ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA SOFTWARE

ANÁLISIS DE VULNERABILIDADES MEDIANTE PENTESTING A LA APLICACIÓN MÓVIL DESARROLLADA PARA EL CENTRO MÉDICO DE ESPECIALIDADES “LA DOLOROSA”

Trabajo de Integración Curricular

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO DE SOFTWARE

AUTOR:

FRANCISCO XAVIER MESIAS FLORES

Riobamba – Ecuador

2023

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA SOFTWARE

ANÁLISIS DE VULNERABILIDADES MEDIANTE PENTESTING A LA APLICACIÓN MÓVIL DESARROLLADA PARA EL CENTRO MÉDICO DE ESPECIALIDADES "LA DOLOROSA"

Trabajo de Integración Curricular

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO DE SOFTWARE

AUTOR: FRANCISCO XAVIER MESIAS FLORES

DIRECTOR: ING. MARCO VINICIO RAMOS VALENCIA

Riobamba – Ecuador

2023

Yo, Francisco Xavier Mesias Flores, declaro que el presente Trabajo de Integración Curricular es de mi autoría y los resultados de este son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Integración Curricular; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 11 de diciembre de 2023

Francisco Xavier Mesias Flores

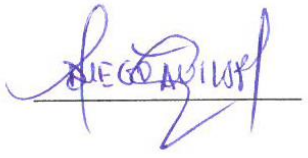
1105640534

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA SOFTWARE

El Tribunal del Trabajo de Integración Curricular certifica que: El trabajo de integración curricular; tipo: Proyecto Técnico, **ANÁLISIS DE VULNERABILIDADES MEDIANTE PENTESTING A LA APLICACIÓN MÓVIL DESARROLLADA PARA EL CENTRO MÉDICO DE ESPECIALIDADES "LA DOLOROSA"**, realizado por el señor: **FRANCISCO XAVIER MESIAS FLORES**, ha sido minuciosamente revisado por los Miembros del Tribunal del trabajo de integración curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	FECHA
Ing. Gisel Katerine Bastidas Guacho **PRESIDENTE DEL TRIBUNAL**		2023-12-11
Ing. Marco Vincio Ramos Valencia **DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**		2023-12-11
Ing. Diego Fernando Avila Pesantez **ASESOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**		2023-12-11

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE GRAFICOS

ÍNDICE DE ANEXOS

RESUMEN

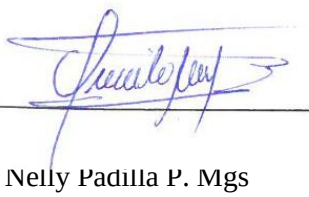
El presente trabajo se centró en la evaluación de la seguridad de la aplicación móvil del Centro de Especialidades "La Dolorosa", para identificar y mitigar las vulnerabilidades garantizando la seguridad de los datos de los pacientes. Para lograrlo, se utilizó una combinación de herramientas para el análisis estático y dinámico como MobSF, BurpSuite y SQLMap. Se aplicó la metodología Mobile Application Security (OWASP) en conjunto con la Guía de Pruebas de Seguridad de Aplicaciones Móviles (MASTG) de OWASP para la definición de las fases y el Estándar de Verificación de Seguridad de Aplicaciones Móviles (MASVS) para la corrección de las vulnerabilidades. Para la clasificación de las vulnerabilidades encontradas se utilizó el modelo DREAD de Microsoft. Posteriormente, se realizaron pruebas de penetración para validar las correcciones implementadas. Como resultado, se identificaron y confirmaron diez vulnerabilidades de nivel alto y medio en la aplicación móvil. Tras la implementación de las correcciones, se logró una reducción significativa del nivel de riesgo asociado a cada vulnerabilidad. El análisis posterior a la corrección mostró una reducción de riesgo del 48% con respecto al estado inicial de la aplicación. Las estrategias de mitigación basadas en un análisis exhaustivo de seguridad pueden mejorar significativamente la seguridad de las aplicaciones, reduciendo así el riesgo para los usuarios y aumentando la seguridad de los datos

Palabras clave: <SEGURIDAD DE APLICACIONES MÓVILES>, <PRUEBAS DE PENETRACIÓN>, <ANÁLISIS DE VULNERABILIDADES>, <MOBILE SECURITY FRAMEWORK (MOBSF)>, <BURPSUITE>, <SQLMAP>, <METODOLOGÍA OWASP>.

SUMMARY

This work focused on the evaluation of the security of the mobile application of "La Dolorosa" Specialty Medical Center, to identify and mitigate vulnerabilities, guaranteeing the security of patient data. To achieve this, a combination of tools for static and dynamic analysis such as MobSF,Burp Suite and SOL Map were used.. The Mobile Application Security (OWASP) methodology was applied in together with the OWASP Mobile Application Security Testing Guide (MASTG) for the definition of the phases and the Mobile Application Security Verification Standard (MASVS) for the correction of the vulnerabilities. Microsoft's DREAD model was used to classify the vulnerabilities found. Subsequently, penetration tests werecarried out to validate the implemented corrections. As a result, ten high- and medium-level vulnerabilities in the mobile application were identified and confirmed. After the implementation of the corrections, a significant reduction in the level of risk associated with each vulnerability was achieved. The post-correction analysis showed a 48% risk reduction compared to the initial state of the application. Mitigation strategies based on comprehensive security analysis can significantly improve application security, thereby reducing risk to users and increasing data security.

Keywords: <MOBILE APPLICATION SECURITY>, <PENETRATION TESTING>, <VULNERABILITY ANALYSIS>, <MOBILE SECURITY FRAMEWORK (MOBSF)>, <BURPSUITE>, <SQLMAP>, <OWASP METHODOLOGY>.



Lic. Nelly Padilla P. Mgs

0603818717

DOCENTE FIE

INTRODUCCIÓN

El avance constante en la era digital ha dado lugar a una proliferación de aplicaciones móviles, convirtiéndolas en plataformas críticas para compartir y acceder a la información. Esta dinámica abarca diversas facetas de nuestra vida cotidiana, incluyendo transacciones financieras, compras y consultas de todo tipo. Sin embargo, la creciente interdependencia en estas aplicaciones también ha abierto la puerta a una multitud de amenazas de seguridad cibernética. Existe técnicas de análisis de vulnerabilidades como el pentesting que se han convertido en herramientas indispensables para evaluar la seguridad de las aplicaciones móviles, reducir la vulnerabilidad ante los ciberdelincuentes y prevenir el robo y la malversación de información.

En la era digital actual, la información es un recurso esencial para empresas, organizaciones e individuos, abarcando desde datos personales hasta registros financieros y comerciales priorizando su seguridad. Una brecha en esta protección compromete la confidencialidad, integridad y disponibilidad, abriendo puertas a ciberdelincuentes que buscan extraerla con fines maliciosos, como robo de identidad o daño reputacional. Para brindar protección adecuada, es vital implementar medidas como autenticación biométrica, cifrado de datos, análisis de código y gestión de información, y después validarlas mediante pentesting, una simulación de ataque para detectar vulnerabilidades.

Las aplicaciones móviles de salud manejan información delicada, como prescripciones médicas, seguros, detalles de facturación, pruebas y diagnósticos, lo que hace que esta información sea atractiva para los ciberdelincuentes. Según datos de la Fiscalía del Ecuador en 2017, existieron 8421 casos de ciberdelincuencia, cantidad que va creciendo conforme pasa el tiempo debido a la digitalización. El presente trabajo de integración curricular tiene como finalidad analizar y evaluar las vulnerabilidades de la aplicación móvil CMED. Para la evaluación, se realizará mediante pentesting basándonos en la información que nos arroja, se procederá a la corrección de las vulnerabilidades de la aplicación, evitando que ciberdelincuentes puedan acceder a la información. Es importante la implementación de medidas de seguridad, debido a que maneja información sensible como prescripciones médicas, si se llegase a alterar, podría poner en riesgo la salud del paciente.

CAPÍTULO I

1 PLANTEAMIENTO DEL PROBLEMA

En el presente capítulo se describen los principales problemas relacionados a la seguridad de la información, y la importancia de implementar medidas de seguridad en aplicaciones mHeatlh, como es la aplicación móvil del centro médico “La Dolorosa”.

1.1 Antecedentes

El Pentesting, también conocido como pruebas de penetración, evalúa la seguridad de lossistemas informáticos simulando ataques reales usando herramientas específicas. Dado que las aplicaciones mHealth recopilan información personal y médica del usuario, es esencial realizar pruebas de seguridad para identificar y solucionar vulnerabilidades. Según la INTERPOL (2020), en el primer cuatrimestre hubo cerca de un millón de ataques dirigidos a infraestructuras de salud. La creciente dependencia del internet amplía la exposición a ciberataques, incluyendo phishing, malware, noticias falsas y dominios malignos. Estos ataques, en su mayoría, buscan extraer datos para fines como la extorsión y venta de información, destacando la importancia de mantener sistemas robustamente seguros.

La seguridad es una preocupación creciente en el mundo, no existe un sistema completamente seguro. De acuerdo con (IBM Corporation 2022) los delincuentes cibernéticos pueden hackear la mayoría de los dispositivos conectados a una red y, para 2025, podría haber más de 75 mil millones de dispositivos conectados a Internet. Los delincuentes cibernéticos actúan de manera grupal o individual, el costo de estos ataques sigue aumentando cada año y el costo promedio de un ataque a los datos de una empresa es de 4,82 millones de dólares en 2022, además de que la organización pierde la reputación y confianza de sus clientes.

Los datos personales y médicos de los pacientes son especialmente valiosos para los ciberdelincuentes. Estos datos pueden ser utilizados para cometer robos de identidad, fraudes, ventas de información y falsificación de recetas médicas. Además, la utilización indebida de esta información puede resultar en la obtención incorrecta de tratamientos y medicamentos, comprometiendo seriamente la salud del paciente. Un claro ejemplo de la magnitud de estas amenazas fue en marzo de 2019, cuando 12,5 millones de historias clínicas de mujeres embarazadas en India fueron expuestas, según ENISA (2020, p. 4).

Para la protección de los datos a través del análisis de vulnerabilidades se usa las pruebas de penetración y puede finalizar al encontrar una vulnerabilidad o cuando se explote la vulnerabilidad encontrada. Estas pruebas se pueden realizar en la web, IoT, aplicaciones móviles entre otros. Los dispositivos más utilizados se encuentra el smartphone. En 2017 se encuestó a profesionales de seguridad y uno de cada tres reportaron algún agravante relacionadocon un dispositivo móvil. (IBM Corporation 2022). Con esta información se llega a la conclusión de que las aplicaciones móviles no están desarrolladas de una manera adecuada, poniendo enriesgo la seguridad de los datos del usuario. Por ello, las pruebas de penetración es una forma que tienen los desarrolladores para proteger la información y evitar el ingreso de intrusos.

La seguridad en aplicaciones móviles es de suma importancia, considerando que se gestionan datos sensibles que pueden ser blanco de actores maliciosos. En el trabajo realizado por (Astély y Ekroth 2022) titulado "Evaluación de seguridad de diez aplicaciones móviles suecas", llevado a cabo en la Universidad KTH Royal Institute of Technology, se adentró en el análisis de posibles vulnerabilidades de diferentes aplicaciones móviles. Durante su investigación, identificaron doce potenciales vulnerabilidades en siete de las diez aplicaciones examinadas. De estas, diez vulnerabilidades fueron seleccionadas para un análisis más detallado y se constató que tres de ellas podían ser explotadas exitosamente. Esta exploración se realizó con el apoyo de la Guía de Pruebas de Seguridad Móvil de OWASP (MSGT), que proporciona un enfoque detallado para llevar a cabo tales pruebas de seguridad.

En el Centro Médico de Especialidades "La Dolorosa", existe una aplicación móvil para los pacientes y la visualización de la prescripción de medicamentos para los mismos. Si no se implementan medidas de seguridad adecuadas en esta aplicación, se pondrían en juego los datos y la salud de los pacientes, ya que contiene información sensible que, en caso de ser intervenida por agentes externos, podría utilizarse para cometer fraudes de identidad u otros delitos más graves.

1.2 Formulación del problema

¿Cuáles son las vulnerabilidades y el nivel de riesgo presentes en la aplicación móvil del centro de especialidades "La Dolorosa" al aplicar pruebas de penetración?

1.3 Sistematización

- ¿Cuáles son los conceptos, características y herramientas para una adecuada aplicación de pruebas de penetración en las aplicaciones móviles?
- ¿Qué vulnerabilidades se pueden identificar en las aplicaciones móviles mediante la aplicación de pruebas de penetración?
- ¿Qué acciones de mejora se pueden aplicar en las aplicaciones móviles a través de la implementación de las pruebas de penetración?
- ¿Cuál es el nivel de mejora en las aplicaciones móviles mediante la implementación de pentesting?

1.4 Justificación

1.4.1 Justificación teórica

Constantemente, las personas usuarias almacenan su información personal en un smartphone, como contraseñas, documentos y archivos, entre otros. Este entorno atrae a personas atacantes que descubren y explotan vulnerabilidades de las aplicaciones, extrayendo información de las personas afectadas para luego hacer un uso indebido de la misma. Por ello, el pentesting es una parte fundamental para evitar este robo de información, ya que permite crear un escenario de análisis y múltiples ataques que podrían realizar ciberdelincuentes a las aplicaciones móviles. Como resultado, se obtiene un reporte de vulnerabilidades encontradas para luego aplicarmedidas de prevención y corrección, logrando así una aplicación más segura y confiable que no ponga en riesgo la información del usuario ni la reputación de la organización.

De acuerdo con (Dehaven 2019, p. 9) el problema más común que se encuentra es el almacenamiento inseguro de datos, que comprende el 76% de todas las vulnerabilidades detectadas. La razón principal de estos problemas de seguridad radica en las debilidades en los mecanismos de seguridad que se incorporan en las aplicaciones durante su creación. Esencialmente, los datos se almacenan de manera insegura en las aplicaciones, lo que crea un punto de entrada para los ciberdelincuentes.

Las aplicaciones mHealth se enfrentan a diversos problemas de seguridad, como la exposición de información personal, robo de identidad, violación de la privacidad del paciente, manipulación de los datos que podrían comprometer la vida, alterar registros médicos como las prescripciones de los medicamentos y poner en riesgo la vida del paciente. (He et al. 2014, p. 645).

Por ello, es muy importante aplicar buenas prácticas y medidas de seguridad sólidas, especialmente para proteger la privacidad, confidencialidad e integridad de los datos de las personas pacientes y garantizar una atención médica segura.

Para saber qué medidas y prácticas de seguridad se deben implementar en las aplicaciones móviles, es necesario aplicar pruebas de penetración. Existen diversas herramientas para la búsqueda de vulnerabilidades, en su mayoría de código abierto, que aportan beneficios como exponer amenazas potenciales que pueden causar pérdida de accesibilidad o tiempo de inactividad, y también ayudan a mantener la credibilidad y confianza de sus grupos de interés. A través de cuatro ejes correspondientes a las pruebas de penetración (análisis estático, análisis dinámico, evaluación de la seguridad del servidor y la inspección de las políticas de seguridad), se evaluaron 154 aplicaciones mHealth y se encontró que el 40% de las aplicaciones presentaban riesgos a la privacidad del usuario. (Knorr y Aspinall 2015, p. 5).

