitar el proceso de autenticación de los usuarios. El lineamiento se enfoca principalmente en la integración de la aplicación móvil con los servicios de biometría soportados por el móvil de forma segura. Las validaciones propuestas son:

- Se debe analizar la mensajería del flujo generado por la aplicación por medio de un webproxy, al momento de realizar el proceso de registro de la biometría. El análisis de la mensajería permite conocer la forma en que la aplicación se integra con el servicio de biometría, con el fin de identificar si las funciones de cifrado se exponen durante el proceso de registro o si generan datos que pueden ser capturados y suplantados en otro dispositivo. Los resultados dependen de la arquitectura propuesta por la aplicación móvil.

- Se debe analizar la mensajería del flujo generado por la aplicación por medio de un webproxy, al momento de realizar el proceso de autenticación con biometría. El análisis de la mensajería permite identificar la forma en que la aplicación se integra con dicho servicio. Uno de los puntos a validar es identificar la forma en que la aplicación responde ante el ingreso de una una dato biométrico válido para la aplicación, de uno no válido. Si el servicio responde con un valores booleano "true" o "false" o las respuestas son del tipo "success" o "fail" se debe interceptar las comunicaciones y cambiar el valor de false a true o de fail a success según corresponda, con el fin de lograr el acceso, ingresando un dato biométrico no válido.

- El siguiente punto a validar consiste en identificar si la aplicación está almacenando los valores de autenticación o llaves criptográficas en bóvedas seguras proporcionada por el sistema operativo, keychain (iOS) o un keystore (Android), al momento de realizar la autenticación biométrica. Para esta validación se debe recurrir al análisis estático de la aplicación, con el propósito de identificar las funciones que intervienen en el proceso y lograr por medio de los diferentes análisis realizados en el punto 5.3.2 (Almacenamiento de datos y la Privacidad), ubicar un posible almacenamiento no seguro de los datos.

- Validar si la aplicación está utilizando criptografía simétrica para cifrar y descifrar los datos de autenticación biométrica. En este caso se debe realizar un análisis en tiempo real en memoria con el fin de capturar las llaves de cifrado con herramientas de debug e identificar si las llaves son generadas por sesión o es igual para todos los usuarios. Para el análisis en memoria se debe utilizar las diferentes técnicas descritas en el Lineamiento L2.11.

- Validar si el usuario es asociado a un huella en particular en el dispositivo, debido a que si el dispositivo cuenta con diferentes huellas registradas, un usuario no autorizado y sin conocer las credenciales de acceso de la aplicación, podrá acceder sin ningún problema. Para la prueba se debe registrar inicialmente una huella en el dispositivo la cual será asociada con el usuario de la aplicación. Posteriormente se debe registrar una segunda huella en el dispositivo de una persona diferente, con la que se intentará acceder a la aplicación.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entrela aplicación móvil y el backed.
- Fridump: https://github.com/Nightbringer21/fridump
- Frida: https://www.frida.re/docs/android/
- ADB: Android Debug Bridg (adb), herramienta utilizada para la ejecución de comandos en el sistema operativo Android.
- HPROF Converter: https://stuff.mit.edu/afs/sipb/project/android/docs/tools/help/hprof-conv.html
- Memory Analyzer Tool (MAT) : https://www.eclipse.org/mat/

Lineamiento L4.8

Control definido: Existe un mecanismo de segundo factor de autenticación (2FA) en el servidor y es aplicado consistentemente en transacciones o acciones que manejan información sensible.
Pruebas de Validación: Las aplicaciones utilizan un segundo factor de autenticación al momento de realizar operaciones consideradas sensibles, como puede llegar a ser un cambio de contraseña, realizar una transferencia financiera o al consultar información de salud como es el caso de resultados de laboratorio entre otros. Dentro del esquema de autenticación se define el segundo factor de autenticación como algo que tenemos, por ejemplo un token físico, un token tipo software, mensajes SMS o notificaciones push. Algunas aplicaciones implementan un segundo factor de autenticación bajo una arquitectura que no garantiza su cumplimiento. Las validaciones a realizar son:

- La primera acción a realizar consiste en registrar y analizar la mensajería intercambiada entre la aplicación móvil y el backend por medio de un webproxy, al momento de invocar el flujo del segundo factor de autenticación. El análisis del flujo es crucial para definir que posibles vectores de ataques se pueden utilizar con el propósito de lograr evadir dicho control.
- La validación consiste en identificar si el canal utilizado para entregar al usuario el segundo factor de autenticación, no es considerado seguro. Si la aplicación utiliza el correo electrónico o el envío de mensajes de texto SMS para la entrega del One Time Password (OTP), la probabilidad de perder las propiedades de “algo que tengo” se ven degradadas debido a que el correo electrónico puede ser comprometido fácilmente por un atacante, mediante técnicas de phishing entre otras y el OTP enviado por SMS se puede ver comprometido por medio de ataques de suplantación de simcard conocido como SIM swapping.

- El siguiente punto a validar consiste en engañar a la aplicación mediante la modificación de la respuesta entregada por parte del servidor. Algunas aplicaciones responden a un valor “true” o “false” por parte del backend como respuesta a la validación del OTP ingresado por el usuario. En la validación se debe ingresar un OTP errado, luego se procede a interceptar las comunicaciones y cambiar el valor “false” por “true” con el fin de alterar el comportamiento de la aplicación. Al no contar con los controles adecuados, el backend pierde la trazabilidad del estado actual del acceso, permitiendo continuar con el flujo normalmente. En caso de no manejar valores booleanos en la respuesta por parte del backend, se puede probar inyectando una respuesta satisfactoria de otra sesión a nivel de las cabeceras y cuerpo del mensaje, para intentar engañar a la aplicación y al backend.

- Otro vector utilizado para evadir el segundo factor de autenticación consiste en inyectar el llamado del siguiente paso en el flujo. Al pasar de forma satisfactoria el segundo factor de autenticación, la aplicación debe generar una petición al backend para consumir el servicio protegido. El fin de la prueba es identificar cual es el endpoint (url) protegido e inyectarla en el request mediante la interceptación de la comunicaciones con un webproxy e intentar obtener el acceso. En el ítem anterior se manipuló la respuesta del servidor para engañar a la aplicación, en este ítem la idea es modificar el request para consumir directamente el servicio sin disparar el flujo del segundo factor.

- La siguiente prueba consiste en poder validar el tiempo de vida configurado para el OTP y si es posible reutilizarlo. Para validar el tiempo de vida, se debe realizar la solicitud del OTP desde la aplicación móvil y esperar 40 segundos para ingresarlo. Como buena práctica el OTP debe tener un tiempo de vida no mayor a 30 segundos, debido a que tiempos mayores a dicho valor pueden permitir a un atacante robar el OTP y utilizarlo con fines malicioso. Para validar el re uso del otp, la idea es utilizar un grupo de OTPs utilizados previamente y validar directamente en la aplicación si es acepatdo o no por el backend.

- La última prueba consiste en validar la posibilidad de adivinar el OTP por medio de un ataque de fuerza bruta, debido a la baja entropía del OTP. La viabilidad del ataque depende de los siguientes factores que deben ser identificados en la aplicación, utilizando un webproxy:
  - Validar si el tamaño del OTP es menor o igual a 5 dígitos. Entre menos tamaño tenga el OTP, la probabilidad de adivinar su valor crece.
  - Validar si el backend no controla un máximo de intentos de ingreso del OTP. Para esta prueba se puede realizar una serie de intentos manuales con el fin de validar si se presenta algún tipo de bloqueo por parte del servidor.
  - Si el backend no controla un máximo de intentos al momento de ingresar el OTP, el siguiente paso consiste en automatizar la peticiones mediante el reenvío de

múltiples request (Replay Attack) con diferente valor, hasta lograr obtener el indicado. La actividad puede ser realizada desde un webproxy o payloads creados mediante lenguajes de programación como python. La lista o diccionario de posibles OTPs a validar, se pueden crear con funcionalidades expuestas en internet que permiten crear números aleatorios especificando tamaño y total de registros.

Herramientas de apoyo:

- pinetools.com es una página en internet que dispone de un generador de cadenas, permitiendo crearlas de forma aleatoria, configurando la cantidad y longitud de la cadena. La página en internet es la siguiente https://pinetools.com/es/generador-cadenas-aleatorias.
- Webproxy Burp Suite, utilizando la opción en la pestaña Intruder. En el siguiente enlace se puede observar un ejemplo de su uso. https://support.portswigger.net/customer/portal/ articles/1964020-using-burp-to-brute-force-a-login-page.
- Webproxy Owasp Zap, utilizando la función Fuzzing. En el siguiente video se puede observar un ejemplo de su uso. https://www.youtube.com/watch?v=tBXX_GAK7BU
- Lenguaje de programación en python, permite crear programar este tipo de ataques dirigidos a protocolos http y https. El programar el payload permite tener mayor flexibilidad para obtener un mejor resultado. Un ejemplo es el siguiente: https://github.com/Sanix-Darker/Brute-Force-Login.

Lineamiento L4.9

Control definido: Controlar riesgos relacionados a la recuperación de credenciales de acceso, atravez de funcionalidades de recordar credenciales.
Pruebas de Validación: Algunas aplicaciones cuentan con funcionalidades de autogestión que permiten recuperar el usuario o la contraseña en caso de olvido. En ocasiones a nivel de diseño, las implementaciones cuentan con debilidades de seguridad que pueden ser aprovechadas fácilmente por un atacante, con el fin de lograr suplantar un usuario. Las validaciones a realizar son las siguientes:

- La primera actividad consiste en identificar la existencia de funcionalidades relacionadas a recuperar usuario o contraseña en la aplicación. Se debe seguir el flujo de recuperación y capturar la mensajería generada. El fin de esta actividad es construir un diagrama de flujos que permita de una forma mas sencilla lograr identificar posibles vectores de ataque.
- Uno de los mecanismo utilizados comúnmente para recuperar datos como usuario o contraseñas, está relacionada al correo electrónico o número de celular previamente

registrado. Un ejemplo es el envío de un link que tiene como fin continuar algún proceso adicional de recuperación del dato o sencillamente envía una contraseña temporal. En estos casos se debe validar:

- Validar si la contraseña es enviada directamente al correo electrónico. Teniendo en cuenta que este medio de envío no es seguro, el usuario está expuesto a una suplantación de identidad debido a que el atacante por medio de diferentes técnicas puede comprometer el correo electrónico.
- Validar si al enviar la solicitud de recuperación de usuario o contraseña, el dispositivo incluye en la mensajería de envío (Request), el correo electrónico o el número de celular previamente registrado por el usuario. De ser así la prueba consiste en reemplazar el correo o número de celular del usuario por el del atacante.
- Si la recuperación consiste en el envío de un link al correo electrónico para continuar un flujo específico, se debe validar que dicho link tenga un tiempo de vida corto y no puede ser reutilizado, como ocurre con los OTP.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entrela aplicación móvil y el backed.

J.0.4. Manejo de Sesiones y autorización

Lineamiento L5.1

Control definido: Implementar procedimientos para verificar que una persona o entidad que busca acceso a la información de salud electrónica protegida, se encuentra debidamente autorizado. (Referencia insegura directa a objetos).
Pruebas de Validación: Normalmente al momento del diseño de arquitectura y desarrollo de las aplicaciones aplicaciones móviles, se utilizan referencias a objetos que pueden ser un fichero, base de datos o directorios, para dar acceso a un recurso en específico. Por ejemplo la variable ïd.[en] una petición (Request) al backend, pueden contener un valor (número de cédula) que hace referencia a un registro en un directorio de forma directa, permitiendo por ejemplo la descargar un archivo con resultados de exámenes de laboratorio. Al no implementar adecuadamente un control de autorización, un atacante con una sesión establecida correctamente, podrá descargar los exámenes de otros usuarios con solo modificar el valor de la variable ïd". Otro vector consiste en la posibilidad de consumir servicios de forma directa, sin contar con los debidos privilegios o autorización para hacerlo. Las validaciones consisten en:

- Se debe recorrer todas las funcionalidades de la aplicación móvil, capturando la mensajería por medio de un webproxy, identificando aquellas funciones o recursos que pueden contener información personal o de salud del paciente.