En la Constitución ecuatoriana, el Artículo 66, numeral 19, establece el derecho a la protección de datos de carácter personal, que incluye el acceso y la decisión sobre información y datos de este carácter, así como su correspondiente protección. La recolección, archivo, procesamiento, distribución o difusión de estos datos o información requerirán la autorización del titular o el mandato de la ley. (Asamblea Constituyente 2012, p. 29). Por lo tanto, la implementación de buenas prácticas de seguridad, como la encriptación de los datos y la realización de pruebas de penetración para mejorar las medidas de seguridad y corregir vulnerabilidades, son dos enfoques que ayudan a que las empresas y desarrolladores cumplan con una adecuada protección de datos personales para el usuario.

1.4.2 Justificación aplicativa

Las funcionales realizadas para la aplicación móvil CMED como: dietas, medicamentos, receta médica se lo hicieron mediante la metodología Kanban y en ninguna etapa de la metodología se contempló la seguridad en la aplicación móvil por lo que se considera insegura, es crucial la mejora en la seguridad de la aplicación móvil, para prevenir que ciberdelincuentes accedan a la información de las personas usuarias y eviten fraudes de identidad u otros delitos. Además, este trabajo de titulación curricular busca preservar la salud de las personas pacientes, debido a que, si se llegara a comprometer la información de la aplicación, se pondría en riesgo tanto la salud del paciente como la reputación del centro médico.

El objetivo es analizar las vulnerabilidades en dispositivos móviles y utilizar estos hallazgos como guía para identificar y abordar las vulnerabilidades en la aplicación móvil del centro médico. Para lograr esto, se implementarán herramientas de pentesting que permitan evaluar dichas vulnerabilidades y generar un informe detallado de las mismas. Este informe será crucial para corregir las fallas y mejorar significativamente la seguridad de la aplicación móvil.

Se llevarán a cabo las siguientes fases:

- **Preparación:** Se definirá el alcance de las pruebas de seguridad, incluida la identificación de los controles de seguridad aplicables, los objetivos de prueba de la organización y los datos confidenciales. Se coordinará con el cliente y se obtendrá autorización escrita para realizar las pruebas.
- **Recopilación de inteligencia:** Se analizará el contexto ambiental y arquitectónico de la aplicación para obtener una comprensión contextual general.
- **Mapeo de la aplicación:** Basado en la información de las fases anteriores, se realizará un escaneo automatizado y exploración manual de la aplicación, identificando puntos de entrada, datos contenidos y vulnerabilidades potenciales.
- **Explotación:** Se intentará penetrar en la aplicación explotando las vulnerabilidades identificadas durante la fase de mapeo, determinando si las vulnerabilidades son positivas reales y verdaderas.
- **Informes:** Se elaborará un informe detallado de las vulnerabilidades encontradas, incluyendo el proceso de explotación, la clasificación del tipo de vulnerabilidad y el riesgo si un atacante pudiera comprometer el objetivo.
- **Buenas prácticas:** Basándose en los resultados del informe, se implementarán soluciones para abordar las vulnerabilidades identificadas y se volverán a aplicar las herramientas de pentesting en la nueva versión de la aplicación para garantizar que las correcciones sean efectivas.

El trabajo de Integración Curricular corresponde a la línea transversal de Tecnología de la Información de la Comunicación. Se alinea con el objetivo de “Garantizar la seguridad ciudadana, orden público y gestión de riesgos” y la política diez punto uno “Fortalecer al Estado para mantener confidencialidad, integridad y disponibilidad de la información frente a amenazas provenientes del ciberespacio y proteger su infraestructura crítica” del plan nacional de desarrollo.

1.5 Objetivos

1.5.1 Objetivo general

Analizar las vulnerabilidades en la aplicación móvil del centro de especialidades “La Dolorosa” utilizando pentesting.

1.5.2 Objetivos específicos

- Estudiar los conceptos, características y herramientas de pentesting en aplicaciones móviles para seleccionar la(s) herramienta(s) más adecuada(s).
- Analizar mediante pentesting las vulnerabilidades en la aplicación móvil del Centro de Especialidades “La Dolorosa”.
- Aplicar acciones de mejora en la aplicación móvil en función del análisis de vulnerabilidades.
- Evaluar las vulnerabilidades en la aplicación móvil.

1.6 Alcance

Este documento define el proyecto, especificando los detalles relacionados con el objetivo, necesidades asociadas, productos a entregar, eventos críticos, requerimientos técnicos, límites y exclusiones, así como la revisión por parte del cliente.

- Objetivo del Proyecto

Realizar un análisis exhaustivo de seguridad, mediante pentesting y la implementación de buenas prácticas de seguridad, a la aplicación móvil del Centro Médico de Especialidades La Dolorosa, con el fin de reducir la cantidad de vulnerabilidades existentes y garantizar la protección de los datos de los usuarios.

- Necesidades Asociadas
    - Identificar y analizar las vulnerabilidades de la aplicación móvil del Centro de Especialidades la Dolorosa.
    - Implementar buenas prácticas de seguridad a la de la aplicación móvil del Centro de Especialidades la Dolorosa.
    - Comparar el número de vulnerabilidades antes y después del manteamiento de la aplicación.

- Productos para Entregar
    - La aplicación móvil con mejoras en su seguridad, producto de las medidas implementadas.
- Momentos Importantes
    - Selección de herramientas adecuada para el análisis de vulnerabilidades.
    - Implementación de buenas prácticas de seguridad en la aplicación móvil, siguiendo los lineamientos de la metodología de seguridad de aplicaciones móviles OWASP.
    - Comparación del estado de seguridad de la aplicación móvil antes y después de la implementación de las buenas prácticas.
    - Entrega del producto.
- Requerimientos Técnicos
    - Uso de herramientas de pentesting específicas para aplicaciones móviles.
    - Aplicación de la metodología OWASP para guiar las pruebas de seguridad y las implementaciones de seguridad.
- Límites y Exclusiones
    - Los procesos para garantizar la Confidencialidad, Integridad y Disponibilidad deben ser implementados, sin embargo, no se desplegará la aplicación en un servidor de la nube, pues estos valores no se contemplan dentro de los costos del proyecto.
- Revisión del Cliente

El desarrollador encargado de realizar la aplicación móvil para el Centro Médico de Especialidades “La Dolorosa” revisará y aprobará las mejoras implementadas en la aplicación móvil, garantizando la implementación de requerimientos de seguridad.

CAPÍTULO II

2 FUNDAMENTOS TEÓRICOS

En este capítulo se describen los conceptos referentes al Pentesting en aplicaciones móviles, herramientas, vulnerabilidades, conceptos técnicos, aspectos sobre la seguridad y conceptos relacionados con la seguridad informática, además de establecer el cómo se medirá la seguridad de la aplicación.

2.1 Aplicación móvil CMED

La aplicación móvil CMED está diseñada específicamente para la gestión y el control de prescripciones médicas, dietas y toma de medicamentos. Proporciona un acceso eficaz y constante a la información de sus pacientes, lo que garantiza que estén bien informados sobresus tratamientos y les permite cuidar su salud de manera más eficaz. En su desarrollo, se utilizó la metodología Kanban y el Framework Flutter para diseñar interfaces de usuario. Además, en términos de gestión de datos, se adoptó ORM Sequelize en combinación con PostgreSQL. A su vez, Charles Web Debugging Proxy desempeñó un papel crucial en el proceso de prueba y análisis, permitiendo la identificación y resolución de problemas durante la fase de desarrollo. (Riera 2023, p. 19).

2.2 Flutter

Flutter, desarrollado por Google, es un framework contemporáneo que ha ganado prominencia en el desarrollo de aplicaciones para dispositivos Android. Esta herramienta se destaca por permitir una compilación casi completa de código en aplicaciones nativas, ofreciendo ventajas significativas en cuanto a la reducción de costos en diversas etapas del desarrollo. Aunque los frameworks tradicionalmente más populares se centran en el desarrollo web, como React de Facebook y Angular de Google, Flutter se presenta como una solución innovadora en el panorama de desarrollo móvil, atendiendo a las crecientes demandas tecnológicas y diversidad de dispositivos y sistemas operativos. (Quisaguano et al. 2022).

2.2.1 *Method Channel*

El Method Channel es una característica esencial para la comunicación entre el código Dart de Flutter y el código nativo de la plataforma. En la **Ilustración 1-2**, se presenta la funcionalidad y el papel fundamental que juega el Method Channel en la arquitectura de Flutter.

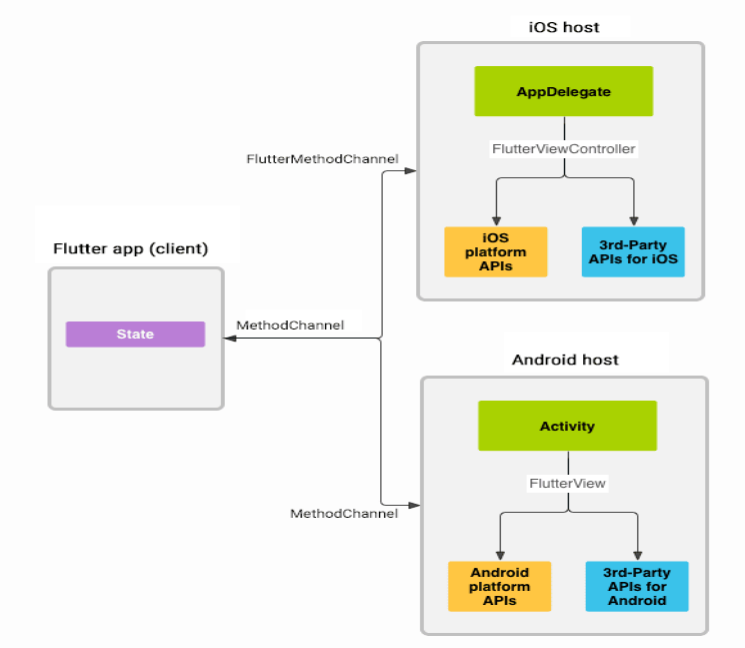


Ilustración 1-2: Method Channel flutter
Fuente: (Flutter dev 2019b)

Los intercambios de mensajes y respuestas se realizan de manera asíncrona para mantener la fluidez de la interfaz de usuario y al utilizar la API MethodChannel en el lado cliente, es posible enviar mensajes que se corresponden con las llamadas a los métodos; en el lado de la plataforma, tanto la API MethodChannel en Android como la API FlutterMethodChannel en iOS permiten recibir estas llamadas y proporcionar un resultado. Gracias a estas clases, se facilita la creación de un plugin de plataforma con un mínimo de código preestablecido y, en relación con Flutter, estas APIs convierten el código de Flutter en funciones operativas para Android y para iOS. (Flutter dev 2019a).

2.3 Tipos de Aplicaciones móviles

2.3.1 Aplicaciones nativas

Las aplicaciones nativas están diseñadas específicamente para operar en una plataforma determinada y requieren una consideración meticulosa del dispositivo, el sistema operativo y su versión. Su principal ventaja es la posibilidad de interactuar con todos los componentes del dispositivo. Este tipo de aplicaciones transforma el código fuente en código ejecutable, un proceso similar al empleado en las aplicaciones de escritorio tradicionales. Para su distribución, se someten a las tiendas de aplicaciones particulares de cada sistema operativo, que realizan una

revisión exhaustiva para asegurarse de que la aplicación cumple con los estándares de la plataforma antes de ponerla a disposición de los usuarios. (Delía et al. 2013, p. 7).

Flutter es un marco de interfaz de usuario que crea aplicaciones nativas y reactivas para iOS y Android mediante el uso del método MethodChannel. Esto se traduce en la creación de aplicaciones móviles nativas a partir de un único código base. Esta característica, en conjunto con el uso del lenguaje de programación Dart, facilita el desarrollo de aplicaciones móviles con un diseño visualmente atractivo. Además, Flutter incorpora la función "hot reload", que agilizael proceso de desarrollo al permitir visualizar en tiempo real los cambios realizados en la aplicación. (Napoli 2019, p. 5). Por consiguiente, y de acuerdo con las definiciones, las aplicacionesnativas son aquellas que ejecutan un código fuente propio del sistema. Flutter, al tener el método MethodChannel, transforma el código en las extensiones propias del sistema Android y iOS.

2.3.2 Aplicaciones Hibridas

Las aplicaciones híbridas constituyen una categoría de desarrollo multiplataforma que, a diferencia de otras, no se ejecutan en un navegador sino en un contenedor web especializado. Este contenedor proporciona un acceso más amplio a las funcionalidades del dispositivo a travésde una API específica. Aunque permiten la reutilización de código en distintas plataformas y facilitan la distribución a través de las tiendas de aplicaciones, pueden tener limitaciones en la interfaz y la velocidad de ejecución. Por su parte, las aplicaciones interpretadas ofrecen una variante en el desarrollo multiplataforma, donde gran parte del código se traduce a código nativo, pero una porción se interpreta al momento de ejecutarse. (Thomas et al. 2018, p. 589). Si bien Flutter opera bajo la premisa de las aplicaciones híbridas, su motor, escrito en C++, le permite interactuar directamente con el SDK de cada sistema operativo. Esto se traduce en que Flutterno sólo compila como aplicación híbrida, sino que puede alcanzar una ejecución similar a la de una aplicación nativa. (Flutter dev 2019b).

2.4 Seguridad Informática

La seguridad informática se refiere a la protección de los sistemas informáticos como puede ser ataques de fuerza bruta, inyecciones SQL XXS entre otros. Se usa metodologías como OWASP para organizar y llevar a cabo procesos de seguridad debido a que permite identificar y reducir los riesgos más relevantes que presentan los sistemas. (Castillo Enríquez, Hidalgo Guijarro y Guano Cárdenas 2022, p. 180).

Según (Romero Castro et al. 2018, pp. 14-15) la seguridad se puede clasificar en tres partes:

- **Usuarios:** Los usuarios representan el punto más vulnerable en cualquier sistema de seguridad, un fallo, olvido o percance puede desencadenar un desastre que anule la seguridad. (Romero Castro et al. 2018, p. 14).
- **Información:** Es un activo de valor que son susceptibles de ser explotados o dañados por diversas amenazas, como la adulteración de datos o las interrupciones del servicio. La anticipación a los posibles impactos y la planificación de respuestas son esenciales en la evaluación de seguridad. (Martorell y Huacarán 2013, p. 12).
- **Infraestructura:** Incluye hardware, servidores, estaciones de trabajo, software,sistemas operativos, base de datos. Si no se controlan las amenazas existe el riesgo de robo de identidad, acceso no autorizado entre otros. (Romero Castro et al. 2018, p. 15).