- Se debe analizar la mensajería en búsqueda de variables que pueden viajar en la url (GET) o en el cuerpo del mensaje (POST), que se comporte como un identificador, por ejemplo la variable ïd". El nombre de la variable es diferente dependiendo de la aplicación, por esta razón se debe analizar cuidadosamente los request que solicitan la información privada y entender su flujo. En las pruebas de reconocimiento funcional de las aplicaciones del sector salud en Colombia, se observó que las aplicaciones suelen utilizar el número de cédula para referencia de forma directa un objeto.

- Identificada las variables susceptibles a Referencia Insegura Directa a Objetos (IDOR), el siguiente paso consiste en modificar la variable por valores que corresponden a otros usuarios. Para la prueba es necesario contar con 2 usuarios válidos, se ingresa con el usuario A y se modifica el valor con los datos del usuario B, o realizar un ataque de diccionario o fuerza bruta con el fin de adivinar referencias directas a objetos válidas en la aplicación. El ataque se puede realizar por medio de funcionalidades de fuzzer que traen algunos webproxys.

- Para validar el otro vector de ataque se debe contar con 2 perfiles de usuario, uno con todos los privilegios sobre la aplicación y otro usuario con acceso limitado a los servicios. Las actividades a realizar son:
  - Con el usuario de mayor privilegio se procede a realizar la captura de la mensajería intercambiada entre la aplicación y el backend, de cada uno de los servicios que no pueden ser consumidos por usuarios con bajo privilegio. Lo anterior es con el fin de analizar el flujo generado e identificar la forma en que son consumidos dichos servicios.
  - Con el usuario de bajo privilegio, se debe intentar de consumir los servicios no autorizados mediante la inyección directa de la url del servicio en el request o inyectando el menú que provee los servicios en el response body. En este punto todo depende del análisis realizado previamente de los flujos.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entrela aplicación móvil y el backed.

- Webproxy Burp Suite, utilizando la opción en la pestaña Intruder para el ataque de diccionario o fuerza bruta. En el siguiente enlace se puede observar un ejemplo de su uso. https://support.portswigger.net/customer/portal/articles/1964020-using-burp-to-brute-force-a-login-page.

- Webproxy Owasp Zap, utilizando la función Fuzzing para el ataque de diccionario o fuerza bruta. En el siguiente video se puede observar un ejemplo de su uso. https://www.youtube.com/watch?v=tBXX_GAK7BU

- Lenguaje de programación en python, permite crear programar este tipo de ataques dirigidos a protocolos http y https. El programar el payload permite tener mayor flexibilidad para obtener un mejor resultado. Un ejemplo es el siguiente: https://github.com/Sanix-Darker/Brute-Force-Login.

Lineamiento L5.2

Control definido: Informar al cliente, al inicio de cada sesión, la fecha y hora del último ingreso a este canal y la dirección IP .
Pruebas de Validación: Informar al usuario final la fecha y hora del último acceso a la aplicación y desde que dirección IP se originó, ayuda de forma preventiva a identificar posibles accesos no autorizados e informar oportunamente a la entidad del servicio y realizar cambio de contraseña para evitar o mitigar una posible exposición de información por suplantación de identidad. La validación no solo consiste en verificar la existencia de la información al iniciar una sesión, también se debe validar su correcta implementación para evitar riesgos de seguridad. Principalmente se debe:

- Validar que la dirección expuesta corresponde a la IP pública del usuario. En ocasiones por error humano, la dirección IP expuesta resulta ser la IP interna del servidor.
- La configuración horaria utilizada para mostrar la información de fecha y hora al usuario, debe corresponder a la ubicación geográfica del usuario.
- Se debe validar si estos valores son capturados desde el móvil y enviados al servidor. Esta mala práctica permitiría a un atacante alterar los valores mediante la interceptación y modificación de la mensajería con un webproxy, con el fin de generar confusión al usuario o dado el caso a un analista forense que se encuentre adelantando una investigación. Los datos deben ser capturados por el backend.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entrela aplicación móvil y el backed.

Lineamiento L5.3

Control definido: Una sesión solo puede ser entregada al cliente por parte del servidor previa autenticación válida. Si la aplicación maneja sesiones antes de autenticación, debe existir un mecanismo que autentique una nueva sesión antes de consumir los servicios.
Pruebas de Validación: Dependiendo de la arquitectura utilizada para las aplicaciones móviles, la sesión es entregada una vez supera correctamente la autenticación. En algunos casos el backend entrega una sesión al momento de abrir la aplicación o después de ingresar el usuario cuando la autenticación de usuario y contraseña se realiza en 2 pasos, es decir en

una primera pantalla valida el usuario y en una segunda pantalla valida la contraseña. Este tipo de diseño puede generar brechas de seguridad si no se cuenta con los debidos controles. Para validar se debe:

- Para el desarrollo del análisis de la mensajería, se debe realizar una interceptación de las comunicaciones entre el flujo generado en la apertura de la aplicación y el flujo generado por una autenticación satisfactoria.
- Se debe verificar en que momento el backend entrega la sesión a la aplicación, se puede dar al abrir la aplicación, al ingresar solo el usuario o al pasar satisfactoriamente la autenticación.
- El siguiente paso consiste en validar 2 vectores principalmente:
  - El primer vector está relacionado a la fijación de la sesión. En este punto se valida si la sesión entregada antes de autenticarse satisfactoriamente es la misma sesión después de autenticación. De presentarse este escenario, un atacante podrá intentar por medio de técnicas de ingeniería social, inyectar a la víctima una sesión generada por el atacante antes de autenticarse. De esta forma, una vez la víctima realiza de forma satisfactoria la autenticación,el atacante contará con una sesión válida para consumir los servicios de la aplicación sin conocer el usuario y la contraseña de la víctima. Una referencia de este tipo de ataque se puede obtener del siguiente link: https://www.owasp.org/index.php/Session_fixation.
  - El segundo vector está relacionada a la posibilidad de utilizar una sesión "sin autenticar", que muy seguramente no permitirá consumir los principales recursos que provee la aplicación, pero puede llegar a permitir el consumo de otros servicios como por ejemplo la carga de archivos, el cambio de contraseña, generador de tokens entre otros que ponen en riesgo la confidencialidad de la información. Para el correcto desarrollo de la prueba se debe listar cada uno de los servicios consumidos por la aplicación, contemplando todas las funcionalidades disponibles incluyendo los servicios de terceros que son llamados desde la aplicación y que forman parte del core del negocio.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entrela aplicación móvil y el backed.

Lineamiento L5.4

Control definido: Si se utiliza la gestión de sesión por estado, el servidor remoto usa tokens de acceso aleatorios para autenticar los pedidos del cliente sin requerir el envío de las credenciales del usuario en cada uno.

Pruebas de Validación: Comúnmente las aplicaciones utilizan la autenticación con estado o conocida en el idioma ingles como "stateful authentication", tiene como principio generar un valor de sesión aleatorio a un usuario que ha superado satisfactoriamente la autenticación. El id de sesión generando identificará al usuario en las siguientes peticiones que se realicen al backend y no contiene ningún tipo de información. Los datos de sesión del usuario (tipo de usuario, rol, acceso permitido entre otros) es almacenada en el backed. Las validaciones a realizar son:

- La prueba consiste en realizar el proceso de autenticación en reiteradas ocasiones utilizando un webproxy, con el fin de capturar varios id de sesión y lograr identificar como es construido por parte del backend, es decir definir si solo son números o son valroes alfanuméricos, el tamaño utilizado y que tan predecible puede llegar a ser. OWASP recomienda un tamaño no menor a 128 bits y con suficiente aleatoriedad (entropía) para evitar los ataques de fuerza bruta. https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md. Si la sesión no cuenta con dichas recomendaciones se puede utilizar las funcionalidades de ***análisis de tokens de sesión*** que traeOWASP ZAP o Butpsuite para intentar adivinar una sesión válida. Otra forma de adivinar la sesión es mediante la construcción de diccionarios, que contemplen los comportamiento de la sesión analizados previamente y lanzar ataques de fuzzing expuestos en el lineamiento 4.8. Autenticación.

- La siguiente prueba consiste en validar la posibilidad de realizar un secuestro de sesión o conocido en el idioma ingles como Session hijacking, al lograr robar"la sesión de la víctima e inyectarla en las cookies de la mensajería utilizada por el atacante por medio de la interceptación de las comunicaciones con un webproxy, con el fin de suplantar a la víctima y consumir correctamente los servicios sin conocer el usuario y la contraseña. La cookie de la víctima se puede obtener de diferentes formas:
  - Predicción de la sesión. este vector fue tratado en el ítem anterior.
  - Session Sniffing. La sesión puede ser capturada al momento de realizar un sniffing de la red debido a que la sesión viaja por métodos http inseguros (GET), es decir en la url, o puede ser capturada por medio de ataques de hombre en el medio. https://www.owasp.org/index.php/Session_hijacking_attack
  - El almacenamiento inseguro de la sesión es otro vector que puede aprovechar un malware instalado en el dispositivo para extraer la sesión activa del usuario. Esta información puede ser almacenada en base de datos SQLite, en archivos de configuración, en los logs del dispositivo móvil o cualquier otra ubicación que pueda ser accedida por un tercero. Ver lineamiento L2 - Almacenamiento de datos y la Privacidad.
  - Dependiendo de la arquitectura de la aplicación, la cookie también puede ser obtenida por medio ataques de inyección de código como es el caso del ataque

Cross Site Scripting XSS, el cual puede enviar la cookie a un servidor en internet. Este tipo de vulnerabilidades aumentan el riesgo de robo de cookies de sesión.

- Validar si la sesión es creada de lado del servidor y no mediante una función en la aplicación móvil. Para realizar la validación se debe analizar la mensajería capturada al momento de realizar la autenticación. Cuando la autenticación es correcta, el backend es quien debe entregar la sesión a la aplicación móvil. La validación se puede realizar utilizando un webproxy.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entrela aplicación móvil y el backed.
- Webproxy Burp Suite, en el siguiente enlace se puede observar un ejemplo de su uso. https://support.portswigger.net/customer/portal/articles/1964169-using-burp-to-test-session-token-generation.
- Webproxy Owasp Zap. En el siguiente link se puede observar un ejemplo de su uso. https://github.com/zaproxy/zap-extensions/wiki/AddOn_tokengen
- Lenguaje de programación en python, permite crear programar este tipo de ataques dirigidos a protocolos http y https. El programar el payload permite tener mayor flexibilidad para obtener un mejor resultado. Un ejemplo es el siguiente: https://github.com/Sanix-Darker/Brute-Force-Login.

Lineamiento L5.5

Control definido: Si se utiliza la autenticación basada en tokens sin estado, el servidor proporciona un token que se ha firmado utilizando un algoritmo seguro, no expone información sensible e implementa controles para evitar la suplantación.
Pruebas de Validación: La autenticación basada en tokens corresponde al envío de un token debidamente firmado o cifrado, que es validado de lado del backend, con el fin de definir que servicios puede consumir y con que tipo de privilegios cuenta el solicitante. El token es enviado en cada petición realizada al backend y no es almacenado en el servido. El estandar RFC 7519 hace referencia a Json Web Token (JWT), que es unos de los formatos comunmente utilizado para controlar la autorización de acceso a los servicios. https://tools.ietf.org/html/rfc7519 Las validaciones a realizar son las siguientes:

- La estructura de un Json Web Token está conformada por un Header un Payload y el Signature, separados por un punto (.) y codificada en Base 64. Bajo esta estructura las pruebas a realizar son las siguientes teniendo en cuenta la guía de owasp https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_Cheat_Sheet_ for _Java.html:

- Inicialmente se debe interceptar las comunicaciones con un webproxy y capturar la mensajería generada al momento de realizar una correcta autenticación en la aplicación y consumir los respectivos servicios. Lo anterior es con el fin de identificar el JWT en la mensajería. Este valor se pude encontrar en los header del request como ***Authorization: Bearer <jwt_token>*** o se puede encontrar en el body del request empezando normalmente de la siguiente forma ***eyJhbGciOi.....*** que equivale a ***{"alg":"***.
- Identificado el token JWT, se debe realizar la decodificación de la información utilizando un decodificador de base64 o utilizando el servicio habilitado en la página web https://jwt.io/.
- Con la información decodificada se debe validar si expone información que puede llegar a ser útil a un atacante con el fin de poder identificar nuevos vectores de ataque. Por ejemplo, puede contener información que indique los roles de seguridad, usuario, software utilizado para el login (LDAP) entre otros.
- El siguiente vector de ataque consiste en modificar el algoritmo definido inicialmente por la opción "none". Si la implementación del JWT permite la definición de este tipo de algoritmos, el backend no validará la firma y permitirá a un atacante modificar los valores en el payload o en el mismo header con el fin de obtener autorizaciones o permisos que no le corresponden, como por ejemplo modificar el rol de üser.ª rol de .ªdmin". Una vez se modifique los valores, se debe codificar nuevamente en base64 e inyectar el nuevo token solo incluyendo el header y el payload (header.payload), el valor de signature debe ser eliminado. Otra forma de atacar el algoritmo es mediante la degradación del mismo. Un ejemplo es cambiar el algoritmo de cifrado asimétrico (RS256) a cifrado simétrico (HS256). como resultado, la aplicación utiliza la llave pública de RSA256 como llave secreta y el algoritmo HS256 para verificar la firma.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backed.
- El decodificador de base64 se pude encontrar dentro de las mismas funcionalidades de los webproxy o utilizando la herramienta online de Json Web Token https://jwt.io/.

Lineamiento L5.6

Control definido: Cuando el usuario cierra la sesión en la aplicación, la sesión debe ser caducada en el servidor.
Pruebas de Validación: Al autenticarse correctamente un usuario en la aplicación, el backend entrega una sesión que permite identificarlo en las próximas peticiones que se realicen

sin tener que enviar nuevamente las credenciales de acceso. Por esta razón, cuando el usuario decide cerrar la sesión desde la aplicación, el backend debe tener la capacidad de caducar la sesión con el fin de evitar que la sesión sea reutilizada por un atacante que logró de alguna manera capturar la sesión de la víctima. La forma de validar el cumplimiento del lineamiento es el siguiente:

- Inicialmente se debe interceptar las comunicaciones con un webproxy y capturar la mensajería generada después de realizar una correcta autenticación en la aplicación y consumir los respectivos servicios.
- Un forma de generar un cierre de sesión es utilizando la función salir o cerrar sesión dispuesta por la aplicación. Al realizar esta acción, la aplicación normalmente debe ubicar al usuario en la pantalla de login. En este punto se procede a realizar un reenvío de la mensajería capturada por el webproxy de algunos de los servicios consumidos antes del cierre de sesión, con el fin de validar si el servidor está rechazando las peticiones o permitiendo el consumo de los servicios.
- Otra forma de generar el cierre de sesión por parte del usuario, consiste en cerrar la ventana relacionada a la aplicación. Normalmente los dispositivos móviles tiene la opción de ver todas las aplicaciones abiertas y cerrarlas dando click en la "X.º simplemente desplazando la aplicación hacia arriba.En este punto se procede a realizar un reenvío de la mensajería capturada por el webproxy de algunos de los servicios consumidos antes del cierre de sesión, con el fin de validar si el servidor está rechazando las peticiones o permitiendo el consumo de los servicios.
- En los dos casos se debe validar en la mensajería capturada, si al momento de realizar alguno de los dos cierres expuestos anteriormente, generó una mensajería de cierre de sesión o "logout". Este analices se debe realizar teniendo en cuenta que posiblemente la aplicación si está enviando el cierre de sesión por el backend no está respondiendo de forma adecuada.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.

Lineamiento L5.7

Control definido: Las sesiones y los tokens de acceso expiran luego de un tiempo predefinido de inactividad.
Pruebas de Validación: Existen diversos motivos por el cual un usuario no realiza de forma voluntaria un cierre de sesión. Por ejemplo una de las causas corresponde al olvido que puede tener el usuario de cerrar la sesión, otro motivo puede ser un cierre no esperado por parte de

la aplicación debido a un defecto propio de la aplicación o por un cierre forzado por parte del mismo sistema operativo. En todos estos casos la sesión quedará activa de lado del backend, abriendo la posibilidad de que un ciber-delincuente realice una suplantación de identidad, reutilizando una sesión válida de la víctima. Las validaciones a realizar son las siguientes:

- Inicialmente se debe interceptar las comunicaciones con un webproxy y capturar la mensajería generada después de realizar una correcta autenticación en la aplicación y al momento en que se genera el cierre de sesión por inactividad, dado que esté configurado dicho control.
- Se debe disponer de un cronómetro físico o preferiblemente un cronometro online como el de chrome que facilitaría la toma de evidencias para el informe. El cronómetro nos dará una idea del tiempo de inactividad que tiene configurada la aplicación antes de realizar el cierre de sesión o si no cuenta con dicho control.
- Según recomendación de OWASP, se recomienda un tiempo de inactividad entre 10 minutos y 2 horas dependiendo de la sensibilidad de las operaciones que realiza la aplicación o la información que gestiona. En algunas aplicaciones financieras se utiliza entre 7 a 10 minutos antes de generar el cierre de sesión por timeout https://github.com/OWASP /owasp-mstg/blob/master/Document/0x04e-Testing-Authentication-and-Session
  -Management.md

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.
- Cronómetro online de Google: http://cronometro-online.chronme.com/

Lineamiento L5.8

Control definido: Controlar las sesiones simultáneas sin importar los diferentes canales que provean el servicio.
Pruebas de Validación: Normalmente los servicios que expone una organización cuenta con un canal móvil y un canal web para su acceso. Dependiendo de la arquitectura utilizada, los servicios pueden ser los mismos o independientes. A nivel funcional un usuario no entra de forma simultanea al mismo canal o a los dos canales, lo cual si representa un riesgo de seguridad, debido a que un atacante que ha logrado capturar la sesión de la víctima o las credenciales de autenticación por medio de diferentes vectores de ataque, podrá acceder de forma simultánea a la aplicación y consumir los servicios sin que la víctima se de por enterado. Las validaciones a realizar son las siguientes:

- Se debe contar con dos dispositivos móviles y dos operadores de internet diferentes para realizar las validaciones.
- Ingresar con la misma sesión desde dos dispositivo móviles diferentes bajo una misma IP pública. En este punto se requiere de un webproxy para interceptar las comunicaciones y poder inyectar la sesión. Se debe navegar en la aplicación para validar los accesos.
- Ingresar con la misma sesión desde dos dispositivo móviles diferentes desde IP públicas diferentes. En este punto se requiere de un webproxy para interceptar las comunicaciones y pode inyectar la sesión. Se debe navegar en la aplicación para validar los accesos.
- Realizar el flujo normal de autenticación desde dos dispositivo móviles diferentes bajo una misma IP pública. Se debe navegar en la aplicación para validar los accesos desde los dos dispositivos.
- Realizar el flujo de autenticación desde dos dispositivo móviles diferentes desde IP públicas diferentes. Se debe navegar en la aplicación para validar los accesos desde los dos dispositivos.
- Ingresar con la misma sesión desde un dispositivo móvil y el canal web bajo una misma IP pública. En este punto se requiere de un webproxy para interceptar las comunicaciones y pode inyectar la sesión. Se debe navegar en la aplicación tanto en el móvil como el navegador web para validar los accesos.
- Ingresar con la misma sesión desde un dispositivo móvil y el canal web desde IP públicas diferentes. En este punto se requiere de un webproxy para interceptar las comunicaciones y pode inyectar la sesión. Se debe navegar en la aplicación tanto en el móvil como el navegador web para validar los accesos.
- Realizar el flujo de autenticación desde un dispositivo móvil y el canal web bajo una misma IP pública. Se debe navegar en la aplicación tanto en el móvil como el navegador web para validar los accesos.
- Realizar el flujo de autenticación desde un dispositivo móvil y el canal web desde IP públicas diferentes. Se debe navegar en la aplicación tanto en el móvil como el navegador web para validar los accesos.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.
- En el navegador web se puede utilizar el plugin Cookie-editor para inyectar o extraer la sesión. https://addons.mozilla.org/en-US/firefox/addon/cookie-editor/

Lineamiento L5.9

Control definido: La aplicación informa al usuario acerca de los accesos a su cuenta desde dispositivos no registrados. El usuario es capaz de ver una lista de los dispositivos conectados y bloquear el acceso desde ciertos dispositivos.
Pruebas de Validación: El fin del lineamiento es brindar al usuario la posibilidad de reaccionar ante una posible suplantación de identidad. Para lograr el objetivo la aplicación debe estar en la capacidad de realizar un proceso de registro e identificación único del dispositivo. Contar con alertar que le informe al usuario por push o vía email sobre accesos realizados a la aplicación en tiempo real. Con este proceso el usuario podrá tener un mayor control sobre cuales dispositivos están permitidos autenticar e identificar si alguien no autorizado ingresó a la aplicación.

- La validación inicial consiste en identificar si la aplicación cuenta con esta funcionalidad y cumple con las capacidades expuestas en el lineamiento.

- La primera acción a realizar consiste en capturar y analizar la mensajería intercambiada entre la aplicación móvil y el backend por medio de un webproxy, al momento de invocar el flujo de autenticación. El análisis del flujo es crucial para definir que posibles vectores de ataques se pueden utilizar con el propósito de lograr evadir dicho control.

- El siguiente punto a validar consiste en intentar engañar tanto a la aplicación como al servidor relacionado al flujo que debe continuar, mediante la manipulación o modificación de la mensajería. Por ejemplo algunas aplicaciones responden a un valor “true” o “false” por parte del backend como respuesta a la validación realizada al dispositivo móvil. La prueba consiste en interceptar las comunicaciones y cambiar el valor “false” por “true” con el fin de alterar el comportamiento de la aplicación. Al no contar con los controles adecuados, el backend pierde la trazabilidad del estado actual del acceso, permitiendo continuar con el flujo normalmente. En caso de no manejar valores booleanos en la respuesta por parte del backend, se puede probar inyectando una respuesta satisfactoria de otra sesión a nivel de las cabeceras y cuerpo del mensaje, para intentar engañar a la aplicación y al backend.

- Otro vector utilizado para evadir el control consiste en inyectar el llamado del siguiente paso en el flujo. Al pasar de forma satisfactoria las validaciones de huella del dispositivo, la aplicación debe generar una petición al backend para consumir el servicio una vez se supera la autenticación. El fin de la prueba es identificar cual es el endpoint (url) protegido e inyectarla en el request mediante la interceptación de la comunicaciones con un webproxy e intentar obtener el acceso. En el ítem anterior se manipuló la respuesta del servidor para engañar a la aplicación, en este ítem la idea es modificar el request para consumir directamente el servicio sin disparar el flujo de validación de la huella del dispositivo.

- El siguiente vector consiste en suplantar el dispositivo móvil, alternado la mensajería mediante la inyección de la huella que lo identifica o los datos del dispositivo móvil que genera dicha huella, en otro dispositivo móvil. De esta forma se lograría la suplantación si no existe una persistencia en la validación o controles adicionales que eviten la manipulación. La información del dispositivo puede ser obtenida fácilmente si la víctima es infectada con malware o por medio de técnicas de ingeniería social.

- Se debe identificar los posibles riesgos de seguridad a nivel de integración de la solución con la aplicación, es decir, el riesgo no solo está en el flujo de autenticación donde se valida la huella del dispositivo móvil, también se debe validar el flujo de registro de la huella del dispositivo y demás funcionalidades que la conforman.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.

J.0.5. Comunicación a través de la red

Lineamiento L6.1

Control definido: Implementar medidas de seguridad técnicas y canales seguros para protegerse contra el acceso no autorizado a información médica protegida electrónica que se transmite a través de una red de comunicaciones electrónicas.
Pruebas de Validación: La información además de ser protegida a nivel del dato (L3-Criptografía) debe ser enviada por medio de un canal seguro, utilizando protocolos y cifrado definidos en el Lineamiento L6.2 de forma consistente en la aplicación. El fin del lineamiento es asegurar que la aplicación en todo momento está utilizando HTTPS en las comunicaciones. El no cumplir con este linemaiento, el backend permitirá la degradación del protocolo de comunicación segura. Las validaciones a realizar son.

- Se debe recorrer todas las funcionalidades de la aplicación móvil, capturando la mensajería por medio de un webproxy e identificar si en algún momento la aplicación realiza llamados utilizando el protocolo no seguro HTTP. Los flujos a validar deben estar protegidos, es decir después de autenticación debido a que algunas aplicaciones cuentan con espacios de acceso público ubicados normalmente en la pantalla de login.