El pentesting o conocido como pruebas de penetración es un componente de la seguridad informática que implica la identificación de vulnerabilidades en un sistema para evaluar su nivel de seguridad y establecer estrategias de seguridad. Se lo puede realizar mediante uso de escáneres de vulnerabilidades que detectan debilidades en un sistema, red. (Chirou 2023, p. 73).

2.4.1 Seguridad en aplicaciones móviles

A diferencia de los sistemas y aplicaciones de escritorio, los sistemas operativos y aplicaciones móviles poseen características únicas, haciendo que vulnerabilidades típicas como los desbordamientos de búfer y problemas de XSS sean menos prevalentes en este ámbito. La seguridad en el entorno móvil se prioriza en la protección de datos, considerando la vasta cantidad de información personal y sensible que las aplicaciones almacenan. Es esencial centrar el enfoque de la seguridad móvil en el manejo, almacenamiento y protección de estos datos, prestando especial atención a aspectos como el almacenamiento de datos, la intercomunicación entre aplicaciones, el uso adecuado de las API criptográficas y la garantía de comunicaciones dered seguras. (Mueller et al. 2023, p. 26).

2.4.1.1 Vulnerabilidades en aplicaciones móviles mHealth

Las aplicaciones móviles mHealth son cada vez más populares debido a la facilidad de acceso y la comodidad que ofrecen en la gestión de la atención médica. Sin embargo, con este aumento en la adopción de aplicaciones mHealth también vienen desafíos relacionados con la seguridad de la información. Es crucial abordar las vulnerabilidades de seguridad en las aplicaciones

móviles mHealth para proteger la confidencialidad, integridad y disponibilidad de los datos sensibles del paciente. (Antonishyn 2020, pp. 49-57).

Las aplicaciones mHealth pueden enfrentar diversos problemas de seguridad, incluyendo la exposición de información personal, robo de identidad, violación de la privacidad del paciente y manipulación de datos que podrían tener consecuencias potencialmente mortales. Los riesgos incluyen la alteración de registros médicos, como las prescripciones de medicamentos, lo que podría poner en peligro la vida del paciente. (He et al. 2014, p. 646).

2.4.2 Pruebas de Penetración

Las pruebas de penetración ayudan a la verificación de la seguridad del software que se aplica mayoritariamente en la fase final del ciclo de vida del software, aplicar estas pruebas en el estado final del software presentan problemas de tiempo y presupuesto para la aplicación de soluciones. Una prueba de penetración solo puede identificar una pequeña muestra de todos los riesgos que pueden existir en un sistema, para tener una mayor efectividad se debe rastrear los problemas desde los requerimientos. (Arkin, Stender y Mcgraw 2005, pp. 84-86).

Como bien se dice en la cita anterior, el Pentesting se lo debe aplicar desde un inicio y para los sistemas ya desarrollados, se conoce que una prueba de penetración fue exitosa cuando se obtiene la capacidad de hacer algo distinto a lo que la organización esperara que hiciera y como nos lo confirma Bishop, el objetivo es verificar si existe al menos una manera de ingresar al sistema o el de conocer cuan fácil sería leer un archivo confidencial, apropiarse de identidades, cambiar datos o reconfigurar el sistema para observar su comportamiento. (Bishop 2007, p. 7).

2.4.3 Tipos de pruebas de penetración

2.4.3.1 Caja negra

Se replica lo más exactamente posible ataques remotos sin tener tanta información del sistema a atacar, se simula el comportamiento de los ciberdelincuentes ya que se ataca al sistema sin conocer la red de la organización, medidas implantadas. (Acosta Santana 2022, p. 3).

2.4.3.2 Caja Blanca

Los auditores tienen acceso a la mayoría de información sobre la aplicación a la que se la está probando como puede ser: resúmenes, diagramas, código, acceso al encargado de desarrollo. Estas pruebas revelan vulnerabilidades que pueden no ser tan claras y producen un resultado

más rápido debido a que el auditor no tiene por qué estar buscando fugas de información. (Geer y Harthorne 2002, pp. 7-8).

2.4.3.3 Caja gris

El auditor tiene cierta información sobre el sistema, puede ser el funcionamiento o el tipo de arquitectura, impidiendo el acceso al código fuente permitiendo tener una evaluación parcial de la seguridad y la identificación de algunas vulnerabilidades importantes. (Figueiredo et al. 2020, p. 6).

2.4.4 Análisis de Vulnerabilidades

El análisis de vulnerabilidades es generalmente el proceso de búsqueda de vulnerabilidades en una aplicación puede hacerse manualmente y utilizando escáneres automáticos para identificar las principales vulnerabilidades existen dos tipos de análisis de vulnerabilidades que son el análisis estático y dinámico. (Mueller et al. 2023. p. 29).

2.4.4.1 Análisis Estático

Este proceso se caracteriza por su capacidad de identificar posibles brechas de seguridad sin la necesidad de ejecutar el programa, su principal ventaja radica en su capacidad de escrutinio del código fuente mismo, permitiendo descubrir fallos de seguridad potenciales antes de su implementación, se lo puede realizar mediante escáneres automáticos o una revisión manual de código. (Wilson 2019, p. 88).

2.4.4.2 Análisis Dinámico

Este tipo de análisis tiene un enfoque distinto. Consiste en interactuar directamente con la interfaz de usuario de una aplicación web para hallar potenciales puntos débiles en su seguridad.Mediante este enfoque, se pueden identificar problemas que podrían no ser evidentes en el código fuente pero que pueden surgir durante la operación de la aplicación, dando un énfasis particular en los aspectos de arquitectura de la aplicación que puedan presentar debilidades. (Wilson 2019, p. 89).

2.4.5 Herramientas pentesting

2.4.5.1 Mobile Security Framework

Mobile Security Framework (MobSF) representa una de las herramientas esenciales en elarsenal de un evaluador de seguridad para aplicaciones móviles. Su capacidad para procesar y analizar archivos binarios, así como código fuente, lo hace altamente versátil y apto para diversas plataformas móviles, incluyendo Android, iOS y Windows. Con un enfoque hacia el pentesting automatizado, MobSF no solo agiliza el proceso de identificación de vulnerabilidades, sino que ofrece una visión detallada de cada hallazgo, lo cual es crucial para desarrolladores y especialistas en seguridad. Uno de los atributos más notables de MobSF es su capacidad para realizar análisis estáticos, dinámicos y de malware, expandiendo aún más su alcance de detección. Además, su compatibilidad con el fuzzing de API web refuerza la importancia de este framework en el mundo actual de aplicaciones interconectadas. (Nurindahsariy Zen 2021, pp. 65-66).

Al ser una herramienta de código abierto, MobSF es ampliamente utilizado en la comunidad, lo que refuerza su fiabilidad y capacidad para detectar vulnerabilidades relacionadas con fugas de información y problemas de autenticación. Sin embargo, como cualquier herramienta, no está exento de limitaciones. Es posible que ocasionalmente reportes falsos positivos, en especial debido a código muerto presente en las aplicaciones, lo que puede llevar a complicaciones en el proceso de mantenimiento y abrir la puerta a posibles errores de seguridad. (Chen et al. 2018, p. 800).

2.4.5.2 Androbugs

AndroBugs es una herramienta especializada en el análisis de seguridad que se dedica a explorar vulnerabilidades en aplicaciones móviles. A pesar de su capacidad para identificar fallos, en particular aquellos vinculados a fugas de datos y autenticación deficiente, enfrenta desafíos cuando se le confronta con soluciones tecnológicas más avanzadas, como MobSF. Esta última se destaca por proporcionar descripciones minuciosas y prácticas de las vulnerabilidades detectadas, a diferencia de AndroBugs, cuyos informes pueden no ser tan precisos o útiles en la toma de acciones correctivas. (Chen et al. 2018, p. 800).

2.4.5.3 BurpSuite

BurpSuite es una herramienta con una interfaz gráfica para realizar pruebas, actúa como un proxy interceptador, indicia y modifica el tráfico HTTP de manera estructurada. Representa una visión general rápida del sistema a evaluar, mensajes transmitidos y parámetros, permite la ejecución de diferentes escenarios de ataque de forma manual. Para facilitar escenarios más complejos, BurpSuite ofrece puntos de extensión que permiten escribir funciones personalizadas, permitiendo a las extensiones monitorear y analizar cualquier mensaje HTTP que pase por su proxy, modificarlos y crear nuevos elementos de interfaz de usuario para mostrarlo. (Mainka et al. 2015, pp. 124-126).

2.4.5.4 Zed Attack Proxy

Zed Attack Proxy (ZAP) es un escáner de seguridad para aplicaciones web de código abierto que ha obtenido reconocimientos por su excelencia. Entre sus características más destacadas se encuentran un Proxy de interceptación, que permite observar y modificar el tráfico entre el navegador y el servidor; escáneres activos y pasivos para detectar vulnerabilidades; posee herramientas para ataques de fuerza bruta y fuzzing, con el propósito de identificar vulnerabilidades potenciales, la extensibilidad de OWASP ZAP permite a los usuarios personalizar y agregar nuevas funcionalidades según sus necesidades. (Ashari et al. 2023, p.25).

2.4.5.5 SQLMap

SQLMap, una herramienta de código abierto se especializa en la identificación y explotación de vulnerabilidades de inyección SQL en aplicaciones web, diseñada para descubrir fallos de seguridad que, al ser explotados, brindan acceso no autorizado a datos esenciales. Tras detectar estas vulnerabilidades, facilita la extracción de información relevante desde la base de datos como sesiones activas, nombres de usuario y detalles de la estructura del sistema de almacenamiento de datos. Lo que la diferencia de otras herramientas es su capacidad para emplear técnicas avanzadas, desde la inyección SQL ciega inferencial, anexando declaraciones SQL a parámetros en solicitudes HTTP, hasta técnicas como la inyección a través de la consulta UNION y la gestión de consultas agrupadas, determinando la posibilidad de múltiples sentencias SQL delimitadas. Sin embargo, su verdadera distinción radica en su enfoque de fuerza bruta después de identificar una vulnerabilidad. (Ciampa, Visaggio, y Di Penta 2010, pp. 44-46; Ojagbule, Wimmer y Haddad 2018, pp. 1-7).

2.4.5.6 Havij

Diseñada para asistir a profesionales en pruebas de penetración, Havij es una herramienta especializada en la automatización de inyecciones SQL. Su propósito principal es identificar y explotar vulnerabilidades de inyección SQL. Havij ofrece tanto versiones gratuitas como comerciales, destacando la versión 1.16 Pro como un recurso de código abierto. Esta herramienta es versátil y compatible con diversas bases de datos, tales como MsSQL, MySQL, Oracle, PostgreSQL y MsAccess. No obstante, presenta ciertos desafíos, entre ellos la tarea de reconocer sitios vulnerables y algunas limitaciones de compatibilidad. (Nagpal et al. 2015, p. 747).

2.4.6 Tabla comparativa de las herramientas

En el amplio espectro de la ciberseguridad, existen diversas herramientas especializadas que se adaptan a diferentes tipos de análisis. Es esencial elegir las herramientas adecuadas que correspondan específicamente a las necesidades y objetivos. Por ello, se ha categorizado las herramientas en función de los dos principales enfoques de análisis de vulnerabilidades en aplicaciones móviles: el análisis estático y el análisis dinámico.

La estructura para la comparación de estas herramientas se inspira en los criterios propuestos por Gamido Heidilyn y Gamido Marlon (2019, p. 4475) como se indica en la **Tabla 1-2** se ilustra estos parámetros, adaptándolos específicamente para el contexto de herramientas de prueba de software en aplicaciones móviles.

Tabla 1-2: Parámetros para la comparación de las herramientas de pruebas de software

Criterios utilizados	**Definición**
Plataformas cruzadas	Sistema operativo compatible
Facilidad de aprendizaje	Qué fácil se usa la herramienta
Habilidades de programación	Habilidades de programación necesarias
Generación de informes	Cómo se representa el resultado
Coste	Gratis o Licencia
Otros	Ventaja / Desventaja / Comentario

Fuente: (Gamido Heidilyn y Gamido Marlon 2019, p. 4475).

Realizado por: Mesias Francisco, 2023.

Utilizando la guía proporcionada por estos parámetros, se realizará una evaluación de las herramientas, clasificándolas según el tipo de funcionalidad. Esta comparativa es fundamental

para determinar qué herramientas son más adecuadas para cada fase del pentesting y para garantizar una evaluación exhaustiva de las vulnerabilidades en la aplicación móvil CMED

2.4.6.1 Tabla de comparación de las herramientas para el análisis estático

Dos de las herramientas más prominentes en el campo de análisis estático son Mobile Security Framework y Androbugs. A continuación, en la **Tabla 2-2** se presenta una comparativa detallada de ambas herramientas.

Tabla 2-2: Comparación de la herramienta MobSF con la herramienta Androbugs

Parámetros	**MobSF**	**Androbugs**
Plataformas cruzadas	Android/iOS/Windows. (Mobile Security Framework 2021).	Android. (Lin Yu-Cheng 2023).
Facilidad de aprendizaje	Es un marco automatizado. (Mobile Security Framework 2021).	Se usa mediante comandos, no tiene interfaz GUI. (Lin Yu-Cheng 2023).
Habilidades de programación	No necesariamente.	Se debe tener conocimientos en comandos mediante terminal para poder utilizar la herramienta.
Generación de informes	MobSF genera un informe de los resultados obtenidos de la aplicación a analizar. (Gómez Zúñiga y Suquilanda Rodríguez 2022, p. 10).	La herramienta entrega un reporte en formato txt. (Argudo Murillo 2019, p. 55).
Coste	Bajo la licencia GNU GPL v3.0 (Codigo abierto). (Mobile Security Framework 2021).	Bajo la licencia GNU GPL v3.0 (Codigo abierto). (Lin Yu-Cheng 2023).
Otros	Procesa archivos binarios y código fuente. Análisis estático, dinámico y de malware. Compatible con fuzzing de API web. Puede reportar falsos positivos. Se recomienda usarlo con otras herramientas. (Nurindahsari y Zen2021, pp. 65-66; Chen et al. 2018, p. 800).	Enfoque en detección de fugas de datos y autenticación. Informes menos detallados que MobSF. (Chen et al. 2018, p. 800).

Realizado por: Mesias Francisco, 2023.

2.4.6.2 Tabla de comparación de las herramientas para el análisis dinámico

Para el análisis de las herramientas para el análisis dinámico se procedió a separar el análisis de acuerdo con la arquitectura de la aplicación de la móvil donde la primera es la comunicación del api y la segunda la base de datos.

2.4.6.2.1 Tabla de comparación entre Burp Suite y Zed Attack Proxy

A continuación, en la **Tabla 3-2** se presenta una comparativa detallada de las herramientas BurpSuite con Zed Attack Proxy que son herramientas proxy para la intercepción de datos a través de la red.

Tabla 3-2: Comparación de la herramienta BurpSuite con la herramienta Zed Attack Proxy

Parámetros	**BurpSuite**	**Zed Attack Proxy**
Facilidad de aprendizaje	Realiza análisis y optimización de ataques. (Bolivar et al. 2021, p. 419).	Proporciona escáneres automáticos. (Zed Attack Proxy 2023).
Habilidades de programación	No necesariamente.	No necesariamente.
Coste	Licenciada, no obstante, la mayoría de las extensiones son gratuitas. (Bolivar et al. 2021, p. 419).	Licencia de código abierto. (Zed Attack Proxy 2023).
Otros	Actúa como un proxy interceptador y permite la creación de funciones personalizadas, ofreciendo puntos de extensión para monitorear y analizar mensajes HTTP. (Mainka et al. 2015, pp. 124-126).	Ofrece un Proxy de interceptación, escáneres activos y pasivos, herramientas para ataques de fuerza bruta y fuzzing, además de la posibilidad de extender funcionalidades según las necesidades del usuario. (Ashari et al. 2023, p.25)

Realizado por: Mesias Francisco, 2023.

2.4.6.2.2 Comparación entre SQLMap y Havij

A continuación, en la **Tabla 4-2** se presenta una comparativa detallada de las herramientas SQLMap con Havij para la seguridad mediante inyección SQL.

Tabla 4-2: Comparación de la herramienta SQLMap con la herramienta Havij

Parámetros	**SQLMap**	**Havij**
Plataformas cruzadas	Gran mayoría de gestión de base datos entre los que se encuentra PostgreSQL. (Damele, Bernardo y Stampar 2023).	Compatible con diversas bases de datos como MsSQL, MySQL, Oracle, PostgreSQL y MsAccess. (Nagpal et al. 2015, p. 747).
Facilidad de	Reconocimiento automático. (Damele, Bernardo y Stampar 2023).	Herramienta automatizada. (Md Zaur Reza 2023).

aprendizaje		
Habilidades de programación	Manejo de comandos.	Interfaz GUI
Coste	Código abierto. (Damele, Bernardo y Stampar 2023).	Bajo la licencia GNU GPL v3.0 Código abierto. (Md Zaur Reza 2023).
Otros	Destaca por su variedad de técnicas avanzadas de explotación, como la inyección SQL ciega inferencial, lainyección SQL a través de la consulta UNION y el manejo de consultas agrupadas. Adopta un enfoque de fuerza bruta tras identificar una vulnerabilidad (Ciampa, Visaggio, y Di Penta 2010, pp. 44-46; Ojagbule, Wimmer y Haddad 2018, pp. 1-7).	Diseñada para profesionales en pruebas de penetración, ofrece versiones gratuitas y comerciales. Presenta desafíos en el reconocimiento de sitios vulnerables y algunas limitaciones de compatibilidad (Nagpal et al. 2015, p. 747).

Realizado por: Mesias Francisco, 2023.

2.5 OWASP Mobile Application Security

OWASP Mobile Application Security (MAS) proporciona un estándar de seguridad para aplicaciones móviles (MASVS) y una guía de pruebas exhaustiva (MASTG), la última guía cubre los procesos, técnicas y herramientas utilizadas durante una prueba de seguridad de aplicaciones móviles, junto con un conjunto de pruebas que permiten a los evaluadores entregar resultados consistentes y completos. (Mueller et al. 2023, p. 11).

2.5.1 Estándar de verificación de seguridad de aplicaciones móviles

El Estándar de Verificación de Seguridad de Aplicaciones Móviles (MASVS) sirve como un pilar fundamental para establecer criterios claros y concisos relacionados con la seguridad de aplicaciones móviles. Abarca problemas clave que van desde el almacenamiento de datos y la criptografía hasta la interacción con la plataforma móvil. Sin embargo, es esencial entender que el MASVS no es un conjunto rígido de reglas, sino más bien una guía que puede ser adaptada según las necesidades específicas de un proyecto. (Schleier et al. 2023, pp. 7-11). Esta flexibilidad permite que las organizaciones y desarrolladores ajusten el estándar a su contexto particular, priorizando las áreas de seguridad más relevantes para su aplicación.

Dentro de este marco de adaptabilidad, el MASVS puede también ser utilizado como una herramienta de corrección de vulnerabilidades. Cuando se identifica un problema en una aplicación, las directrices proporcionadas por el MASVS pueden ser empleadas para entender la naturaleza de la vulnerabilidad y determinar las mejores prácticas para mitigar o resolver el problema.

2.5.1.1 MASVS-ALMACENAMIENTO

La salvaguarda de información delicada, tales como las credenciales de los usuarios e información privada, es fundamental en la seguridad móvil. Si una aplicación utiliza inadecuadamente APIs del sistema operativo, como el almacenamiento local o la comunicación entre procesos (IPC), podría exponer datos sensibles a otras aplicaciones en el mismo dispositivo. (Schleier et al. 2023, p. 17).

2.5.1.2 MASVS-CRIPTOGRAFÍA

Es vital garantizar que la aplicación utilice la criptografía de acuerdo con las mejores prácticas de la industria, incluyendo el uso de bibliotecas criptográficas probadas, una elección y configuración adecuada de primitivas criptográficas y un generador de números aleatorios adecuado cuando se requiera aleatoriedad. (Schleier et al. 2023, p. 19).

2.5.1.3 MASVS-AUTENTICACIÓN Y AUTORIZACIÓN

Normalmente, dirigir a los usuarios para que inicien sesión en un servicio remoto es una parte integral de la arquitectura general de la aplicación móvil. A pesar de que la mayoría de la lógica de autenticación y autorización se lleva a cabo en el extremo, también existen algunos desafíos de implementación en el lado de la aplicación móvil. (Schleier et al. 2023, p. 21).

2.5.1.4 MASVS-RED DE COMUNICACIÓN

Los dispositivos móviles se conectan regularmente a una variedad de redes, incluyendo redes Wi-Fi públicas compartidas con otros clientes (potencialmente maliciosos). Esto crea oportunidades para una amplia variedad de ataques basados en la red, desde simples a complicados. (Schleier et al. 2023, p. 23).

2.5.1.5 MASVS-PLATAFORMA

Todos los sistemas operativos móviles implementan sistemas de permisos de aplicaciones que regulan el acceso a APIs específicas. También ofrecen instalaciones de comunicación entre procesos IPC más ricas en Android o menos ricas en iOS que permiten a las aplicaciones intercambiar señales y datos. (Schleier et al. 2023, p. 25).

2.5.1.6 MASVS-CÓDIGO

Las aplicaciones móviles raramente se encuentran con problemas tradicionales de inyección o de gestión de memoria debido a su superficie de ataque más reducida. Las características de seguridad gratuitas que ofrecen los compiladores y los SDK móviles ayudan a aumentar la seguridad y a mitigar los ataques. (Schleier et al. 2023, p. 27).

2.5.1.7 MASVS-RESILENCIA

En el mundo de las apps móviles, los controles de protección de software se utilizan ampliamente, por lo que los evaluadores de seguridad necesitan formas de lidiar con estas protecciones. Creemos que las protecciones del lado del cliente pueden ser beneficiosas si se emplean con un propósito claro y expectativas realistas, y no se utilizan para reemplazar los controles de seguridad. (Schleier et al. 2023, p. 30).

2.5.2 Guía de pruebas de seguridad de aplicaciones móviles

La Guía de Pruebas de Seguridad de Aplicaciones Móviles (MASTG) es un manual integral que cubre los procesos, técnicas y herramientas utilizadas durante el análisis de seguridad de aplicaciones móviles. Puede ser utilizada en las etapas de planificación y diseño de la arquitectura de una aplicación, y puede servir como una línea de base para pruebas de seguridadmanual o como una plantilla para pruebas de seguridad automatizadas durante o después del desarrollo. (Mueller et al. 2023, p. 24).

Esta guía nos ofrece las siguientes fases para realizar pruebas de penetración en aplicaciones móviles:

2.5.2.1 Preparación

Se define el alcance de las pruebas de seguridad, se sincroniza con el cliente, así también como la protección legal. (Mueller et al. 2023, p. 31).

2.5.2.1.1 Preparación con el cliente

Se deben coordinar las variantes de la aplicación a probar y solicitar al equipo de desarrollo una versión de la aplicación. Se debe definir en conjunto con el cliente el alcance de las pruebas de las pruebas formalizando un acuerdo legal entre el pentester entre el cliente, usualmente en forma de contrato respetando las leyes que prohíben las pruebas de penetración no autorizadasen muchas jurisdicciones. (Mueller et al. 2023, p. 32).

2.5.2.1.2 Identificación de datos confidenciales

Los datos confidenciales se clasifican basándose en su valor e importancia, y estos pueden estar en tres estados: en reposo (archivados), en uso (cargados en una aplicación) y en tránsito (intercambiados entre aplicaciones). El nivel de revisión de cada estado depende de la relevancia de los datos y su probabilidad de acceso. (Mueller et al. 2023, p. 32).

2.5.2.2 Recopilación de inteligencia

Se enfoca en obtener una comprensión contextual de la aplicación, recogiendo información sobre su arquitectura, su propósito comercial y el contexto en que opera. Esta información puede categorizarse como ambiental o arquitectónica. (Mueller et al. 2023, p. 33).

2.5.2.2.1 Información ambiental

Incluye los objetivos de la organización para la aplicación, la industria relevante, las partes interesadas y los inversores, y los procesos internos, flujos de trabajo y estructuras organizativas. Esta información ofrece una visión sobre cómo interactúan los usuarios con la aplicación, posibles superficies de ataque y posibles vulnerabilidades en la lógica empresarial. (Mueller et al. 2023, p. 33).

2.5.2.2.2 Información arquitectónica

Conocer cómo la aplicación accede y gestiona los datos, cómo se comunica con otros recursos, cómo administra las sesiones de usuario, cómo reacciona si detecta que se ejecuta en dispositivos con jailbreak o rooteados, los sistemas operativos y sus versiones en los que se ejecuta la aplicación, la aplicación de protocolos de transporte seguros, el uso de claves seguras y algoritmos criptográficos para asegurar el cifrado del tráfico de red, y los servicios remotos que la aplicación consume que podrían comprometer al cliente si se ven comprometidos. (Muelleret al. 2023, p. 33).

2.5.2.3 Asignación de la aplicación

El modelado de amenazas es un recurso relevante. los documentos resultantes de estas sesiones suelen ser de gran ayuda para identificar información necesaria para el tester de seguridad (puntos de entrada, activos, vulnerabilidades, gravedad, etc.). Por tanto, se recomienda a los probadores discutir con el cliente la disponibilidad de estos documentos. (Mueller et al. 2023, p. 33).

2.5.2.4 Explotación

En esta etapa, se identifican y explotan las vulnerabilidades de la aplicación. Se emplean herramientas de pruebas de penetración y análisis de seguridad para el análisis estático y dinámico, depende del enfoque se permite una evaluación completa, segura y precisa de la estructura y funcionalidad de la aplicación. (Mueller et al. 2023, pp. 33-34).

2.5.2.5 Informes

El probador de seguridad informa de las vulnerabilidades, incluyendo el proceso de explotación en detalle, clasifica el tipo de vulnerabilidad, documenta el riesgo si un atacante pudiera comprometer el objetivo y describe a qué datos ha podido acceder el probador de formailegítima. Esencial para el cliente. (Mueller et al. 2023, p. 34).

2.5.3 Buenas Practicas

Según la Open Web Application Security Project (OWASP) se refieren a un conjunto de recomendaciones, directrices y procedimientos estandarizados que se han identificado como métodos efectivos y eficientes para asegurar aplicaciones web y móviles. Estas prácticas se derivan de la amplia experiencia y conocimientos de la comunidad de seguridad, y tienen como objetivo prevenir, detectar y remediar vulnerabilidades y amenazas en el ámbito de las aplicaciones. (The OWASP Foundation, 2020, p. 12).

2.6 Evaluación de Amenazas enfoque DREAD

El método DREAD, recomendado por OWASP que fue establecido por (Microsoft 2023a) Microsoft y se diseñó con el propósito principal de evaluar y clasificar amenazas según su nivel de riesgo. Su aplicabilidad se enfoca en estimar la probabilidad de explotación de una vulnerabilidad desde diferentes perspectivas. El método se descompone en cinco atributos de riesgo:

- **Daño (D):** Mide el impacto en los activos, es decir, cuán afectados pueden estar estos.
- **Reproducibilidad (R):** Estima la facilidad con la que un ataque puede ser reproducido.
- **Explotabilidad (E):** Determina la sencillez con la que puede lanzarse un ataque.
- **Usuarios afectados (A):** Se refiere al número de usuarios que podrían resultar afectados por un ataque.
- **Descubrimiento** (D): Evalúa la facilidad con la que se puede descubrir una vulnerabilidad.

Sin embargo, el modelo DREAD no está exento de críticas. Una de las principales limitaciones señaladas es la vaguedad en la descripción de cada parámetro de riesgo, lo que puede aumentar la subjetividad en su aplicación. (Zhang et al. 2022). En respuesta a esta ambigüedad, han surgido adaptaciones con el objetivo de simplificar y concretar este modelo.

Una de estas adaptaciones propone un sistema de puntuación concreto. Según Olivares y Eduardo (2017, p. 9), las puntuaciones se dividen en: alto riesgo (12-15), riesgo medio (8-11) y riesgo bajo (5-7). Esta adaptación se visualiza en la siguiente **Tabla 5-2**.

Tabla 5-2: Tabla de puntuación DREAD

Puntuación	**Alto (3)**	**Medio (2)**	**Bajo (1)**
Daño	El atacante podría Ejecutar aplicaciones con permiso de administrador	Divulgación de información sensible	Divulgación de información trivial.
Reproducibilidad	El ataque es fácilmente reproducible.	El ataque se podría reproducir, pero únicamente en condiciones muy concretas.	Ataque difícil de reproducir.
Explotabilidad	Un programador novato podría implementar el ataque en poco tiempo	Un programador experimentado podría implementar el ataque	Se requieren cierta habilidad y conocimiento
Todos los usuarios	Algunos Usuarios	Pocos usuarios afectados.	Todos los usuarios.
Descubrimiento	Existe información	Vulnerabilidad afecta	El fallo no es trivial,

	pública que explica el ataque	a una parte de la aplicación que casi no se utiliza	no es muy probable utilizarlo para causar un daño.

Fuente: (Olivares y Eduardo 2017, p. 9)

Realizado por: Mesias Francisco, 2023.

A continuación, se procede aclarar algunas de las métricas de daño establecidas para reducir la ambigüedad de la adaptación de este modelo:

- **Información sensible:** Se refiere a aquellos datos privados y específicos de un individuo que, de ser divulgados, podrían comprometer su seguridad, privacidad ointereses. Esta abarca desde información personal, como domicilio, datos bancarios y contraseñas de correo, hasta información relacionada con la propiedad intelectual, que incluye documentos estratégicos, código fuente y precios. También se categoriza dentro de la información sensible aquella de carácter confidencial, relacionada con la reputación de una compañía, historias clínicas, estrategias y comunicaciones internas. (Álzate, Romaña y Barco 2015, p. 68; Ducuara Cruz, Moya Molano 2017, p. 8).
- **Información trivial:** La información trivial se refiere a aquellos datos o detalles que carecen de peso y relevancia significativa. Debido a esta característica, las personas a menudo procesan esta información de manera superficial y con un esfuerzo mental mínimo. (Vargas Bianchi 2003, p. 230).