- con los flujos ya identificados, se debe interceptar las comunicaciones con un webproxy e intentar forzar el uso de HTTP. En ocasiones el backed permite el consumo de los servicios por HTTP y HTTPS permitiendo a un atacante por medio de un ataque de hombre en el medio, degradar las comunicaciones a la víctima y observar la mensajería intercambiada en texto claro. En otros casos, el backend tiene habilitado el protocolo HTTP pero realiza redireccionamientos al protocolo HTTPS, lo que también permitiría en algún momento por error humano, llegar a consumir el servicio por HTTP.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend.

Lineamiento L6.2

Control definido: Las configuraciones del protocolo TLS siguen las mejores prácticas o tan cerca posible mientras que el sistema operativo del dispositivo lo permite y configura Ciphersuite que no estén considerados como débiles o vulnerables.
Pruebas de Validación:TLS y SSL son protocolos diseñados para brindar seguridad en la transferencia de información entre el cliente y el servidor. Los cipher suites son utilizados por los protocolos TLS y SSL para asegurar la transferencia de la información. Están conformado por una combinación de algoritmos de autenticación, cifrado y el código de autenticación de mensajes (MAC). Un ejemplo de un cipher suite es el siguiente: DHE-RSA-AES256-GCM-SHA384. Si alguno de estos algoritmos cuenta con debilidades de seguridad conocidas, un atacante podrá intentar degradar o romper el cifrado de la comunicación con el fin de acceder a la información. Las pruebas a realizar son: https://www.acunetix.com/blog/articles/tls-ssl-cipher-hardening/

- El objetivo consiste en identificar que versión de protocolos TLS/SSL y que cipher suites tiene habilitado el servidor. Para realizar la validación se puede utilizar herramientas que permiten fácilmente identificar el grado de fortalece con el cuenta el servidor para proteger las comunicaciones. Las herramientas de validación pueden ser dependiente de 2 tipos de escenarios:
    - Servidores expuestos a internet: Diferentes herramientas online se encuentran disponibles para validar los portocolos y los cipher suites configurados en el servidor, presentado información detallada sobre los hallazgos encontrados. En el informe se debe verificar principalmente si el servidor está permitiendo protocolos SSL, TLS 1.0 o TLS 1.1 que ya son considerados no seguros y validar si cuenta con cipher suite débiles o inseguros. Normalmente este tipo de portal presenta una calificación general dependiendo de la fortaleza encontrada.
    - Servidores en ambientes no expuestos a internet: Este escenario normalmente se presenta para pruebas del tipo Ethical Hacking, debido a que corre en servidores que no son publicados a internet y por lo tanto no pueden ser alcanzados por herramientas públicas. En este caso se pueden utilizar herramientas que cuentan con la información necesaria para realizar un escaneo al servidor y entregar reportes de los resultados. Las verificaciones tienen el mismo alcance descrito en el ítem anterior.

Herramientas de apoyo:

- Herramienta para validación online:
  https://www.ssllabs.com/ssltest/
- Herramienta para validaciones en servidores no expuestos a internet:
  https://github.com/drwetter/testssl.sh
- Herramienta que permite consultar el grado de fortaleza de un ciphersuite:
  https://ciphersuite.info/

Lineamiento L6.3

Control definido: 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.
Pruebas de Validación: El ataque conocido como Hombre en el Medio (MiTM) no solo tiene como fin intentar interceptar el tráfico de la víctima para alterar la información en tránsito u obtener datos confidenciales, también es utilizado para analizar el comportamiento de la aplicación y encontrar nuevos vectores de ataque que permitan comprometer la aplicación o el backend. Por lo anterior el lineamiento tiene como objetivo evitar que un atacante intercepte las comunicaciones. Las siguientes validaciones permiten identificar el grado de fortaleza de la aplicación para evitar la interceptación de las comunicaciones: https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05g-Testing-Network-Communication.md.

- La primera prueba consiste en utilizar certificados auto firmados para establecer la comunicación entre la aplicación móvil y el backend por medio de un webproxy. Si bajo esta condición se logra interceptar las comunicaciones en HTTPS, se puede definir que la aplicación no cuenta con el control para evitar el uso de certificados auto firmados.
- La siguiente prueba consiste en validar si la aplicación y el backend acepta el uso de certificados inválidos. En el webproxy se debe configurar la opción de generar un certificado auto firmado especificando el hostname, es decir el Nombre Común (CN). Esta prueba se realiza en caso de que el backend valide el CN del certificado. Si se logra interceptar las comunicaciones https, se puede definir que la aplicación acepta todos los certificados.
- Con el fin de evitar el control de las validaciones de certificados de confianza por parte de la aplicación, se procede a instalar el certificado del webproxy en el módulo de certificados de usuario del dispositivo móvil. Si se logra interceptar las comunicaciones https, se puede definir que la aplicación acepta certificados instalados por el usuario y que el backend no está validando el certificado intercambiado para las comunicaciones.

- Con el fin de evitar el control que valida el no uso de certificados de confianza instalados por el usuario en el dispositivo móvil, se procede a instalar el certificado del webproxy en el sistema. Para lograr este obejtivo se debe contar con un dispositivo rooted o con jailbreak y que tenga instalado el Framework Magisk y el módulo "Move Vertificateçon el fin de pasar todos los certificados de üser.ª "system". Si se logra interceptar las comunicaciones https, se puede definir que el backend no está validando el certificado intercambiado para las comunicaciones.

- Si el backend realiza validaciones del certificado utilizado para establecer las comunicaciones con la aplicación, es decir cuenta con Çertificate pinning"se debe realizar las siguientes validaciones con el fin de realizar un salto o bypass del control.Si se logra interceptar las comunicaciones https con algunos de los vectores descritos a continuación, se puede definir que la implementación no es la correcta o carece de otros controles que ayuden a fortalecer la implementación inicial.

  - Una forma de validar si la aplicación está utilizando certificate pinning, consiste en realizar un escaneo de la aplicación con la herramienta Mobile Security Framework (Mobsf), disponible tanto para Andorid como para iOS. La información se encuentra al dar click en la opción "File Analysis.en la pestaña "Security Analysis".

  - Una forma de evadir el control consiste en instalar el certificado del webproxy en el directorio que contiene los certificados de la aplicaciónr. Para realizar este proceso se debe:

    - Se de realizar un proceso de decompilación de la aplicación con APKTOOL.
    - En la carpeta decompilada, se debe buscar el directorio que contiene los certificados y pegar el certificado del webproxy.
    - Se debe realizar el proceso de compilación de la aplicación con APKTOOL.
    - Se debe realizar el proceso de firmado de la aplicación y luego instalar la aplicación en el dispositivo móvil.

  - Otra forma de poder realizar un bypass al control de certificate pinning, consiste en utilizar las funcionalidades "SSL Unpinning.º "TrustMeAlready"por medio del framework XPOSED, el cual permite agregar funciones individuales a la ROM del dispositivo móvil, que debe estar rooted o jailbreak para funcionar correctamente.

  - Una de las herramientas que toma fuerza hoy en día es Frida, una herramienta tipo cliente - servidor que funciona en dispositivos móviles sin requerir en algunos casos acceso como súper usuario o root. Permite inyecciones de scripts y realizar "Hook.ª las funciones cuando la aplicación se está ejecutando. Con las diversas funcionalidades, Frida permite realizar bypass a controles de certificate pinning como el ejemplo expuesto en el siguiente tutorial para android.

https://medium.com/@ved_wayal/hail-frida-the-universal-ssl-pinning-bypass-for-android-e9e1d733d29, un ejemplo para iOS es el siguiente: https://medium.com/@macho_reverser/bypassing-certificate-pinning-on-ios-12-with-frida-809acdb875e7

- Otra opción consiste en alterar el binario modificando el código (smali). Esta técnica es utilizada cuando las herramientas anteriormente expuestas no logran el objetivo de evadir el control que provee el certificate pinning. Para el desarrollo de esta técnica se debe realizar los siguientes pasos: https://serializethoughts.com/2016/08/18/bypassing-ssl-pinning-in-android-applications/
  - Se debe decompilar el instalador utilizando herramienta especializadas como APKTOOL, el cual genera un carpeta con el código smali.
  - Identificar funciones relacionadas a implementaciones de certificate pinning, como son por ejemplo X509TrustManager, checkServerTrusted, SSLContext, X509Certificate, TrustManager, TrustManagerFactory, CertificateFactory, CertPathValidator, PKIXParameters, CertPath, KeyStore entre otras. Se recomienda utilizar la herramienta ByteCode Viewer que permite visualizar el código java y smali al mismo tiempo en ventanas paralelas.
  - Identificadas las funciones, se debe entender la lógica de la implementación con el fin de cambiar su comportamiento mediante la modificación del código smali y lograr evadir el control. En ocasiones solo con modificar el condicional que realiza la validación es suficiente para lograr el objetivo. Un ejemplo es cambiando el condicional ïf-eqz.ª ïf-nez"según el caso.
  - Con las modificaciones ya realizadas en el o los archivos smali, se procede a recompilar la aplicación con la herramienta APKTOOL y posteriormente se debe firmar la aplicación resultante con herramientas del tipo signapk antes de instalarla en el dispositivo móvil.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend y modificar el tipo de certificado a utilizar.
- Magisk cuenta con módulos que ayudan a evadir algunos controles de seguridad. Link: https://github.com/topjohnwu/Magisk/
- Xposed cuenta con módulos que ayudan a evadir algunos controles de seguridad. Link: https://repo.xposed.info/
- Mobile Security Framework (MobSF) es una herramienta que ayuda en las pruebas estáticas y dinámicas en Android y iOS. Link: https://github.com/MobSF/Mobile-Security-Framework-MobSF

- Apktool es una herramienta que permite decompilar los binarios en Android (.apk). Link https://ibotpeaches.github.io/Apktool/
- Bytcode es una herramienta que permite visualizar el código java del binario en android. Link https://bytecodeviewer.com/
- Signapk https://github.com/techexpertize/SignApk

J.0.6. Interacción con la Plataforma

Lineamiento L7.1

Control definido: La aplicación requiere la mínima cantidad de permisos.
Pruebas de Validación: Cuando la aplicación expone componentes Inter process communication (IPC), se debe realizar las respectivas configuraciones con los permisos necesarios para evitar que aplicaciones maliciosas puedan acceder a dichos componentes.La forma de gestionar los permisos en Android es diferente al de iOS a nivel conceptual pero bajo un mismo objetivo de proteger la data y los accesos a los recursos. Para validar los permisos se debe realizar las siguientes acciones.

- Validaciones en Android:
  - Inicialmente se debe validar los permisos solicitados por la aplicación al sistema operativo android. En ocasiones las aplicaciones integran SDKs de terceros para cubrir ciertas funcionalidades. Bajo este escenario se puede dar el caso de incluir solicitud de permisos que no se requieren teniendo en cuenta el alcance de la solución, inyectando posibles riesgos de seguridad y una posible afectación de la reputación antes los clientes. Por esta razón se debe validar cada uno de los permisos solicitados al sistema operativo, apoyado de la herramienta Drozer por medio del comando: *dz¿run app.package.info -a package_name*.
  - Otro punto a tener en cuenta se relaciona con el componenete Provider. Si el componente es expuesto y no cuenta con permisos de lecturas y escritura, una atacante podrá realizar consultas directas a la base de datos. Para validar los permisos del componente provider se puede utilizar la herramienta Drozer por medio del siguiente comando: dz¿run app.provider.info -a package_name Como referencia para el uso de de los comandos, se puede acceder al siguiente link: https://mobiletools.mwrinfosecurity.com/Using-Drozer-for-application-security -assessments/
- Validaciones en iOS: En sistemas operativos iOS es un poco mas complejo este tipo de valdiaciones debido a que depende principalmente de un análisis estático para

identificar los diferentes permisos con los que cuenta la aplicación. En este punto se recomendaría utilizar la herramienta Mobile Security Framework (MobSF) para validar los permisos encontrados en la aplicación.