2.7 Trabajos relacionados

En base a las investigaciones realizadas se puede tomar en cuenta los resultados obtenidos por (Ekenblad y Stefan 2022) en su tesis "Evaluación de seguridad de diez aplicaciones móviles suecas", trabajo realizado en la Universidad KTH Royal Institute of Technology, se exploró las vulnerabilidades de distintas aplicaciones móviles. Descubrieron doce posibles vulnerabilidades en siete de las diez aplicaciones analizadas de las cuales diez vulnerabilidades fueron seleccionadas para examinarlas a profundidad y de las cuales tres de las diez vulnerabilidades encontradas pudieron ser explotadas con éxito. Se logro siguiendo la Guía de Pruebas de Seguridad Movil de OWASP (MSGT) que ofrecía un manual exhaustivo para las pruebas de seguridad.

De igual manera (Borja Saltos 2021) en su Tesis "Análisis de seguridad de aplicaciones Android en base al top 10 de OWASP 2016" realizo pruebas de penetración de seguridad exhaustivas en varias aplicaciones Android, siguiendo los riesgos más comunes de acuerdo con OWASP

Mobile 2016 y utilizando diferentes herramientas como Drozer, Dex2jar y Android Debug Bridge que luego de llevar a cabo todas las pruebas de seguridad en diez escenarios diferentes,se ha observado que los problemas de seguridad están presentes en todas las aplicaciones estudiadas. En general, se han encontrado múltiples vulnerabilidades en cada aplicación analizada y estas vulnerabilidades son relativamente fáciles de explotar, lo que en todos loscasos conduce a varios problemas de seguridad.

CAPÍTULO III

3 Marco metodológico

En este capítulo se hace referencia a los mecanismos que serán usados para la resolución de la problemática principal que la aplicación móvil del Centro Médico de Especialidades “La Dolorosa”, esto haciendo uso de la metodología propuesta por OWASP en el Mobile Security Testing Guide usada para gestionar las fases de las pruebas de penetración de una aplicación móvil.

3.1 Diseño de investigación

Se detallan los elementos necesarios para realizar la investigación, estos son: tipo de estudio, los métodos, técnicas y fuentes de estudio. Además, se realiza la operacionalización conceptual de la variable: número de vulnerabilidades y la operacionalización metodológica de la misma. En la primera, se describe la característica a utilizar, su tipo de variable: cualitativa o cuantitativa y su concepto. Para la segunda se especifica además de su característica, la subcaracterística, la técnica a utilizar y la fuente de donde se tomarán los datos. Se definen los indicadores para realizar la medición de la variable, y se realiza el análisis de población y muestra de estudio, a más del planteamiento de hipótesis.

3.1.1 Tipo de Investigación

El tipo de estudio realizado en este trabajo es de naturaleza aplicativa y evaluativa, dado que pone en práctica las habilidades técnicas y conceptuales adquiridas durante la carrera. El propósito de este estudio fue aplicar pruebas de penetración a la aplicación móvil CMED, ya desarrollada, con el objetivo de identificar y evaluar su nivel de seguridad y vulnerabilidades potenciales.

Este enfoque permite una doble aplicación del conocimiento, por un lado, la implementación de herramientas y técnicas especializadas en pruebas de penetración y, por otro lado, la evaluación crítica y el análisis de los resultados de estas pruebas para determinar el nivel de seguridad de la aplicación. De esta manera, el estudio no solo se centra en la aplicación de conocimientos técnicos, sino que también cumple con el objetivo de mejorar la seguridad y protección de los datos del usuario en la aplicación móvil.

Este estudio también es aplicativo en la medida en que busca proporcionar soluciones prácticas y tangibles a los problemas de seguridad encontrados. A través de este enfoque, se espera no solo identificar las vulnerabilidades de la aplicación, sino también proponer buenas prácticas para corregirlas, garantizando así un mayor nivel de protección para los usuarios de la aplicación.

3.1.2 Métodos, técnicas y fuentes de estudio

La implementación de las medidas de seguridad al aplicativo movil se establecen los métodos, técnicas y fuentes en base a los objetivos planteados, como se puede ver en la **Tabla 1-3.**

Tabla 1-3: Métodos, técnicas y fuentes

Objetivos	Métodos	Técnicas	Fuentes
Estudiar los conceptos, características y herramientas de pentesting en aplicaciones móvilespara seleccionar la(s) herramienta(s) más adecuada(s).	• Analítico	• Revisión de documentación • Observación	• Proyecto de Seguridad Móvil de OWASP • Documentación de herramientas de pentesting • Artículos de revistas de ciberseguridad • Foros técnicos de seguridad informática
Analizar mediante pentesting las vulnerabilidades en la aplicación móvil.	• Metodología Seguridad de aplicaciones móviles OWASP para pruebas de penetración	• Preparación • Recopilación de Inteligencia • Mapeo de la Aplicación • Explotación • Informes	• OWASP • Herramientas de pentesting • Informes de pruebas de penetración, • Recursos de ciberseguridad
Aplicar acciones de mejora en la aplicación móvil en función del análisis de vulnerabilidades.	• Estándar de verificación de seguridad de aplicaciones móviles	• Buenas Practicas	• Guías de buenas prácticas de seguridad en aplicaciones móviles • Flutter Security • Documentation • Foros técnicos de seguridad informática
Evaluar las vulnerabilidades en la aplicación móvil.	• Estadístico • Analítico	• Evaluación de seguridad post-mejora	• OWASP • Herramientas de pentesting,

		• Pruebas de penetración iterativas	• Informes de pruebas de penetración • Recursos de ciberseguridad

Realizado por: Mesias Francisco. 2023

Para una mejor comprensión se describen cada uno de los métodos empleados en relación con su utilización en el desarrollo de este trabajo.

3.1.2.1 Método Analítico

Este método se utilizará para estudiar los conceptos, características y herramientas de pentesting en aplicaciones móviles con el fin de seleccionar la(s) herramienta(s) más adecuada(s). Este método se basa en el análisis y la revisión de documentos y literatura existente sobre el tema, como guías técnicas, artículos de revistas especializadas en ciberseguridad y discusiones en foros técnicos de seguridad informática.

3.1.2.2 Metodología Seguridad de aplicaciones móviles OWASP

Este método se empleará para analizar las vulnerabilidades en la aplicación móvil mediante pentesting. En este enfoque, se parte de un conocimiento general, en este caso la metodología seguridad de aplicaciones móviles para pruebas de penetración, y se aplica a un caso particular, la aplicación móvil que estás estudiando, se evaluará cómo interactúan los distintos componentes de la aplicación y cómo estos pueden ser susceptibles a distintos tipos de ataques.

3.1.2.3 Estándar de verificación de seguridad de aplicaciones móviles

Este método se utilizará para aplicar acciones de mejora en la aplicación móvil en función del análisis de vulnerabilidades. Después de identificar las vulnerabilidades, se realizará un análisis de estas para desarrollar y aplicar acciones de mejora. Posteriormente, se realizará una nueva ronda de pruebas de penetración para evaluar la eficacia de las mejoras implementadas. Este es un ciclo continuo de mejora.

3.1.2.4 Método Estadístico

Este método se utilizará para evaluar las vulnerabilidades en la aplicación móvil después de aplicar las acciones de mejora. Este método se basa en la observación directa y la experiencia

práctica. Después de aplicar las mejoras, se realizará una nueva ronda de pruebas de penetración para evaluar si las vulnerabilidades han sido efectivamente mitigadas o eliminadas.

3.1.3 Población y muestra de estudio

La población y muestra para el trabajo de integración curricular se basa en el sistema de aplicación móvil CMED. La población para este estudio incluye todas las funcionalidades, red ybase de datos de la aplicación.

De acuerdo con Garg y Baliyan (2021, p. 2), los sistemas operativos móviles más utilizados son Android, que posee una cuota de mercado de 72.95%. Para este estudio se realizó un muestreo no probabilístico por conveniencia, seleccionando exclusivamente el sistema operativo Android para la evaluación.

3.1.4 Planteamiento de Hipótesis

El planteamiento de hipótesis se realiza para el indicador de tiempo de respuesta para cada uno de los procesos principales para la gestión de la atención médica. Las hipótesis planteadas son:

- **Hipótesis de seguridad de la aplicación móvil CMED**

H_0 = La seguridad de la aplicación móvil CMED no experimenta una mejora significativa después de la implementación de las correcciones basadas en las pruebas de pentesting.

H_1 = La seguridad de la aplicación móvil CMED experimenta una mejora significativa después de la implementación de las correcciones basadas en las pruebas de pentesting.

3.1.5 Estudio de factibilidad

En el estudio de factibilidad, se proporciona al cliente una estimación del costo asociado con el análisis de vulnerabilidades y las pruebas de penetración de la aplicación móvil de salud CMED. Esta estimación incluye la consideración de los recursos económicos, técnicos y operativos necesarios para determinar si el proyecto de mejora de la seguridad es viable o no. Los diversos tipos de factibilidad se describen en el **Anexo A**.

3.2 Análisis comparativo de las herramientas de pruebas de penetración

Basándose en la información recopilada durante la investigación de herramientas de pruebas de penetración para aplicaciones móviles, se fundamenta la elección de cada instrumento. Estas

herramientas fueron categorizadas según los tipos de análisis de vulnerabilidades: estático y dinámico. A continuación, se proporcionan las justificaciones para la selección de MobSF, BurpSuite y SQLMap.

3.2.1 Análisis entre Mobile Security Framework y Androbugs

Ilustración 1-3: Análisis entre Mobile Security Framework y Androbugs
Realizado por: Mesias Francisco, 2023.

Al analizar Mobile Security Framework y AndroBugs basándose en la **Tabla 2-2**, MobSF destaca por varias razones. Una de las más notables es la facilidad de aprendizaje siendo un marco automatizado, MobSF brinda una experiencia intuitiva al usuario; en contraste, AndroBugs, al carecer de una interfaz gráfica, demanda mayores habilidades de programación. En la generación de informes, la diferencia se mantiene: MobSF ofrece informes detallados y bien estructurados que simplifican el trabajo de desarrolladores y especialistas en seguridad al implementar medidas correctivas, mientras que AndroBugs produce informes en formato txtque carecen del mismo nivel de detalle.

3.2.2 Análisis entre Burp Suite y Zed Attack Proxy

Ilustración 2-3: Análisis entre Burp Suite y Zed Attack Proxy
Realizado por: Mesias Francisco, 2023.

Basando en la recopilación de la **Tabla 3-2**. Tanto BurpSuite como Zed Attack Proxy tienen características que las sitúan como herramientas líderes en el análisis de seguridad, se ha optado

por BurpSuite porque se adapta mejor al proyecto debido a la representación de mensajes HTTP que intercepta. Al servir como un proxy interceptador y permitir una amplia personalización mediante funciones propias, ha mostrado una eficiencia notable en la identificación y gestión de vulnerabilidades.

3.2.3 Análisis entre SQLMap y Havij

Ilustración 3-3: Análisis entre SQLMap y Havij
Realizado por: Mesias Francisco, 2023

SQLMap sobresale notablemente sobre Havij gracias a su destacada habilidad para detectar vulnerabilidades, emplear técnicas avanzadas de explotación, gestionar consultas agrupadas y ofrecer una diversidad en técnicas de inyección; mientras que Havij tiene sus propias competencias, SQLMap presenta una especialización y alcance superiores. Al considerar todo lo que SQLMap ofrece en términos de profundidad, versatilidad y enfoque detallado, se evidencia su significativa ventaja en el ámbito de inyecciones SQL frente a Havij como se puede ver evidenciada esta información en la **Tabla 4-2**.

3.3 Análisis de vulnerabilidades de la aplicación móvil CMED utilizando OWASP

Para llevar a cabo las pruebas de penetración en la aplicación móvil CMED, se utiliza el marco de trabajo propuesto por la guía de Pruebas de Seguridad Móvil de OWASP, este marco de trabajo está compuesto por cinco fases, las cuales permiten un enfoque estructurado y exhaustivo en el proceso de pentesting.

Al emplear este enfoque, no solo se puede realizar una evaluación sistemática y exhaustiva de la seguridad de la aplicación móvil CMED, sino también proporcionar recomendaciones concretas y basadas en la evidencia para mejorar su seguridad.

3.3.1 Preparación

En esta etapa inicial, se establecen las bases para la implementación de las pruebas de penetración a la aplicación móvil CMED. Esta preparación implica la definición del alcance de las pruebas de seguridad, la sincronización con el cliente, y la obtención de protección legal.

3.3.1.1 Coordinación con el cliente

Durante la fase de preparación con el cliente, se acordó un documento legal entre Francisco Xavier Mesias Flores, referido en adelante como "Pentester", y Jhony Ruperto Riera Ortiz, conocido como "Cliente". Este acuerdo establece el alcance y las condiciones para la realizaciónde las pruebas de penetración a la aplicación móvil CMED antes que salga a producción.

Este contrato estipula que el Pentester llevará a cabo una evaluación de la seguridad de la aplicación móvil CMED del Cliente, utilizando una serie de herramientas y técnicas específicas, MobSF, Burp Suite y Sqlmap. El propósito de estas pruebas es identificar, analizar y proporcionar recomendaciones para corregir cualquier vulnerabilidad encontrada en laaplicación antes de su despliegue.

En el marco de este contrato, se trabajará en un entorno de caja blanca, es decir, con un acceso completo al código fuente y al entorno de desarrollo de la aplicación. En este caso las pruebas de seguridad se realizarán únicamente en la versión de desarrollo de la aplicación. Este enfoque se basa en el objetivo de identificar y analizar las posibles vulnerabilidades en la aplicación tal y como se encuentra en su estado final, simulando la perspectiva de un atacante potencial, pero con el beneficio de tener acceso al código fuente para un análisis más detallado. Para obtener un acercamiento más detallado sobre el documento legal, se puede revisar el **ANEXO B**.

3.3.1.2 Identificación de datos confidenciales

Hay tres tipos de datos a los cuales se puede acceder que son:

- **En estado de reposo:** los datos existen almacenados en un fichero o en una base de datos
- **En estado de uso:** los datos han sido cargados en el espacio de memoria utilizado por una aplicación.

- **En estado de tránsito:** los datos se están transfiriendo entre la aplicación móvil y el endpoint, o entre los procesos que consumen la información en el dispositivo, como en la comunicación entre procesos

En base a la clasificación de los datos que se pueden acceder propuesta por OWASP, se ha procedido a la identificación y categorización de los datos presentes en la aplicación móvil de estudio que se muestran en la **Tabla 2-3.**

Tabla 2-3: Datos confidenciales.

Datos en reposo	Datos del usuario, recetas, medicamento, dieta.
Datos en uso	Datos cargados paras las pruebas
Datos de transito	Datos del usuario, recetas, medicamento, dieta.

Realizado por: Mesias Francisco. 2023

3.3.2 Recopilación

En esta etapa, nos enfocamos en recopilar información crítica que ofrece un entendimiento contextual de la aplicación móvil CMED. Este proceso de recopilación de inteligencia se centra en obtener información detallada acerca de su arquitectura, su propósito comercial, y el contexto en el que opera. Esta información se categoriza en dos tipos principales: ambiental y arquitectónica.

3.3.2.1 Información Ambiental

La aplicación móvil ha sido creada con el propósito primordial de servir como un instrumento eficaz para la administración y supervisión de recetas médicas, notifica la medicación para los pacientes y el acceso inmediato a la información de sus recetas médicas y dietas, proporcionando a los pacientes un control más eficaz de sus tratamientos y un acceso fácil a la información de salud relevante.

La aplicación móvil CMED se ubica dentro de las aplicaciones mHealth donde los usuarios almacenan información personal en la aplicación móvil, la seguridad de las aplicaciones móviles se ha convertido en una preocupación crítica especialmente para las aplicaciones de salud móviles mHealth, que manejan datos sensibles como la información persona, de salud y las prescripciones médicas.

Estos datos necesitan ser protegidos de manera rigurosa debido a los riesgos de exposición de información personal, robo de identidad y violación de la privacidad del paciente. Además, el

marco legal ecuatoriano establece el derecho a la protección de datos personales, lo que subraya la necesidad de garantizar la seguridad de la información en la aplicación móvil. Por lo tanto, la implementación de buenas prácticas de seguridad y la realización de pruebas de penetración son esenciales para garantizar una adecuada protección de los datos personales de los usuarios en la aplicación.

3.3.2.2 Información Arquitectónica

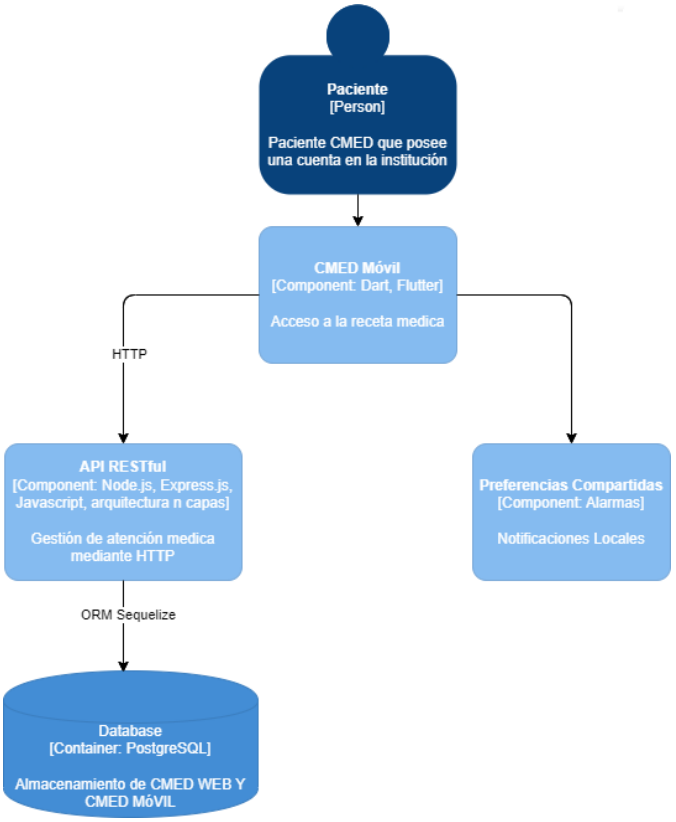


Ilustración 4-3: Diagrama C4 de Contenedores
Fuente: (Riera 2023, p. 51).
Realizado por: Mesias Francisco, 2023.

En el lado del cliente, tenemos a los pacientes que interactúan con la aplicación móvil permitiendo los pacientes controlar sus prescripciones médicas, conocer su progreso de medicación y acceder a sus recetas médicas, en el lado del servidor, la aplicación se comunica a través de HTTP con una API RESTful construida con Node.js y Express.js siguiendo una arquitectura de n-capas esta arquitectura permite una clara separación de responsabilidades y hace que el mantenimiento y la escalabilidad sean más sencillos por último la API se comunica con la base de datos a través de un ORM, facilita la manipulación de datos que se almacenan en la base de datos echa en PostgreSQL.

3.3.3 Mapeo de la aplicación

Como parte de una evaluación de caja blanca, el objetivo es obtener una comprensión completa de la estructura y funcionalidad de la aplicación móvil, con el acceso a la documentación interna del proyecto, como diagramas de arquitectura, especificaciones funcionales y el código, podemos analizar y mapear minuciosamente cada aspecto de la aplicación como puntos de entrada, características, y datos, lo cual será fundamental para una evaluación de seguridad, todos estos puntos se lo hace mediante el modelo de amenazas.

3.3.3.1 Descomponer la aplicación

Los casos de uso es un componente de la seguridad de aplicaciones debido a que proporciona una visión de como los usuarios interactúan con el sistema haciendo posible identificar posibles lugares de ataques, cada caso de uso define una ruta que un ciberdelincuente podría explotar, así como también ayuda a identificar áreas de riesgo y potenciales vulnerabilidades. A través de estos casos de uso se los toma como fundamento para realizar los siguientes pasos del procesode amenazas propuesta por OWASP.

3.3.3.1.1 Información del modelo de amenazas

Nombre de la aplicación: CMED

Versión de la aplicación: 1.0

Descripción: La aplicación móvil CMED proporciona servicios de salud personalizados a paciente, la aplicación permite a los pacientes visualizar su receta médica, gestionar notificaciones locales, registrar y monitorear su consumo de medicamentos, y visualizar su progreso diario de medicación.

Los pacientes podrán iniciar sesión, visualizar su receta médica, visualizar y verificar su consumo de medicamentos, gestionar notificaciones locales y visualizar su progreso diario de medicación.

3.3.3.1.2 Dependencias externas

Las dependencias externas se refieren a todos aquellos elementos o componentes que son esenciales para el funcionamiento de la aplicación pero que no están directamente bajo el

control del equipo de desarrollo. Pueden incluir bibliotecas, marcos de trabajo, sistemas operativos, servidores, servicios en la nube, bases de datos, APIs de terceros, entre otros.

A continuación, en la **Tabla 3-3** se muestra el listado de las dependencias externas de la aplicación móvil CMED.

Tabla 3-3: Dependencias externas de la aplicación móvil CMED

Identificación	Descripción
1	ORM Sequelize se utiliza como una dependencia para el manejo eficiente de la base de datos. Cualquier vulnerabilidad en Sequelize puede ser una amenaza para la seguridad de la aplicación.
2	La aplicación depende de Flutter para la creación de interfaces de usuario, lo que puede introducir vulnerabilidades si no se mantienen actualizados los parches y las versiones de Flutter.
3	La aplicación fue desarrollada para interactuar con otras plataformas y sistemas, lo que significa que cualquier vulnerabilidad en estos sistemas o en el proceso de interoperabilidad podría ser una amenaza para la seguridad de la aplicación.
4	Dado que la aplicación está diseñada para manejar datos sensibles de los pacientes, cualquier dependencia externa que interactúe con estos datos debe ser segura y confiable para evitar cualquier amenaza a la privacidad del paciente.

Realizado por: Mesias Francisco, 2023.

3.3.3.1.3 Puntos de entrada/salida

Los puntos de entrada y salida son componentes esenciales en la arquitectura de seguridad de una aplicación. Los puntos de entrada representan las vías por las cuales los datos son introducidos en la aplicación, sucediendo cada vez que un usuario interactúa con la misma, ya sea iniciando sesión, completando formularios, entre otros. Estos puntos son susceptibles de ser explotados por atacantes que pueden introducir datos malintencionados o intentar manipular la aplicación, las amenazas habilitadas por los puntos de salida están relacionadas con las amenazas del punto de entrada correspondiente.

A continuación, en la **Tabla 4-3** se muestra el listado de los puntos de entrada y salida de la aplicación móvil CMED.

Tabla 4-3: Punto de entrada y salida de aplicación móvil CMED

Id	Nombre	Descripción	Niveles de Confianza
1	Ingreso de credenciales	El paciente ingresa sus credenciales para acceder a la aplicación	Paciente registrado
2	Visualización de dietas	El paciente visualiza sus dietas prescritas.	Paciente registrado
3	Visualización de recetas médicas	El paciente visualiza su receta médica.	Paciente registrado
4	Registro de consumo de medicamentos	El paciente registra el consumo de un medicamento en particular.	Paciente registrado

Realizado por: Mesias Francisco, 2023.

3.3.3.1.4 Activo

Un activo físico puede referirse a elementos tangibles como servidores, ordenadores, dispositivos móviles, mientras que un activo abstracto puede ser intangible como la reputación de una empresa, la información de los clientes, la propiedad intelectual, entre otros. Ambostipos de activos necesitan protección, ya que un ataque exitoso puede resultar en daño financiero, pérdida de confianza, violación de la privacidad, entre otros.

A continuación, en la **Tabla 5-3** se muestra el listado de activos de la aplicación móvil CMED.

Tabla 5-3: Activos de aplicación móvil CMED

Id	Nombre	Descripción	Niveles de Confianza
1	Usuarios del sistema	Pacientes que utilizan la aplicación para gestionar sus dietas y medicación.	Paciente registrado
1.1	Credenciales de los pacientes	Las credenciales de los pacientes son necesarias para acceder a la aplicación. Necesitan ser protegidas para evitar el acceso no autorizado.	Paciente registrado
2	Sistema	Involucra todo el sistema subyacente y la infraestructura que sostiene la aplicación.	Desarrollador del sistema
2.1	Aplicación móvil	La aplicación móvil es el principal punto de interacción con los pacientes. Debe estar protegida contra amenazas como	Desarrollador del sistema

		inyecciones de código, suplantación de identidad, entre otros.	
3	Información de los pacientes	Los datos manejados por la aplicación son un activo crítico.	Paciente registrado, Desarrollador del sistema
3.1	Información de dieta de los pacientes	La información de la dieta de cada paciente es un activo importante que necesita protección.	Paciente registrado
3.2	Receta médica de los pacientes	La receta médica es un activo sensibleque necesita protección, contiene información sobre los medicamentos prescritos a cada paciente.	Paciente registrado

Realizado por: Mesias Francisco, 2023.

3.3.3.1.5 Niveles de Confianza

Corresponden a las autorizaciones de acceso que una aplicación asigna a los distintos actores o entidades que interactúan con ella. En esencia, estos niveles establecen qué permisos o privilegios tiene cada actor para interactuar con los diversos componentes de la aplicación, ya sean puntos de entrada, activos o procesos.

A continuación, en la **Tabla 6-3** se muestra el listado de activos de la aplicación móvil CMED.

Tabla 6-3: Niveles de confianza de aplicación móvil CMED

Id	Nombre	Descripción
1	Paciente registrado	Un usuario que se ha registrado e iniciado sesión en la aplicación con credenciales válidas.
2	Desarrollador del sistema	Una persona con acceso al código fuente de la aplicación, capaz de realizar cambios en el sistema.

Realizado por: Mesias Francisco. 2023.

3.3.3.1.6 Diagramas de flujo de datos

Los Diagramas de Flujo de Datos (DFDs) son una herramienta esencial que se utiliza en la modelación de aplicaciones. Los DFDs facilitan una representación visual de cómo una aplicación procesa y maneja los datos, a través de estos diagramas se puede identificar lospuntos críticos de la aplicación móvil que se representan a través de de símbolos que se los explica en la **Tabla 7-3**.

Tabla 7 -3: Símbolos para los diagramas de flujo de datos

Símbolo	Nombre	Descripción
	Entidad Externa	La forma de entidad externa se usa para representar una entidad fuera de la aplicación que interactúa con la aplicación a través de un punto de entrada.
	Flujo de datos	La forma del flujo de datos representa el movimiento de datos dentro de la aplicación. La dirección del movimiento de datos está representada por la flecha.
	Límite de privilegios	La forma del límite de privilegios se utiliza para representar el cambio de los niveles de confianza a medida que los datos fluyen a través de la aplicación. Los límites muestran cualquier ubicación donde cambie el nivel de confianza.
	Almacenamiento de datos	La forma del almacén de datos se utiliza para representar las ubicaciones donde se almacenan los datos. Los almacenes de datos no modifican los datos, solo almacenan datos.

Fuente: (Conklin 2023)

Realizado por: Mesias Francisco, 2023.

Como se puede visualizar en la **Ilustración 1-3** se representa el flujo de datos en la aplicación móvil CMED, desde el punto de vista del paciente registrado y el desarrollador del sistema. El diagrama ilustra cómo estos actores interactúan con la aplicación, el flujo de datos a través de ella, y cómo estos datos se procesan y almacenan en nuestra base de datos.

Ilustración 5-3: Diagrama de flujo de datos de la aplicación móvil CMED
Realizado por: Mesias Francisco, 2023

Es importante mencionar que la elaboración de este DFD no solo tiene como objetivo ilustrar cómo funciona la aplicación. Más allá de eso, nos permite identificar potenciales puntos críticos que podrían convertirse en vulnerabilidades de seguridad o privacidad. Estos puntos críticos representan partes de la aplicación donde los datos se ven más expuestos a posibles ataques o donde las amenazas a la seguridad podrían tener un impacto mayor. En el siguiente apartado, se detallan los puntos críticos identificados en el DFD y se explican las posibles amenazas y vulnerabilidades asociadas a cada uno de ellos.

- **Puntos de Entrada:** Si la aplicación móvil no maneja correctamente la autenticación y autorización de los usuarios, un atacante podría potencialmente acceder a las funcionalidades de un paciente registrado sin necesidad de tener sus credenciales.
- **Servidor:** Si la aplicación no valida y escapa correctamente los datos del usuario, un atacante podría inyectar código malicioso.
- **ORM Sequelize:** Si las consultas a la base de datos no están bien estructuradas, podrían ser susceptibles a inyecciones SQL. Un atacante podría manipular las consultas para obtener, alterar o borrar datos de la base de datos.
- **Información de los Pacientes:** Si estas viajan por la red texto claro o no están debidamente protegidas, podrían ser vulnerables a ataques como el robo de información sensible.

3.3.4 Explotación

Durante esta etapa se determina las vulnerabilidades de la aplicación móvil CMED para lograr esto se utiliza herramientas de pruebas de penetración y análisis de seguridad como MobSF,

Burp Suite y SQLMap con un enfoque en de caja blanca, debido al conocimiento total de la estructura y funcionalidad que a través del modelado de amenazas se procedió a analizar cada estructura mediante el diagrama DFD permitiendo una evaluación más segura, detallada y precisa.

3.3.4.1 Análisis estático

3.3.4.1.1 Revisión manual

Como parte de la revisión manual del sistema, se realizó un análisis de los Diagramas de Flujo de Datos y de código para entender cómo los datos se mueven a través de la aplicación y para identificar posibles puntos débiles que podrían ser explotados por un atacante. Este análisis permitió identificar la siguiente vulnerabilidad potencial.