Herramientas de apoyo:

- Drozer: Herramienta utilizada para realizar pruebas de seguridad a las aplicaciones móviles en plataformas android. https://github.com/mwrlabs/drozer
- Mobile Security Framework (MobSF) es una herramienta que ayuda en las pruebas estáticas y dinámicas en Android y iOS. Link: https://github.com/MobSF/Mobile-Security-Framework-MobSF

Lineamiento L7.2

Control definido: La aplicación no exporta funcionalidades sensibles vía esquemas de URL, salvo que dichos mecanismos estén debidamente protegidos.
Pruebas de Validación: Tanto en plataformas iOS como Android permiten la comunicación entre aplicaciones móviles por medio de esquemas URL personalizados. Los Identificador de Recursos Uniforme (URI) definen que tipo de acción se va a realziar en la aplicación y que parámetros debe utilizar. Si los esquemas URL no son protegidos de forma adecuada, el usuario de la aplicación será susceptible a fuga de información o posibles afectaciones económicas. Las validaciones a realizar son las siguientes: https://github.com/OWASP/owasp-mstg/blob/
1.1.1/Document/0x05h-Testing-Platform-Interaction.md y Mobile App Hackers Handbook.pdf o IntentScheme.pdf

- En primer lugar se debe enumerar los esquemas de URL en la aplicación móvil, que tenga la posibilidad de ser llamadas por un navegador web. Para realizar esta validación se puede utilizar la herramienta Drozer, por medio del módulo scanner.activity.browsable. El comando a utilizar es: dz¿run scanner.activity.browsable -a NombrePaquete.
- Una vez identificados los posibles esquemas URL en el punto anterior, se debe ubicar en el código java de la aplicación, la función que utiliza el esquema url con el fin de construir correctamente el intent y enviar los parámetros requeridos por medio de la herramienta Drozer. Un ejemplo del comando a utilizar es: dz¿run app.activity.start –action xxxxaction.VIEW –data-uri "sms://xxxxxx".
- Este tipo de ataques son muy particulares y su explotación depende de la forma en que esté construido el esquema URL. Por lo anterior el siguiente documento es una guía útil que explica diferentes formas de explotar dicho vector en Android. Whitepaper – Attacking Android browsers via intent scheme URLs. En iOS se tiene las siguientes indicaciones dada por OWASP https://github.com/OWASP/owasp-mstg/blob/1.1.1/Document/0x06h-Testing-Platform-Interaction.md

Herramientas de apoyo:

- Drozer: Herramienta utilizada para realizar pruebas de seguridad a las aplicaciones móviles en plataformas android. https://github.com/mwrlabs/drozer
- JADX: Es una herramienta que permite decompilar de Dex a Java y sirve de cliente para visualizar el código fuente decompilado. Se puede cargar directamente el instalador .apk o archivos con extensión .jar. Página web de la herramienta: https://github.com/skylot/jadx

Lineamiento L7.3

Control definido: La aplicación no exporta funcionalidades sensibles a través de mecanismos IPC salvo que los mecanismos estén debidamente protegidos.
Pruebas de Validación: Dependiendo del alcance de la aplicación, los desarrollares utilizan los mecanismo IPC con el fin de permitir que otras aplicaciones consuma ciertos recursos expuesto por la aplicación. Para permitir esta comunicación, los recursos como por ejemplo los activities, broadcast receivers, content providers y services deben ser exportados. De no aplicar una correcta configuración, los mecanismos pueden llegar a exponer información personal o sensible del usuario. La validación consiste en:

- El primer paso consiste en identificar los componentes IPC existentes en la aplicación y que a su vez son configurados como "exported". Bajo esta característica, los componentes se convierten en un posible vector de ataque. La herramienta a utilizar para dicha tarea es Drozer, por medio del siguiente comando dz¿run app.package.attacksurface -a PacketName. El resultado definirá la superficie de ataque para iniciar las pruebas de seguridad.
- **Content Providers:** Este componente está principalmente relacionado al uso de base de datos internas de la aplicación, permitiendo utilizar técnicas del tipo inyección sql. Para validar su explotación se debe enumerar o identificar cuales son los content providers susceptibles a ser explotados por medio de la herramienta Drozer. El comando a utilizar es dz¿run app.provider.info -a PacketName y para identificar los URIs de los content providers se debe utilizar el módulo scanner.provider.finduris. Desde este punto y dependiendo de los resultados obtenidos, se puede utilizar los siguientes módulos de Drozer con el fin de acceder a información en bases de datos: app.provider.query, app.provider.insert, app.provider.update, app.provider.delete y para la inyección sql se puede utilizar la opción scanner.provider.injection. Para obtener una mejor guía sobre la ejecución de los comandos se puede remitir a la siguiente página https://mobile-security.gitbook.io/mobile-security-testing-guide/android-testing-guide/0x05d-testing-data-storage#testing-content-providers.

- **Broadcast Receivers:** Este componente está principalmente relacionado a la posibilidad de registrarse a eventos del sistema o de la aplicación. Un ejemplo sería las notificaciones push. Si el componente es exportado por la aplicación, el primer paso consiste en enumerar los Broadcast expuestos por medio de la herramienta Drozer con el comando dz¿run app.broadcast.info -a PacketName. Si el broadcast no requiere de permisos, se debe realizar un análisis estático con el fin de poder encontrar la función que realiza el llamado e identificar los parámetros utilizados para construir correctamente la petición. Con dichos datos se puede utilizar el módulo de Drozer app.broadcast.send para su ejecución. Adicionalmente se puede utilizar el siguiente módulo de Drozer run app.broadcast.sniff con el fin de realizar un monitoreo de los intents ejecutados dependiendo de la acción a monitorear y que especificada en la ejecución del módulo. Para obtener una mejor guía sobre la ejecución de los comandos se puede remitir a la siguiente página: https://github.com/OWASP/owasp-mstg/blob/1.1.1/Document/0x05h-Testing-Platform-Interaction.md#broadcast-receivers-1.

- **Activities:** Este componente está principalmente relacionado al llamado de pantallas en la aplicación. Uno de los principales llamados está relacionado al menú principal de la aplicación que normalmente es público, es decir que no requiere de una autenticación previa para visualizarla. Si el componente es exportado por la aplicación y son pantallas que solo son visualizadas después de una autenticación, existe la posibilidad de exposición de información del usuario debido a que una aplicación externa puede realizar el llamado directamente de la pantalla sin una autenticación previa. El primer paso consiste en enumerar los activities exportados por medio de la herramienta Drozer con el comando dz¿run app.activity.info -a PacketeName. Con la lista de de Activities susceptibles a explotar, se procede a realizar el llamado de cada uno de ellos por medio del módulo de Drozer app.activity.start. Para obtener una mejor guía sobre la ejecución de los comandos se puede remitir a la siguiente página: https://github.com/OWASP/owasp-mstg/blob/1.1.1/Document/0x05h-Testing-Platform-Interaction.md#activities-1.

- **Services:** Es un componente que puede ejecutarse en segundo plano y no requiere de la interacción del usuario con la aplicación, razón por la cual no necesita de una interfaz gráfica para su normal funcionamiento. Si el componente es exportado por la aplicación, se podrá consumir el servicio directamente y obtener información sensible del usuario o realizar cambios que permita a un atacante lograr el acceso a información privada o confidencial. El primer paso consiste en enumerar los servicios exportados por medio de la herramienta Drozer con el comando: dz¿run app.service.info -a PacketName. Si el servicio no requiere de permisos, se debe realizar un análisis estático con el fin de poder encontrar la función que realiza el llamado e identificar los parámetros utilizados para construir correctamente la petición. Con dichos datos se puede utilizar el módulo de Drozer app.service.send. Para obtener una mejor guía sobre la ejecución de los

comandos se puede remitir a la siguiente página: https://github.com/OWASP/owasp-mstg/blob/1.1.1/Document/0x05h-Testing-Platform-Interaction.md#services-1.

- Para el caso de sistema operativos iOS, el tipo de herramientas a utilizar es reducido y requiere principalmente de validaciones iniciales de código para su explotación. La herramienta utilizada para estos casos es Frida. Una guía para su uso se expone en el siguiente link: https://github.com/OWASP/owasp-mstg/blob/1.1.1/Document/0x06h-Testing-Platform-Interaction.md.

Herramientas de apoyo:

- Drozer: Herramienta utilizada para realizar pruebas de seguridad a las aplicaciones móviles en plataformas android. https://github.com/mwrlabs/drozer
- JADX: Es una herramienta que permite decompilar de Dex a Java y sirve de cliente para visualizar el código fuente decompilado. Se puede cargar directamente el instalador .apk o archivos con extensión .jar. Página web de la herramienta: https://github.com/skylot/jadx
- Firda: una herramienta tipo cliente - servidor que funciona en dispositivos móviles sin requerir en algunos casos acceso como súper usuario o root. Permite inyecciones de scripts y realizar "Hoo" a las funciones cuando la aplicación se está ejecutando. La página web de la herrmaienta es: https://www.frida.re/docs/ios/

Lineamiento L7.4

Control definido: JavaScript se encuentra deshabilitado en los WebViews salvo que sea necesario.
Pruebas de Validación: Webview es un navegador web inmerso en una aplicación móvil pero con limitantes funcionales. Su entorno de ejecución es independiente al navegador nativo de la aplicación y utiliza la sandbox propia de la aplicación. Bajo esta defineión, los webview son susceptible a ataques de inyección de código como cualquier navegador web. Las pruebas a realizar son:

- La primera prueba consiste en inyectar código java script desde puntos de entrada disponibles en la aplicación, que normalmente suelen ser formularios o campos con el que interactúa el usuario. El fin es explotar vulnerabilidades del tipo Cross Site Scripting XSS del tipo reflejada, almacenada o DOM. El ataque es muy parecido al realizado en páginas web.
- La segunda prueba consiste en inyectar código java script desde la mensajería enviada por la aplicación al backend, por medio de un ataque de hombre en el medio. El fin es explotar vulnerabilidades del tipo Cross Site Scripting XSS del tipo reflejada, almacenada o DOM. El ataque es muy parecido al realizado en páginas web.

- La tercera prueba consiste en redireccionar la página web original a otra página clonada con el fin de obtener las credenciales de acceso a a la aplicación. Este tipo de ataque conocido como phishing se logra mediante la modificación de la cabecera de redireccionamientos conocida como "Location" como parte de la respuesta del backend o manipulando el body del response por medio de inyección de scripts.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend y modificar el tipo de certificado a utilizar.

Lineamiento L7.5

Control definido: 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.
Pruebas de Validación: Los webview tienen la propiedad de cargar contenido de forma remota o cargar contenido local ubicado en la sandbox de la aplicación, aprovechando manejadores (Handlers) considerados peligrosos, como es el caso de file:// o realizar llamadas sin autorización. Las pruebas a realizar son:

- Inicialmente se debe identificar que tipo de uso se le está dando a los webview y que manejadores está utilizando la aplicación por medio de análisis estático. En el siguiente link se detalla el análisis estático a realizar para sistemas operativos andorid: https://github.com/OWASP/owasp-mstg/blob/1.1.1/Document/0x05h-Testing-Platform-Interaction.md#testing-webview-protocol-handlers. Para sistemas operativos iOS se puede acceder al siguiente link: https://github.com/OWASP/owasp-mstg/blob/1.1.1/Document/0x06h-Testing-Platform-Interaction.md/#testing-webview-protocol-handlers.

- La prueba consiste en acceder a información personal del usuario o del sistema, que están contenidas en archivos almacenados en la sandbox de la aplicación. La lectura de estos archivos es posible si la aplicación cuenta con configuraciones débiles relacionadas al manejo de los webview. El vector de ataque se relaciona principalmente al uso de file scheme. por ejemplo en sistemas operativos android se puede intentar inyectar file://data/data/com.whatsapp/shared_prefs/preferemces.xml. En sistemas operativos iOS se puede intentar inyectar file://private/var/mobile/Containers/Data/Application/¡UUID¿/Library/Preferences/file.plist.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para capturar todas las peticiones realizadas entre la aplicación móvil y el backend y modificar el tipo de certificado a utilizar.

- JADX: Es una herramienta que permite decompilar de Dex a Java y sirve de cliente para visualizar el código fuente decompilado. Se puede cargar directamente el instalador .apk o archivos con extensión .jar. La página de la herramienta: https://github.com/skylot/jadx. En el caso de iOS se puede utilizar OTOOL incorporada en las utilidade de liena de comandos en XCODE. Una guia de eso se encuentra en la siguiente página: https://resources.infosecinstitute.com/penetration-testing-for-iphone-applications-part-5/#gref.