3.3.4.1.1.1 Insuficiencia de autenticación

La aplicación en estudio actualmente sólo maneja la autenticación a través de contraseñas y claves, lo que es insuficiente para proteger la cuenta de un usuario contra amenazas más sofisticadas. Además, la aplicación puede estar en riesgo si las sesiones de usuario no están correctamente manejadas y protegidas para prevenir ataques de secuestro de sesión, ataques de fuerza bruta, etc.

3.3.4.1.2 MobSF

Se utilizó la herramienta MobSF, examino las entidades externas y procesos, la herramienta analizo permisos, APKID, análisis de red, certificados, manifiesto, código, bibliotecas. NIAP v.1.3. Estos análisis han proporcionado una visión más profunda de la estructura de la aplicación y los posibles puntos débiles y vulnerabilidades existentes en la aplicación, el informe del análisis de MobSF se encuentra en el **Anexo C.**

En el apartado de “shared library binary analysis” que se encuentra en el **Anexo C** sobre el informe que nos arroja MobSF, no se procederá a tener en cuenta estas vulnerabilidades, debido a que Flutter ya cuenta con ciertas características de vulnerabilidad integradas, incorpora bit NXlo que previene la ejecución de código malicioso inyectado por atacantes, aparte flutter no incorpora stack canary debido a que dart no genera esto porque no tiene matrices de pila asignada a comparación como C/C++. Las funciones fortificadas MobSF obtiene falsos positivos debido a que Dart no usa estas funciones en absoluto. Tampoco se tomará en cuenta la

vulnerabilidad “Aplicación puede leer/escribir en almacenamiento externo”, las herramientas de escaneo interpretan la capacidad de la imagen Plugins de selector como una amenaza, esto no es una vulnerabilidad. (Flutter dev 2023b).

A continuación, se detallan las vulnerabilidades encontradas con la herramienta MobSF.

3.3.4.1.2.1 Android.Permission.REQUEST_INSTALL_PACKAGES

La siguiente vulnerabilidad implica que una aplicación en Android puede solicitar permiso para instalar otros paquetes o aplicaciones, lo cual puede ser explotado por aplicaciones maliciosas para instalar software no deseado sin el conocimiento o el consentimiento del usuario, puede ser manipulado para instalar automáticamente malware o aplicaciones intrusivas. (Google play developer 2023).

3.3.4.1.2.2 Android.Permission.REQUEST_EXTERNAL_STORAGE

La siguiente vulnerabilidad está relacionada con el acceso a los archivos y atributos de directorio del dispositivo de un usuario. Esos elementos son considerados datos personales y sensibles, y, en consecuencia, están protegidos por políticas de privacidad. Si una aplicación solicita este permiso y se lo concede, puede tener acceso a la totalidad del almacenamiento del dispositivo, lo que puede ser explotado para obtener acceso a información confidencial, realizar modificaciones no deseadas o incluso instalar software malicioso. Este permiso solo debe ser solicitado por las aplicaciones cuando es esencial para su funcionalidad, y no debe ser usadopara solicitar acceso al almacenamiento del dispositivo en nombre de terceros, el mal uso deeste permiso puede tener graves implicaciones para la privacidad y seguridad del usuario. (Android dev, 2020a).

3.3.4.1.2.3 Janus Vulnerability

Un atacante agregar un archivo DEX malicioso a un archivo APK, sin afectar su firma, el tiempo de ejecución de Android luego acepta el archivo APK como una actualización válida de una versión anterior legítima de la aplicación, pero la VM Dalvik carga el código del archivo DEX inyectado. Esto permite al atacante eludir el proceso de verificación de la firma e instalar código no verificado con permisos poderosos en el dispositivo de usuarios desprevenidos, esta vulnerabilidad afecta a los dispositivos Android (Android 5.0 < 8.1) firmados con el esquema de firma. (Kal 2021).

3.3.4.1.2.4 Application signed with debug certificate

La siguiente vulnerabilidad ocurre cuando una aplicación Android está firmada con un certificado de depuración en lugar de uno de lanzamiento. Android requiere que todos los APK estén firmados digitalmente con un certificado antes de ser instalados o ejecutados. Usar una firma de lanzamiento se utiliza para verificar la identidad del propietario durante las actualizaciones de la aplicación, evitando que la aplicación sea manipulada o modificada para incluir código malicioso. (Android dev, 2020b).

3.3.4.1.2.5 Certificate algorithm vulnerable to hash collision

Estos algoritmos de firma son conocidos por ser vulnerables a ataques de colisión, un atacante puede aprovechar esto para generar otro certificado con la misma firma digital, lo que permite al atacante hacerse pasar por el servicio afectado. (Microsoft 2018).

3.3.4.1.2.6 App can be installed on a vulnerable Android version

Se refiere a la capacidad de una aplicación para instalarse en una versión antigua y potencialmente vulnerable de Android, que contiene varias vulnerabilidades de seguridad no corregidas. (Lake, 2018).

3.3.4.1.2.7 Application Data can be Backed up

Si el android:allowBackup está establecido en verdadero en una aplicación, entonces toda la información de la aplicación en tiempo de ejecución, como las preferencias compartidas y las bases de datos, puede ser respaldada por cualquiera que tenga acceso al dispositivo. Esta posibilidad presenta una vulnerabilidad de que un actor malintencionado pueda obtener acceso a dicha información. (Doss et al. 2019).

3.3.4.1.2.8 The App uses an insecure Random Number Generator

La siguiente vulnerabilidad se refiere al uso inseguro de un Generador de Números Aleatorios en la aplicación, los generadores de números pseudoaleatorios estándar no son adecuados para resistir ataques criptográficos. (Microsoft 2023b).

3.3.4.2 Análisis dinámico

En el análisis dinámico, nos enfocamos en la aplicación móvil mientras está en ejecución y su interacción con los componentes del backend, como el servidor y la base de datos. A través de esta metodología, se puede observar cómo la aplicación se comporta durante la ejecución y cómo gestiona los datos y las solicitudes en tiempo real. Este enfoque nos permite identificar vulnerabilidades que pueden no ser evidentes durante un análisis estático. En la **Ilustración 6-3** diagrama, mostramos el escenario de interacción que se analizó:

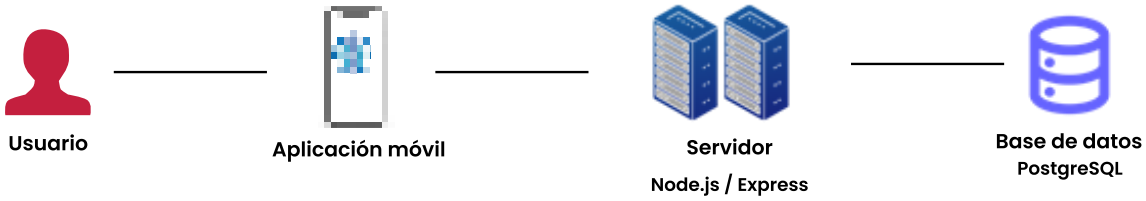


Ilustración 6-3: Escenario general para el análisis dinámico de la aplicación móvil CMED
Realizado por: Mesias Francisco, 2023

A partir de este escenario, realizamos pruebas de penetración utilizando las siguientes herramientas.

3.3.4.2.1 Burpsuite

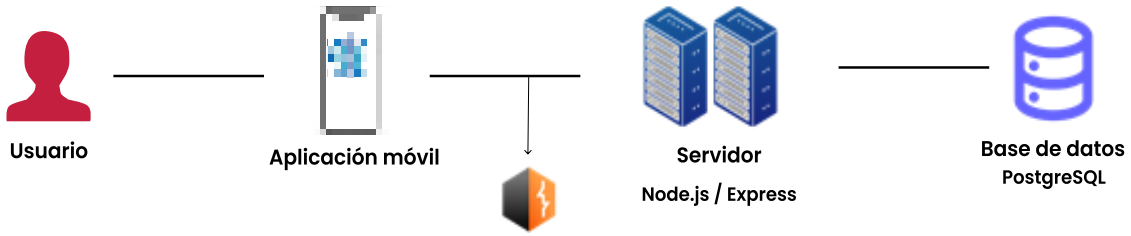


Ilustración 7-3: Escenario de análisis con la herramienta BurpSuite
Realizado por: Mesias Francisco, 2023

Como se indica en la **Ilustración 7-3** parte del análisis dinámico de la aplicación móvil, se empleó Burp Suite, una de las principales herramientas de seguridad para realizar pruebas de intrusión. El escenario de prueba implicó el montaje de un servidor local y la intercepción del tráfico entre el servidor y la aplicación móvil, permitiendo inspeccionar en tiempo real todas las solicitudes y respuestas que circulaban entre ambos.

3.3.4.2.1.1 Transmisión de datos inseguros

Como se muestra en la **Ilustración 8-3** se muestra el análisis realizado con Burp Suite permitió observar que la aplicación móvil no estaba realizando una adecuada encriptación de los datos durante su transmisión. Este hallazgo fue preocupante, ya que los datos que circulaban sin el debido cifrado incluían información delicada como las credenciales de usuario, las recetas médicas, las dietas y otros detalles médicos y personales de los pacientes.

Ilustración 8-3: Resultado de la transmisión de datos con la herramienta BurpSuite
Realizado por: Mesias Francisco, 2023

En relación con las recetas médicas, existe el riesgo de que un atacante pueda suplantar la identidad del paciente con el objetivo de acceder a servicios médicos. Además, el atacante podría intentar extorsionar al paciente amenazándolo con divulgar su información médica privada. Por otra parte, basándose en los detalles de la receta médica, dieta, medicamentos, el atacante tiene la capacidad de generar correos electrónicos fraudulentos, solicitando información adicional o dinero al paciente, lo cual aumenta aún más el peligro de la exposición de información sensible.

3.3.4.2.2 Sqlmap

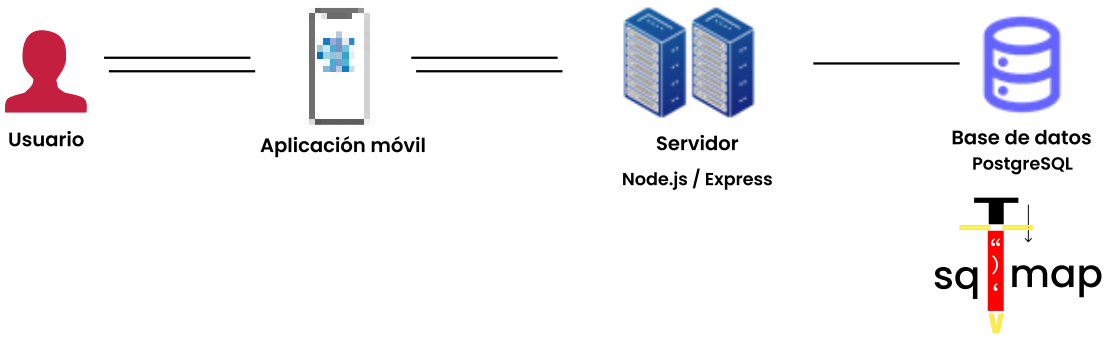


Ilustración 9-3: Escenario de análisis con la herramienta SQLMap
Realizado por: Mesias Francisco, 2023

Como se indica en la **Ilustración 9-3**, el proceso de prueba implica un flujo desde el usuario, pasando por la aplicación móvil y el servidor, hasta llegar a la base de datos, donde sqlmap realiza sus intentos de inyección. SQLMap actúa directamente sobre la base de datos, intentandoexplotar posibles fallos en la conexión entre el servidor y la misma. Para llevar a cabo la prueba,se realizó un ataque de fuerza bruta durante un período de aproximadamente 20 minutos. Se utilizaron una variedad de sentencias SQL con el propósito de identificar posibles brechas en la seguridad de la aplicación.

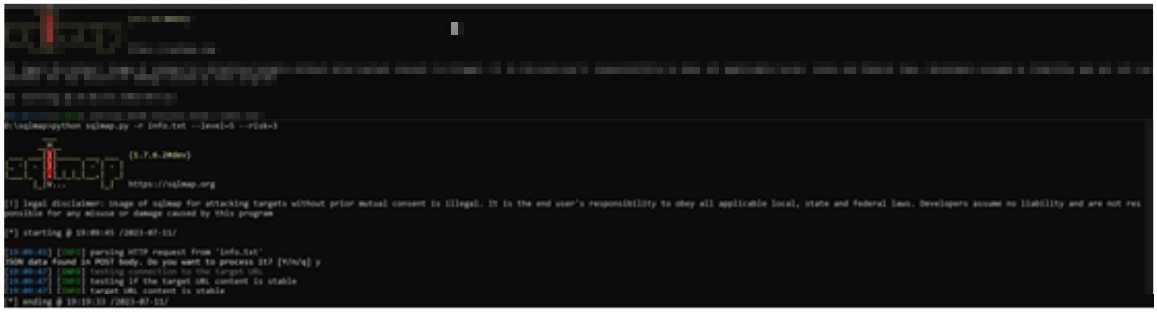


Ilustración 10-3: Resultado de la herramienta SQLMap
Realizado por: Mesias Francisco, 2023

La **Ilustración 10-3** muestra los comandos exactos utilizados durante la prueba. Se optó por un nivel de intensidad escalado hasta llegar a --level=5 y un nivel de riesgo --risk=3, para asegurar una exploración exhaustiva de las posibles vulnerabilidades. La prueba comenzó exactamente a las 19:02:36 y finalizó a las 19:19:33. Tras finalizar la prueba, se observó que la aplicación demostró una robusta resistencia contra intentos de inyección SQL. A lo largo de toda la prueba, sqlmap no detectó ninguna vulnerabilidad de inyección de SQL en la aplicación.

3.3.5 Informe

3.3.5.1 Resumen Ejecutivo

En este informe se presenta el resultado del análisis de seguridad de la aplicación móvil CMED. Con la ayuda de herramientas de análisis estático y dinámico como MobSF, Burpsuite ySqlmap, se identificaron un total de 10 amenazas en la aplicación. Estas amenazas fueron posteriormente clasificadas y priorizadas utilizando el método DREAD, destacando 7 amenazas de alto riesgo, 3 de riesgo medio y 0 de bajo riesgo.

3.3.5.2 Descripcion y Alcance

El objetivo de este análisis de seguridad fue evaluar la aplicación móvil "CMED", específicamente desarrollada con el marco de trabajo Flutter. El análisis de seguridad se realizó tanto a nivel de código (análisis estático) como a nivel de red y base de datos (análisis dinámico).

3.3.5.3 Métodos Utilizados

Para llevar a cabo este análisis se utilizó un enfoque de dos etapas. En primer lugar, se realizó un análisis estático del código de manera manual y utilizando la herramienta MobSF, lo que permitió identificar vulnerabilidades a nivel de código. En segundo lugar, se realizó un análisis dinámico utilizando Burpsuite para la intercepción de paquetes de red y Sqlmap para analizar la seguridad de la base de datos.