J.0.7. Calidad de Código y Configuración del Compilador

Lineamiento L8.1

Control definido: La aplicación es firmada bajo las versiones 1 y 2 y provista con un certificado válido.
Pruebas de Validación: El instalador de las aplicaciones móviles se firman antes de ser cargadas en las respectivas tiendas. Si la firma no cuenta con un algoritmo seguro (SHA1, MD5, etc), un atacante podrá inyectar código malicioso en la aplicación original y firmarla nuevamente obteniendo el mismo valor hash de los archivos originales logrando engañar a los controles antimalware de los dispositivos móviles para evitar cualquier alertamiento. Para este lineamiento solo aplica validaciones de tipo estático siguiendo los pasos descritos a continuación.

- En Android, La primerava validación consiste en identificar si la aplicación utiliza el esquema de firma APK v1. El no utilizar el esquema de firma APK v2 sería susceptible a la vulnerabilidad CVE-2017-13156, la cual permite inyectar malware sin afectar la firma original. La prueba a realizar es por medio de la herramienta "apksigner" con la opción "verify", quien indicará bajo su análisis que versión de esquema se está utilizando en la aplicación. De ser vulnerable, la prueba de concepto a realizar es la explotación de la vulnerabilidad por medio de la herramienta "janus".

- La siguiente validación consiste en identificar el tipo de algoritmo utilizado para la firma de la aplicación. Si el algoritmo utilizado es considerado vulnerable como por ejemplo un MD5, SHA1 entre otros, la aplicación puede ser modificada y su vez mantener su valor de integridad original. La validación se puede realizar con la herramienta Mobile Security Framework (MobSF) en su módulo "Signer Certificate", tanto en android como en iOS.

Herramientas de apoyo:

- Apksigner: Forma parte de las herramientas de compilación del SDK de Android. Su uso se puede validar en el siguiente link: https://developer.android.com/studio/command-line/apksigner

- Mobile Security Framework (MobSF) es una herramienta que ayuda en las pruebas estáticas y dinámicas en Android y iOS. Link: https://github.com/MobSF/MobileSecurity-Framework-MobSF

- Janus, es una herramienta creada en lenguaje de programación python que permite explotar la vulnerabilidad CVE-2017-13156. La url que permite la descarga y describe su uso es: https://github.com/odensc/janus

Lineamiento L8.2

Control definido: La aplicación fue liberada en modo release y con las configuraciones apropiadas para el mismo. No debuggable y sin Backup activo.
Pruebas de Validación: En la etapa de desarrollo, los instaladores generados normalmente tiene la opción de debug activo para realizar una trazabilidad a los diferentes errores que se pueden presentar en las pruebas funcionales. Para un atacante o un malware, el debug le permite interactuar con la aplicación por medio de herramientas de desarrollo como es el caso de Android Debug Bridg (adb), que permite acceder a la sandbox de la aplicación y extraer información. También se puede utilizar herramientas como andbug para realizar un seguimiento paso a paso de una funcionalidad con el fin de obtener información privada y sensible de la aplicación como es el caso de llaves de cifrado o data privada del usuario. Normalmente los instaladores tienen habilitada la opción de backup, que genera un riesgo de exposición de información debido a que un atacante o un malware podrá extraer información privada de la sandbox de la aplicación sin contar con permisos de root.

- Para validar si la aplicación en su configuración tiene activo la opción de debug, una de las formas de realizar la validación es por medio de pruebas estáticas. En el caso de android, el análisis se realiza al archivo manifest mediante la decompilación de la aplicación o automatizando el proceso por medio de la herramienta Mobile Security Framework (MobSF) en el módulo “Manifest Analysis”. A nivel dinámico la validación se puede realizar con la herramienta Drozer en el módulo app.package.attacksurface. El comando a utilizar es dz¿run app.package.attacksurface PacketName.

- Para validar si la aplicación en su configuración tiene activo la opción de backup, una de las formas de realizar la validación es por medio de pruebas estáticas. En el caso de android, el análisis se realiza al archivo manifest mediante la decompilación de la aplicación o automatizando el proceso por medio de la herramienta Mobile Security Framework (MobSF) en el módulo “Manifest Analysis”.

Herramientas de apoyo:

- Mobile Security Framework (MobSF) es una herramienta que ayuda en las pruebas estáticas y dinámicas en Android y iOS. Link: https://github.com/MobSF/MobileSecurity-Framework-MobSF

- El detalle de la ejecución de los comandos en Drozer se pueden consultar en el siguiente link:
  https://mobiletools.mwrinfosecurity.com/Using-Drozer-for-applicationsecurity-assessments/.

Lineamiento L8.3

Control definido: La aplicación captura y maneja debidamente las posibles excepciones.
Pruebas de Validación: En ocasiones las aplicaciones pueden entrar en un estado no esperado por diferentes causas, generando una excepción. La excepción puede ser originada por defectos o bugs propios en la aplicación o de forma intencional, al obligar a la aplicación a recibir datos no esperados o alterando su flujo funcional. En consecuencia, la aplicación responde con errores que pueden revelar información útil para un atacante, relacionada al componente afectado (Sistema operativo, plataforma web, base de datos entre otros). Las pruebas a realizar son:

- Se debe identificar todos los puntos de entrada existentes en la aplicación, que puede ser un formulario o variables que no se visualizan en el front pero viajan en el request.
- Se debe interceptar las comunicaciones por medio de un webproxy, con el fin de realizar las diferentes inyecciones y manipulaciones en la mensajería entre la aplicación y el backend.
- Con los puntos de entrada previamente identificados y con las comunicaciones interceptadas con un webproxy, se debe:
  - Inyectar caracteres especiales y analizar la respuesta del backend.
  - Ingresar valores fuera del formato propio del dato, es decir si es solo numérico se debe inyectar letras y analizar como se comporta el backend.
  - Si el contenido de las variables en la mensajería están definidas por un valor numérico, como posible índice de referencia a una base de datos, se debe inyectar valores mayores hasta lograr un comportamiento no esperado de la aplicación.
  - Modificar los flujos de la aplicación mediante la inyección de parámetros o urls.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para interceptar y modificar la mensajería entre la aplicación móvil y el backend.

Lineamiento L8.4

Control definido: La lógica de manejo de errores en los controles de seguridad deniega el acceso por defecto.

Pruebas de Validación: A nivel de arquitectura de las aplicaciones, diferentes controles de seguridad son incorporados con el fin de proteger la aplicación de diferentes tipos de ataques que puedan comprometer la disponibilidad, integridad y confidencialidad de la información. En caso de generarse un error, por ejemplo a nivel de comunicaciones ya sea intencional o no, los controles de seguridad deben por defecto denegar el acceso al recurso protegido. Algunos ejemplos son, funciones captcha, análisis de riesgo, módulos de autenticación entre otros. Las pruebas a realizar son:

- Inicialmente se debe identificar los diferentes controles de seguridad susceptibles a ser validados, bajo el contexto del lineamiento de seguridad propuesto. Los controles a tener en cuenta principalmente están relacionadas a la autenticación, autorización y control de acceso.
- En primera instancia las pruebas se deben realizar generando errores de comunicación controlados, entre el backend y el componente de seguridad. Bajo este escenario se debe validar si la aplicación deniega o no el acceso.
- En caso de no lograr los errores de comunicación de forma controlada, se puede intentar realizar otro tipo de interrupciones como por ejemplo apagar y encender la red wifi o realizar inyecciones de caracteres especiales que generen comportamientos no esperados por la aplicación. Es de aclarar que los métodos especificados en este ítem no es igual de efectivo al ítem anterior.

Herramientas de apoyo:

- Webproxy Owasp Zap o BurpSuite para interceptar y modificar la mensajería entre la aplicación móvil y el backend.

J.0.8. Impedir el Análisis Dinámico y la Manipulación

Lineamiento L9.1

Control definido: La aplicación detecta y responde a la presencia de un dispositivo rooted o con jailbreak, ya sea alertando al usuario o finalizando la ejecución de la aplicación.
Pruebas de Validación: Normalmente para el desarrollo de las pruebas de seguridad en aplicaciones móviles, los hackers éticos y ciberdelincuentes utilizan dispositivos móviles rooted para android y con jailbreak para iOS, con el fin de instalar una serie de herramientas para el análisis de la aplicación y lograr accesos al sistema operativo que normalmente no se puede obtener desde un dispositivo móvil sin root. Este control ayuda a evitar el análisis dinámico de la aplicación. Otro grupo de usuarios disponen de dispositivos móviles con root con el fin de poder instalar aplicaciones móviles de pago de forma gratuita y realizar personalizaciones en la configuración que son bloqueados de fábrica. Utilizar aplicaciones móviles en dispositivos con acceso a root, permiten que un malware logre una mayor afectación de la

privacidad de la información gestionada por la aplicación móvil. Para validar el lineamiento se debe:

- Para la prueba se requiere de un dispositivo con Jailbreak y otro Rooted con el fin de validar el comportamiento de la aplicación al momento de su ejecución. Bajo este escenario, la aplicación no debe permitir su apertura y debe alertar al usuario sobre el riesgo de seguridad que representa el dispositivo móvil o simplemente generando el cierre de la aplicación.

Herramientas de apoyo:

- Las pruebas a realizar no requiere de herramientas adicionales o especializadas debido a que su alcance es a nivel funcional.

Lineamiento L9.2

Control definido: La aplicación previene el debugging o detecta y responde al debugging de la aplicación. Se deben cubrir todos los protocolos.
Pruebas de Validación: Un atacante utiliza la opción de debug en la aplicación con el fin de evaluar en tiempo de ejecución las diferentes funcionalidades y crear puntos de interrupción para capturar los valores de las diferentes variables y acceder de esta forma a información privada de la aplicación que permita saltar ciertos controles de seguridad como por ejemplo las validaciones de root o el control de certificados SSL Pinning, entre otros. Las validaciones a realizar son:

- Para realizar la prueba en android, la aplicación debe tener activa la opción de debuggable en el manifest y estar abierta en dispositivo móvil. En caso de no estar activa (true), se debe realizar el proceso de decompilación, modificación, re compilación y firma del instalador. Si la instalar la aplicación modificada se logra ejecutarla correctamente, los controles para el cumplimiento del lineamiento son débiles.

- Con el debug activo en la aplicación, el siguiente paso consiste en validar la conexión entre la maquina de pruebas y el dispositivo móvil por medio del comando “adb devices”. Para una mejor visualización del proceso que está corriendo la aplicación a validar, es recomendable solo dejar la aplicación de interés abierta y proceder a ejecutar el comando “adb jdwp”, quien dará como resultado la entrega del proceso de la aplicación en ejecución. Luego se procede a ejecutar el siguiente comando para establecer la conexión con el dispositivo móvil “adb forward tcp:55555 jdwp:id-Proceso” y por último se ejecuta la aplicación jdb para iniciar el proceso de debug con el comando “jdb -attach localhost:55555”. Si la aplicación es susceptible a ser analizada, se observará la inicialización de la aplicación “Initializing jdb ...” quedando en espera de recibir los diferentes comando propios de la herramienta, como por ejemplo el listar las diferentes clases con el comando “classes”. En el siguiente link se podrá encontrar un ejemplo

de como utilizar jdb. https://securitygrind.com/how-to-exploit-a-debuggable-android-application/

Herramientas de apoyo:

- Android Debug Bridg (adb), con la opción jdwp para obtener el número de proceso de la aplciación a validar. link: http://adbshell.com/commands/adb-pull
- jdb: Es un debugger de java en linea de comandos. Un guia para su uso se encuentra en el siguiente link: https://docs.oracle.com/javase/7/docs/technotes/tools/windows/jdb.html
- Andbug: Debugger utilizado por linea de comandos. La descripción de esta herramienta se encuentra en el siguiente link: https://github.com/swdunlop/AndBug.

Lineamiento L9.3

Control definido: La aplicación detecta y responde a modificaciones de ejecutables y datos críticos de la propia aplicación.
Pruebas de Validación: Una de las formas de evadir ciertos tipos de controles de seguridad implementados en la aplicación, consiste en alterar el instalador a nivel de configuración y modificación del código fuente. La modificación del binario permite en algunos casos evadir la validación de certificados, detección de equipos con root, activar el debug para realizar análisis en tiempo de ejecución, modificar los flujos funcionales de la aplicación cambiando la lógica desde el código fuente entre otros. Por lo anterior, la aplicación debe contar con controles eficaces que ayuden a evitar alteraciones en el binario generado. Las pruebas a realizar son:

- Una de las formas de validar si la aplicación cuenta o no con controles de seguridad que evite la alteración del binario, consiste en lograr ejecutar de forma normal la aplicación después de ser decompilada, recompilada y firmada nuevamente. Al superar este proceso se puede intentar modificar algunas configuraciones en el manifest e intentar ejecutar nuevamente la aplicación. De igual forma se debe realizar modificaciones en los archivos smali con el fin de realizar cambios funcionales.
- Para la decompilación y recompilación del binario se recomienda utilizar la herramienta Apktool y utilizar sus diferentes opciones de decompilación para evadir los controles que evitan la modificación del binario. Por ejemplo se puede intentar la decompilación solo para modificar el manifest o solo para modificar los archivos smali.

Herramientas de apoyo:

- Apktool: Herramienta de reversing para aplciaciones Android. link: https://github.com/iBotPeaches/Apktool

- Signapk: Herramienta que permite firmar las aplicaciones de andorid link:
  . https://github.com/techexpertize/SignApk

- Bytcode: es una herramienta que permite visualizar el código java del binario en android. link: https://bytecodeviewer.com/

Lineamiento L9.4

Control definido: La aplicación detecta la presencia de las herramientas de ingeniería reversa o frameworks mas utilizados.
Pruebas de Validación: Al momento de realizar pruebas de ingeniería inversa en las aplicaciones móviles, los atacantes suelen utilizar una serie de herramientas que facilitan dicha tarea y son instaladas en los dispositivos móviles. El fin del lineamiento es evitar que la aplicación se ejecute en caso de detectar este tipo de herramientas, dificultando el actuar del atacante. Las pruebas a realizar son:

- La primera actividad consiste en identificar el conjunto de herramientas que son utilizadas para analizar las aplicaciones móviles en tiempo de ejecución. Un ejemplo de herramientas utilizadas para este tipo de análisis son: Frida, Xposed, Objection, Introspy-Android, Drozer, Magisk entre otros.

- La siguiente actividad consiste en instalar cada una de las herramientas previamente identificadas de forma individual y probar la posibilidad de ejecutar la aplicación y que sea funcional.

Herramientas de apoyo:

- **Frida:** una herramienta tipo cliente - servidor que funciona en dispositivos móviles sin requerir en algunos casos acceso como súper usuario o root. Permite inyecciones de scripts y realizar "Hook" a las funciones cuando la aplicación se está ejecutando. La página web de la herrmaienta es:
  https://www.frida.re/docs/ios/ o https://www.frida.re/docs/android/

- **Xposed:** cuenta con módulos que ayudan a evadir algunos controles de seguridad. Link: https://repo.xposed.info/

- **Objection:** Herramienta de análisis en tiempo de ejecución de aplicaciones móviles iOS, sin la necesidad de contar con un dispositivo móvil con jailbreak. La página web es https://github.com/sensepost/objection.

- **Introspy-Android:** Herramienta que permite el análisis de aplicaciones móviles en dispositivos android en tiempo de ejecución. https://github.com/iSECPartners/Introspy-Android.

- **Drozer:** Herramienta utilizada para realizar pruebas de seguridad a las aplicaciones móviles en plataformas android. La página web es https://github.com/mwrlabs/drozer.
- **Magisk:** Cuenta con módulos que ayudan a evadir algunos controles de seguridad. Link: https://github.com/topjohnwu/Magisk/.

Lineamiento L9.5

Control definido: La aplicación detecta y responde al ser ejecutada en un emulador.
Pruebas de Validación: Una de las formas mas sencillas para validar una aplicación móvil sin tener a disposición un dispositivo móvil físico y con acceso a root, es por medio de emuladores. Los emuladores permiten además del acceso como root al dispositivo, permite instalar programas que ayudan a realizar las pruebas de ingeniería inversa sobre la aplicación. Por lo anterior, el fin del lineamiento es dificultar el análisis en tiempo de ejecución de la aplicación al atacante, obligándolo a buscar técnicas que le permita saltar el control o de buscar un móvil y realizar el proceso de rooted.

- La validación consiste en instalar la aplicación en un emulador, ejecutarla y acceder a sus diferentes funcionalidades. En el mercado existente diferentes tipos de emuladores, entre los mas conocidos está el emulador de Android Studio y Genymotion.

Herramientas de apoyo:

- Android Studuio: Es ei IDE oficial de Android y cuenta con la opción de crear emuladores. La url es https://developer.android.com/studio
- Genymotion: Es un emulador para Android en arquitectura x86. que propone una ejecución rápida y fluida. La url de la aplicación es https://www.genymotion.com

Lineamiento L9.6

Control definido: La aplicación implementa múltiples mecanismos de detección (Resiliencia) para los puntos del 9.1 al 9.5. Nótese que a mayor cantidad y diversidad de mecanismos usados, mayor la resistencia.
Pruebas de Validación: El fin del lineamiento es validar que los controles aplicados en los puntos del 9.1 al 9.5 se implementaron de forma efectiva. Un atacante intentará buscar la forma de evadir los controles utilizando diferentes técnicas. Cada uno de estos lineamientos se soportan entre sí para brindar una mayor efectividad en el control definido.

- Si la aplicación cuenta con controles para validar si el dispositivo móvil está con acceso a root, tiene activo el debug, permite modificar el binario, detecta herramientas de ingeniería inversa o entornos emulados, se debe identificar que vectores de ataque pueden ser utilizados para intentar evadir el control, por ejemplo el control que permite identificar equipos con root, se puede evadir intentando modificar el código

fuente en el binario, también se puede intentar utilizando los framework como Frida, Xposed, Magisk entre otros que permiten cumplir con dicho objetivo. Una guia para este tipo de pruebas se puede encontrar en las guías de OWASP. El link de la página web es: https://github.com/OWASP/owasp-mstg/blob/1.1.1/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md

Lineamiento L9.7

Control definido: La ofuscación es aplicada a las defensas del programa y código funcional, lo que a su vez impide la des-ofuscación mediante el análisis dinámico.
Pruebas de Validación: El proceso de ofuscación corresponde a la transformación del código y la data, con el fin de dificultar su compresión y evitar que un atacante entienda las funciones relacionada a la lógica de negocio y procesos críticos. El entendimiento del código en funciones críticas de la aplicación, le permite a un atacante alterar el comportamiento de la aplicación, mediante la modificación del código java en el binario o en tiempo de ejecución. Aunque el lineamiento corresponde más a pruebas estáticas, es relevante a aclarar que la ofuscación del código es un método que dificulta o limita el uso de otros vectores de ataque del tipo dinámico.

- Para la validación de la prueba se pueden utilizar herramientas de análisis de código estático que permitan ver el código java de la aplicación móvil o los archivos .js que contenga el instalador. El fin de la prueba consiste en ubicar las clases que contenga funciones críticas, un ejemplo son los flujos de cifrado, la autenticación entre otros. Si al validar el código se entiende de forma clara el flujo funcional, quiere decir que no se está ofuscado el código o al menos la función analizada. Es de recordar que no es necesario aplicar este control a todo el código, lo importante para el lineamiento es proteger las funciones críticas de la lógica de negocio.

Herramientas de apoyo:

- Mobile Security Framework (MobSF) es una herramienta que ayuda en las pruebas estáticas y dinámicas en Android y iOS. Link: https://github.com/MobSF/MobileSecurity-Framework-MobSF

- JADX: Es una herramienta que permite decompilar de Dex a Java y sirve de cliente para visualizar el código fuente decompilado. Se puede cargar directamente el instalador .apk o archivos con extensión .jar. La página web de la herramienta es: https://github.com/skylot/jadx

Bibliografía

[1] S. Krishnan, "A Hybrid Approach to Threat Modelling," febrero 2017.

[2] R. Martinez, *Arquitectura para la Implementación de Sistemas Móviles basados en servicios de Geolocalización y Crowdsourcing (Tesis de Maestría)*. Universidad Abierta Interamericana, Argentina, 2015.

[3] K. Huckvale, J. T. Prieto, M. Tilney, P.-J. Benghozi, and J. Car, "Unaddressed privacy risks in accredited health and wellness apps: A cross-sectional systematic assessment," *BMC medicine*, vol. 13, p. 214, 09 2015.

[4] A. Mense, P. Urbauer, S. Sauermann, and H. Wahl, "Simulation environment for testing security and privacy of mobile health apps," *Modeling and Simulation in Medicine Symposium*, abril 2016.

[5] Congreso de Colombia, "Ley Estatutaria 1581," octubre 2012.

[6] NowSecure, "Mobile security report," Marzo 2018, accedido 25-10-2019. [Online]. Available: https://books.nowsecure.com/mobile-security-report/en/index.html

[7] Health Insurance Portability and Accountability Act, "Largest healthcare data breaches of 2017," 2016, accedido 25-10-2019. [Online]. Available: https://www.hipaajournal.com/largest-healthcare-data-breaches-2017

[8] Verizon, "Protected health information data breach report," p. 15, 2018, accedido 25-10-2019. [Online]. Available: https://enterprise.verizon.com/resources/reports/2018/protected_health_information_data_breach_report.pdf

[9] J. Mirkovic, E. Skipenes, E. K. Christiansen, and H. Bryhni, "Security and privacy legislation guidelines for developing personal health records," *2015 Second International Conference on eDemocracy and eGovernment, ICEDEG 2015*, 2015.

[10] Ponemon Institute, "Sixth annual benchmark study on privacy & security of healthcare data," *Ponemon Institute Research Report, 8*, 2016.

[11] B. Gleeson, "Survey: Enterprise security pros doubtful they can prevent mobile breaches," 2017, accedido 25-10-2019. [Online]. Available: https://blog.checkpoint.com/2017/04/12/survey-enterprise-security-pros-doubtful-can-prevent-mobile-breaches/

[12] Superintendencia de Industria y Comercio, "Por violaciones de datos personales, superindustria ha impuesto sanciones por más de $21 mil millones de pesos," 2017, accedido 25-10-2019. [Online]. Available: http://www.sic.gov.co/noticias/por-violaciones-de-datos-personales-superindustria-ha-impuesto-sanciones-por-mas-de-21-mil-millones-de-pesos

[13] Superintendencia Industria y Comercio, "Sanciones de protección de datos personales 2016," 2016, accedido 25-10-2019. [Online]. Available: http://www.sic.gov.co/sanciones-2016

[14] Y. Stefinko, A. Piskozub, and R. Banakh, "Manual and automated penetration testing. benefits and drawbacks. modern tendency," in *Modern Problems of Radio Engineering, Telecommunications and Computer Science, Proceedings of the 13th International Conference on TCSET 2016*, 2016, pp. 488–491.

[15] M. Felderer, M. Buchler, M. Johns, R. Achim D. B., Breu, and A. Pretschner, "Chapter one - security testing: A survey," in *Advances in Computers*. Elsevier, 2016, pp. 1–51.

[16] C. Treacy and F. McCaffery, "Medical mobile apps data security overview." The Second International Conference on Advances and Trends in Software Engineering. SOFTENG 2016 Lisbon, Portugal: IARIA XPA Press, 2016.

[17] M. Papageorgiou, A.and Strigkos, E. Politou, E. Alepis, A. Solanas, and C. Patsakis, "Security and privacy analysis of mobile health applications: The alarming state of practice." *IEEE Access*, vol. 6, pp. 9390–9403, 2018.

[18] A. Ahmad, K. Li, C. Feng, S. M. Asim, A. Yousif, and S. Ge, "An empirical study of investigating mobile applications development challenges," *IEEE Access*, vol. PP, marzo 2018.

[19] J. A. Sancho, "¿App Nativa o Híbrida? Difícil decisión...," 2017. [Online]. Available: https://justdigital.agency/app-nativa-vs-hibrida/

[20] S. Xanthopoulos and S. Xinogalos, "A comparative analysis of cross-platform development approaches for mobile applications," *ACM International Conference Proceeding Series*, septiembre 2013.

[21] IEEE-SA Standards Board, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems," *IEEE Std*, vol. 1471-2000, pp. 1–23, 2000.

[22] INCIBE, "Unidad 2: Estudio de las de las arquitecturas de tecnologías móviles," *INCIBE*, p. 19, 2017.

[23] INCIBE , "Glosario de términos de ciberseguridad: Una guía de aproximación para el empresario," *INCIBE*, p. 33, 2017.

[24] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," enero 2000.

[25] K. Mockford, "Web services architecture," *BT Technology Journal*, vol. 22, pp. 19–26, enero 2004.

[26] P. Saint-Andre, "RFC Standard 6455– The WebSocket Protocol," *RFC 6455 (Proposed Standard)*, 2011.

[27] Open Web Application Security Project, "Owasp mobile security testing guide." GitHub, 2018, accedido 25-10-2019. [Online]. Available: https://github.com/OWASP/owasp-mstg

[28] INCIBE, "Unidad 5: Desarrollo seguro de aplicaciones móviles," *INCIBE*, p. 56, 2017.

[29] Open Web Application Security Project, "Authentication cheat," 2017, accedido 25-10-2019. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

[30] Open Web Application Security Project , "Authentication cheat," 2017, accedido 25-10-2019. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

[31] Open Web Application Security Project, "Top 10-2017 Top 10 - OWASP," 2017.

[32] Real Academia Española, "Diccionario de la lengua española," 2017.

[33] A. Van der Stock and D. Cuthbert, "Application Security Verification Standard," *OWASP*, 2015.

[34] C. Taejoo, K. Hyunki, and H. Y. Jeong, "Security assessment of code obfuscation based on dynamic monitoring in android things," *IEEE Access*, vol. PP, abril 2017.

[35] A. Casanova, "Seguridad en las comunicaciones móviles," 2016, accedido 25-10-2019. [Online]. Available: https://digimodes.wordpress.com/2016/05/08/seguridad-en-las-comunicaciones-moviles-hack-beers-valencia-2016/

[36] Open Web Application Security Project, "Transport layer protection cheat sheet," 2018, accedido 25-10-2019. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html

[37] A. C. Gutierrez, "¿qué es y por qué hacer un análisis de riesgos?" agosto 2012, accedido 25-10-2019. [Online]. Available: https://www.welivesecurity.com/la-es/2012/08/16/en-que-consiste-analisis-riesgos/

[38] Instituto Colombiano de Normas Técnicas y Certificación. ICONTEC, "Norma técnica colombiana ntc-iso/iec 27005: Tecnología de la información. técnicas de seguridad. gestión del riesgo en la seguridad de la información." p. 3, 2009.

[39] © ISO/IEC 2014, "INTERNATIONAL STANDARD ISO / IEC Information technology — Security techniques — Information security management systems — Overview and," pp. 9–10, 2014.

[40] ICONTEC, "Norma Técnica NTC-ISO/IEC Colombiana 27001," *Icontec*, 2013.

[41] P. G. Perez, *Metasploit para Pentesters*, 2nd ed. Oxword, 2014, pp.23.

[42] P. Gonzalez, *Ethical hacking: Teoria y practica para la realizacion de un pentesting*. Oxword, 2014, p.13.

[43] W. Bonney, "Mobile health technologies - theories and applications." Rijeka: InTech, agosto 2016, p. 36.

[44] Organización Panamericana de la Salud, "Electronic medical records in latin america and the caribbean: An analysis of the current situation and recommendations for the region - 2016," 2016, accedido 25-10-2019. [Online]. Available: http://iris.paho.org/xmlui/handle/123456789/28210

[45] P. C. Tang, J. S. Ash, D. W. Bates, J. M. Overhage, and D. Z. Sands, "Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption," *Journal of the American Medical Informatics Association : JAMIA*, vol. 13, pp. 121–126, marzo 2006.

[46] P. Schwartz and D. Solove, "The PII problem: Privacy and a new concept of personally identifiable information," *New York University Law Review*, vol. 86, diciembre 2011.

[47] International Organization For Standardization, "Health informatics - Electronic health record - Definition, scope and context," *Definition of Electronic Health Record*, octubre 2005.

[48] J. Mirkovic, E. Skipenes, E. Christiansen, and H. Bryhni, "cloping personal health records," *2015 2nd International Conference on eDemocracy and eGovernment, ICEDEG 2015*, pp. 77–84, mayo 2015.

[49] S. U. Al Ayubi, A. Pelletier, G. Sunthara, N. Gujral, V. Mittal, and F. C. Bourgeois, "A Mobile App Development Guideline for Hospital Settings: Maximizing the Use of and Minimizing the Security Risks of "Bring Your Own Devices"Policies," *JMIR mHealth and uHealth*, 2016.

[50] C. Dorazio and K. K. R. Choo, "A generic process to identify vulnerabilities and design weaknesses in iOS healthcare apps," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2015.

[51] L. Martinez, A. Soto, L. Eraso, A. Ordóñez, and H. Ordoñez, "Towards guidelines for management and custody of electronic health records in Colombia," in *Communications in Computer and Information Science*, 2017.

[52] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner, "Chapter One – Security Testing: A Survey," in *Advances in Computers*, 2016.

[53] Superintendencia de Industria y Comercio, "Normativa," enero 2019, accedido 25-10-2019. [Online]. Available: https://www.sic.gov.co/repositorio-de-normatividad

[54] Ministerio de Salud, "Resolución número 1995 de 1999 - por la cual se establecen normas para el manejo de la historia clínica." 1999.

[55] Congreso de Colombia, "Ley Estatutaria 1266," diciembre 2008.

[56] Ministerio de Salud, "Ministerio de salud y protección social resolución número 1521 de 2014," 2014.

[57] Superintendencia de Industria y Comercio, "Título v protección de datos personales."

[58] Superintendencia Nacional de Salud, "Circular Única," noviembre 2007, accedido 25-10-2019. [Online]. Available: https://www.supersalud.gov.co/es-co/normatividad/circular-unica

[59] Superintendencia Financiera de Colombia, "Circular básica jurídica (c.e. 029/14)," vol. Parte I / Titulo II Capt I, noviembre 2014, accedido 25-10-2019. [Online]. Available: https://www.supersalud.gov.co/es-co/normatividad/circular-unica

[60] Health Insurance Portability and Accountability Act , "Security rule guidance material - administrative safeguards and technical safeguards," 2007, accedido 25-10-2019. [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/security/guidance/index.html

[61] Parlamento Europeo y del Consejo, "General data protection regulation GDPR," 2016, accedido 25-10-2019. [Online]. Available: http://www.privacy-regulation.eu/es/

[62] National Information Assurance Partnership, "Requirements for vetting mobile apps from the protection profile for application software," abril 2016, accedido 25-10-2019. [Online]. Available: https://www.niap-ccevs.org/MMO/PP/394.R/pp_app_v1.2_table-reqs.htm

[63] D. Medianero, "Open android security assessment methodology OASAM," 2016, accedido 25-10-2019. [Online]. Available: https://github.com/b66l/OASAM/

[64] M. Ogata, J. Franklin, J. Voas, V. Sritapan, and S. Quirolgico, "Vetting the security of mobile applications," in *Withdrawn NIST Technical Series Publication - Special Publication 800-163*, abril 2019.

[65] Open Web Application Security Project, "OWASP™ Foundation," 2019, accedido 25-10-2019. [Online]. Available: https://www.owasp.org/index.php/Main_Page

[66] Open Web Application Security Project , "OWASP Mobile Security Testing Guide," 2019, accedido 25-10-2019. [Online]. Available: https://github.com/OWASP/owasp-mstg/tree/master/Checklists

[67] M. Castellaro, S. Romaniz, J. C. Ramos, C. Feck, and I. Gaspoz, "Aplicar el Modelo de Amenazas para incluir la Seguridad en el Modelado de Sistemas," *V Congreso Iberoamericano de Seguridad Informática*, p. 5, 2009, llevado a cabo en Montevideo, Uruguay.

[68] A. Shostack, "Threat modeling: Designing for security," *Indianapolis, IN: Wiley*, pp. 61–80, 2014.

[69] A. Shostack, "Threat modeling: Designing for security," *Indianapolis, IN: Wiley*, pp. 87–98, 2014.

[70] A. Shostack, "Threat modeling: Designing for security," *Indianapolis, IN: Wiley*, pp. 101–108, 2014.

[71] Presidencia de la República, "GUÍA PARA LA CALIFICACIÓN DE LA INFORMACIÓN DE ACUERDO CON SUS NIVELES DE SEGURIDAD," pp. 11 –12, marzo 2017, versión 06.

[72] Congreso de la República, "Ley 1712: Por medio de la cual se crea la ley de transparencia y del derecho de acceso a la información pública nacional y se dictan otras disposiciones," pp. 3 –4, marzo 2014, artículo 6.

[73] I. Chorzevski, *Introduction to Android Application Hacking.* AppSec Labs, 2017, pp. 7 - 8.

[74] A. G. del Moral Miguel, *Malware en Android: discovering, reversing forensics*, primera ed. Mostoles (Madrid): ZeroXword computing, 2016, pp.35.

[75] Open Web Application Security Project, "Android cryptographic apis." GitHub, 2019, accedido 25-10-2019. [Online]. Available: https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05e-Testing-Cryptography.md

[76] Instituto Nacional de Ciberseguridad, "Algo que sabes, algo que tienes y algo que eres. ¿sabes de qué estamos hablando?" diciembre 2014, accedido 25-10-2019. [Online]. Available: https://www.incibe.es/protege-tu-empresa/blog/algo-sabes-tienes-eres-contrasenas-identidad-online

[77] Open Web Application Security Project , "Testing for insecure direct object references (otg-authz-004)," 2014, accedido 25-10-2019. [Online]. Available: https://www.owasp.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004)

[78] Open Web Application Security Project, "Mobile app authentication architectures." GitHub, 2019, accedido 25-10-2019. [Online]. Available: https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04e-Testing-Authentication-and-Session-Management.md

[79] Open Web Application Security Project , "Testing network communication." GitHub, 2019, accedido 25-10-2019. [Online]. Available: https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04f-Testing-Network-Communication.md

[80] Open Web Application Security Project, "Code quality and build settings of android apps," 2014, accedido 25-10-2019. [Online]. Available: https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05i-Testing-Code-Quality-and-Build-Settings.md

[81] Open Web Application Security Project , "Android anti-reversing defenses." GitHub, 2019, accedido 25-10-2019. [Online]. Available: https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md

[82] Open Web Application Security Project, "Reporting," diciembre 2014, accedido 25-10-2019. [Online]. Available: https://www.owasp.org/index.php/Reporting

[83] L. N. C. Uivaru, "La protecciÓn de datos de caracter sensible en el Ámbito europeo," p. 42, 2017.

Glosario

ADB Android Debug Bridg. 146, 159, 162, 171

API Application Programming Interface. 15–17, 19, 20, 40, 87

BYOD Bring Your Own Device. 31

CAPEC Common Attack Pattern Enumeration and Classification. 55, 57

CORBA Common Object Request Broker Architecture. 18

CWE Common Weakness Enumeration. 55, 57

EHR Electronic Health Record. 29–31

EPS Entidades Prestadoras de Salud. 3

GDPR General Data Protection Regulation. 41, 43

HIPAA Health Insurance Portability and Ac-countability Act. 2, 30, 31, 36, 41–43, 46

IPC Inter Process Communica-tion. 76, 77, 98, 111, 112, 114, 117, 162, 163, 201, 203

MBT Model-based testing. 27

NIAP National Information Assurance Partnership. 37, 39

NIST National Institue of Standars and Technology. 37, 39

OASAM Open Android Security Assessment Methodology. 38, 39

OTP One Time Password. 74, 76, 108, 110, 182, 183, 185

OWASP Open Web Application Security Project. 22, 37, 39–43, 47, 54, 55, 57, 58, 78, 79, 91, 103, 121, 147–151, 164, 173, 189, 193

PCI Payment Card Industry. 4, 90, 114, 170

PHR Personal Health Record. 29, 30

PII Personally identifiable information. 29, 31, 86, 114, 158

QA Quality Assurance. 91, 157, 174

REST Representational State Transfer . 16, 58

SIC Superintendencia de Industria y Comercio. 30, 34, 35

SSL Open Web Application Security Project. 24, 97, 101, 197

STRIDE Spoofing, Tampering, Repudiation, Information disclosure, Denial of service y Elevation of privilege. 25, 50, 51, 57

TI Tecnología de la información. 42

TLS Open Web Application Security Project. 61, 97, 117, 197

XML Extensible Markup Language. 18, 21, 52, 86, 113, 156

XSS Cross Site Scripting. 71, 86, 90, 105, 113, 156, 169, 190, 205

XXE XML external entity. 156