3.3.5.4 Hallazgos

Una vez identificadas las amenazas, basándonos en el enfoque adaptado de DREAD, hemos clasificado cada una de las amenazas identificadas en la aplicación móvil. Estas amenazas se detectaron mediante análisis dinámico y estático, utilizando herramientas especializadas en pruebas de penetración. Esta clasificación proporciona un panorama detallado de los riesgos a los que está expuesta nuestra aplicación, permitiéndonos así definir medidas de mitigación adecuadas.

En la **Tabla 8-3** se analiza una vulnerabilidad crítica en el sistema de autenticación de la aplicación, específicamente en la ausencia de autenticación biométrica, que presenta desafíos significativos en términos de seguridad de los datos del usuario.

Tabla 8-3: Calificación DREAD vulnerabilidad autenticación biométrica

	Puntaje	Justificación
D	2	La autenticación solo mediante contraseñas y claves puede permitir el acceso no autorizado a la información sensible del usuario si estas credenciales son descubiertas o comprometidas, pero no permite a un atacante ejecutar aplicaciones con permisos de administrador.
R	3	Un atacante podría intentar adivinar las contraseñas o utilizar técnicas de fuerza bruta para ganar acceso, lo cual es fácilmente reproducible si no hay suficientes medidas de seguridad, como bloqueo de cuenta después de varios intentos fallidos.
E	2	Un atacante necesitaría cierta experiencia y conocimientos para realizar un ataque exitoso, como entender cómo funcionan los ataques de fuerza bruta y tener la capacidad de ejecutarlos, o ser capaz de emplear técnicas de ingeniería social para obtener las credenciales del usuario.
A	3	Todos los usuarios de la aplicación están potencialmente en riesgo ya que todos usan el sistema de autenticación proporcionado por la aplicación.
D	2	Existen recursos y herramientas públicamente disponibles que explican cómo realizar ataques de fuerza bruta o adivinación de contraseñas, lo que facilitaría el descubrimiento de esta vulnerabilidad por parte de un atacante.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 9-3** se presenta un análisis detallado de la vulnerabilidad asociada con el permiso de Android 'REQUEST_INSTALL_PACKAGES'. Esta vulnerabilidad, si se explota, podría permitir la instalación no autorizada de paquetes o aplicaciones maliciosas en los dispositivos de los usuarios.

Tabla 9-3: Calificación DREAD vulnerabilidad REQUEST_INSTALL_PACKAGES

	Puntaje	Justificación
D	3	La explotación de esta vulnerabilidad puede llevar a la instalación de software malicioso en el dispositivo del usuario sin su consentimiento, lo que puede dar al atacante un control considerable sobre el dispositivo del usuario.
R	3	La reproducción de este ataque sólo requiere que una aplicación maliciosa solicite el permiso y que el usuario lo conceda, lo que puede ser fácilmente logrado a través de ingeniería social o interfaces de usuario engañosas.
E	2	Un programador experimentado con conocimiento de Android puede diseñar una aplicación para explotar esta vulnerabilidad.
A	3	Todos los usuarios que instalen una aplicación que explote esta vulnerabilidad estarían en riesgo.
D	2	Existe información pública sobre cómo las aplicaciones pueden solicitar e instalar paquetes, aunque el conocimiento de cómo abusar de este permiso para fines maliciosos puede no ser tan fácilmente accesible.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 10-3** se examina la vulnerabilidad asociada con el permiso 'REQUEST_EXTERNAL_STORAGE' de Android. Este permiso, si se explota, otorga acceso

total al almacenamiento del dispositivo, lo que puede resultar en la divulgación de información confidencial, alteraciones no deseadas en los datos del usuario, o incluso la instalación de software malicioso.

Tabla 10-3: Calificación DREAD vulnerabilidad REQUEST_EXTERNAL_STORAGE

	Puntaje	Justificación
D	3	Al tener acceso a los archivos y directorios del dispositivo, un atacante puede obtener información sensible del usuario y, potencialmente, modificar los datos almacenados o instalar software malicioso.
R	2	El ataque depende de que la aplicación maliciosa solicite el permiso y que el usuario lo conceda, lo que puede lograrse a través de ingeniería social o interfaces de usuario engañosas.
E	2	Un programador experimentado con conocimiento de Android podría crear una aplicación que abuse de este permiso.
A	3	Todos los usuarios que instalen una aplicación que explote este permiso estarían en riesgo, ya que el permiso da acceso a todo el almacenamiento del dispositivo.
D	2	La información sobre cómo una aplicación puede solicitar y utilizar el permiso de almacenamiento externo es pública, aunque explotarla con fines malintencionados puede no ser trivial.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 11-3** se analiza la vulnerabilidad conocida como 'Janus', la cual afecta a ciertas versiones de dispositivos Android. Esta vulnerabilidad permite a los atacantes instalar código no verificado con permisos potentes en los dispositivos de los usuarios, eludiendo así el proceso de verificación de la firma. Esto representa una amenaza significativa para la seguridad y la privacidad del usuario.

Tabla 11-3: Calificación DREAD vulnerabilidad Janus

	Puntaje	Justificación
D	3	El atacante puede instalar código no verificado con permisos potentes en los dispositivos de los usuarios. Esta acción puede resultar en la ejecución de aplicaciones con permisos de administrador, lo que podría causar un daño significativo.
R	2	La vulnerabilidad es reproducible, pero sólo en ciertas condiciones, ya que depende de la versión de Android y del esquema de firma usado. Sólo los dispositivos Android (Android 5.0 < 8.1) firmados con el esquema de firma v1 son vulnerables.
E	3	Un programador novato podría aprovechar la vulnerabilidad si conoce los parámetros específicos, como la versión de Android y el esquema de firma usado.
A	2	Aunque esta vulnerabilidad puede ser crítica, no todos los usuarios se ven afectados, ya que depende de la versión de Android y del esquema de firma usado.
D	1	Aunque existe información pública sobre la vulnerabilidad, su explotación no es trivial y afecta a una parte de la aplicación que rara vez se utiliza, por lo que es menos probable que se utilice para causar daño.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 12-3** aborda la vulnerabilidad asociada con las aplicaciones Android firmadas con un certificado de depuración en lugar de uno de lanzamiento. Este tipo de práctica insegura puede permitir a los atacantes manipular o modificar la aplicación para incluir código malicioso, comprometiendo la seguridad del usuario y la integridad de los datos.

Tabla 12-3: Calificación DREAD vulnerabilidad application signed with debug certificate

	Puntaje	Justificación
D	2	Una aplicación firmada con un certificado de depuración en lugar de uno de lanzamiento puede llevar a la divulgación de información sensible, ya que puede permitir la manipulación o modificación de la aplicación para incluir código malicioso.
R	3	Este ataque es fácilmente reproducible, ya que solo requiere que el atacante encuentre una aplicación firmada con un certificado de depuración.
E	2	Si bien la explotación requiere un conocimiento específico de las aplicaciones Android y su proceso de firma, un programador experimentado podría implementar el ataque.
A	2	Solo se ven afectados los usuarios que han instalado aplicaciones firmadas con un certificado de depuración, lo cual podría no incluir a todos los usuarios.
D	3	La información sobre este tipo de vulnerabilidad es de dominio público y su explotación es bastante conocida en la comunidad de seguridad.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 13-3** se examina la vulnerabilidad asociada con el uso de algoritmos de certificado que son susceptibles a colisiones de hash. Este tipo de vulnerabilidad puede ser explotada por un atacante para generar un certificado con la misma firma digital, permitiéndole hacerse pasar por el servicio afectado. Este problema pone en peligro la seguridad del usuario y la integridad de los datos.

Tabla 13-3: Calificación DREAD vulnerabilidad certificate algorithm vulnerable Android version

	Puntaje	Justificación
D	2	Si un atacante genera otro certificado con la misma firma digital, podría tener acceso a información sensible, permitiéndole hacerse pasar por el servicio afectado.
R	2	El ataque se podría reproducir, pero requiere de condiciones muy específicas, como el uso de un algoritmo vulnerable a colisiones de hash y el conocimiento para explotar esta vulnerabilidad.
E	2	Un programador experimentado podría implementar este ataque, aunque requiere un conocimiento específico sobre colisiones de hash y certificados digitales.
A	2	Los usuarios afectados serían aquellos que confían en certificados firmados con algoritmos

		vulnerables a colisiones de hash. No todos los usuarios podrían estar afectados, solo aquellos que usan servicios que emplean estos certificados.
D	3	Existen muchas informaciones públicas que explican cómo se puede explotar este tipo de vulnerabilidad, lo que facilita su descubrimiento y potencial explotación.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 14-3** se examina la vulnerabilidad versión minSDK de Android. Estavulnerabilidad permite que una aplicación se instale en una versión antigua y potencialmente vulnerable de Android, la cual puede contener numerosas vulnerabilidades de seguridad no corregidas. Esto puede resultar en la divulgación de información sensible y poner en peligro la seguridad del usuario.

Tabla 14-3: Calificación DREAD vulnerabilidad App can be installed on a vulnerable Android version

	Puntaje	**Justificación**
D	2	La aplicación podría instalarse en una versión antigua y potencialmente vulnerable de Android, que contiene varias vulnerabilidades de seguridad no corregidas, lo que puede resultar en la divulgación de información sensible.
R	3	Este ataque es fácilmente reproducible ya que simplemente se basa en instalar la aplicación en una versión antigua de Android.
E	2	Un programador experimentado podría explotar fácilmente estas vulnerabilidades, aunque el atacante necesita tener acceso al dispositivo que ejecuta una versión antigua de Android.
A	3	Esto afectaría a todos los usuarios que todavía están usando versiones antiguas de Android.
D	3	Existen numerosos informes y documentación pública que explican las vulnerabilidades de versiones antiguas de Android, lo que facilita su descubrimiento y explotación.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 15-3** se examina a la configuración de copia de seguridad de Android. Cuando 'android:allowBackup' está establecido en verdadero en una aplicación, toda la información de la aplicación, como las preferencias compartidas y las bases de datos, puede ser respaldada por cualquiera que tenga acceso al dispositivo. Esta característica puede convertirse en una vulnerabilidad si un actor malintencionado consigue acceso a esta información

Tabla 15-3: Calificación DREAD vulnerabilidad Application Data can be Backed up

	Puntaje	**Justificación**
D	2	Si el atacante tiene acceso al dispositivo, puede respaldar y obtener la información de la aplicación, lo que podría resultar en la divulgación de información sensible.
R	2	El ataque se podría reproducir, pero requeriría condiciones específicas, como el acceso físico al dispositivo.

E	2	Un programador experimentado podría implementar este ataque si tiene acceso al dispositivo.
A	2	Afecta a aquellos usuarios que tienen la configuración de copia de seguridad activada y a quienes un atacante tiene acceso a su dispositivo.
D	3	Existen recursos públicos que explican cómo respaldar la información de las aplicaciones en Android, lo que hace que esta vulnerabilidad sea fácil de descubrir.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 16-3** se examina al uso inseguro de un generador de números aleatorios en la aplicación. Los generadores de números aleatorios estándar pueden ser predecibles y, por lo tanto, no son adecuados para resistir ataques criptográficos. Un atacante con el conocimiento necesario podría predecir los números generados y comprometer la seguridad de las funcionalidades que dependen de estos números

Tabla 16-3: Calificación DREAD vulnerabilidad the app uses an insecure Random Number Generator

	Puntaje	**Justificación**
D	2	El atacante puede predecir los números generados y, por lo tanto, puede comprometer la seguridad de las funcionalidades que dependen de estos números, lo que podría llevar a la divulgación de información sensible.
R	3	El ataque es fácilmente reproducible si el atacante tiene el conocimiento necesario para analizar los patrones en los números generados.
E	3	Un programador novato, con conocimientos básicos de criptografía, podría implementar este ataque.
A	2	Afecta a todos los usuarios de la aplicación que utiliza el generador de números inseguro.
D	2	Existen recursos que explican las debilidades de los generadores de números aleatorios estándar, pero puede requerir conocimientos de criptografía para explotarlos.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 17-3** se examina vulnerabilidad se refiere a la transmisión insegura de datos sensibles a través de un protocolo HTTP no seguro en nuestra aplicación móvil. La aplicación está actualmente enviando información delicada de los usuarios, como credenciales, recetas médicas, y detalles personales, sin la debida encriptación durante su transmisión. Un atacante con las herramientas y el conocimiento adecuados podría interceptar estos datos, comprometiendo gravemente la privacidad y la seguridad de nuestros usuarios. Este escenario abre la puerta a una variedad de ataques potenciales, incluyendo suplantación de identidad, fraudes de seguro, extorsión, entre otros.

Tabla 17-3: Calificación DREAD vulnerabilidad transmisión insegura de datos

	Puntaje	Justificación
D	3	La exposición de datos sensibles puede permitir el robo de identidad, fraudes o extorsiones.
R	3	El ataque es fácilmente reproducible ya que depende de una falla inherente en la aplicación y no de condiciones específicas
E	3	Un programador novato podría explotar la vulnerabilidad, ya que requiere cierta habilidad para interceptar y analizar el tráfico de la aplicación.
A	3	Todos los usuarios se ven afectados ya que sus datos personales y médicos están en riesgo.
D	3	La vulnerabilidad es fácilmente descubríble, cualquier atacante con el conocimiento de uso de una herramienta de prueba de intrusión como Burp Suite puede detectarla.

Realizado por: Mesias Francisco, 2023.

En la **Tabla 18-3**, se resume una síntesis clasificatoria de las vulnerabilidades identificadas en la aplicación móvil CMED, basada en el enfoque metodológico DREAD. Esta clasificación proporciona una perspectiva clara sobre la gravedad y el impacto potencial de cada vulnerabilidad detectada.

Tabla 18-3: Síntesis de las vulnerabilidades identificadas

Vulnerabilidad	D	R	E	A	D	Total	Clasificación
Autenticación Biométrica	2	3	2	3	2	12	Alto
android.permission.REQUEST_INSTALL_PACKAGES	3	3	2	3	2	13	Alto
android.permission.READ_EXTERNAL_STORAGE	3	2	2	3	2	12	Alto
Application vulnerable to Janus Vulnerability	3	2	3	2	1	11	Medio
Application signed with debug certificate	1	3	2	3	2	11	Medio
Certificate algorithm vulnerable to hash collision	1	3	2	3	2	11	Medio
App can be installed on a vulnerable Android version	2	3	2	3	2	12	Alto
Application Data can be Backed up	2	3	2	3	2	12	Alto
The App uses an insecure Random Number Generator	2	3	2	3	3	13	Alto
Transmisión insegura de datos sensibles	3	3	3	3	3	15	Alto

Realizado por: Mesias Francisco, 2023.

3.3.5.5 Conclusiones

A partir de las vulnerabilidades identificadas con estas herramientas, se clasificaron de acuerdo con el método DREAD. Se identificaron un total de 10 amenazas en la aplicación examinada, de

las cuales 7 fueron categorizadas como de alto riesgo, 3 como de riesgo medio y ninguna como de bajo riesgo. A continuación, en la **Ilustración 6-3** se representa el número devulnerabilidades y la clasificación.

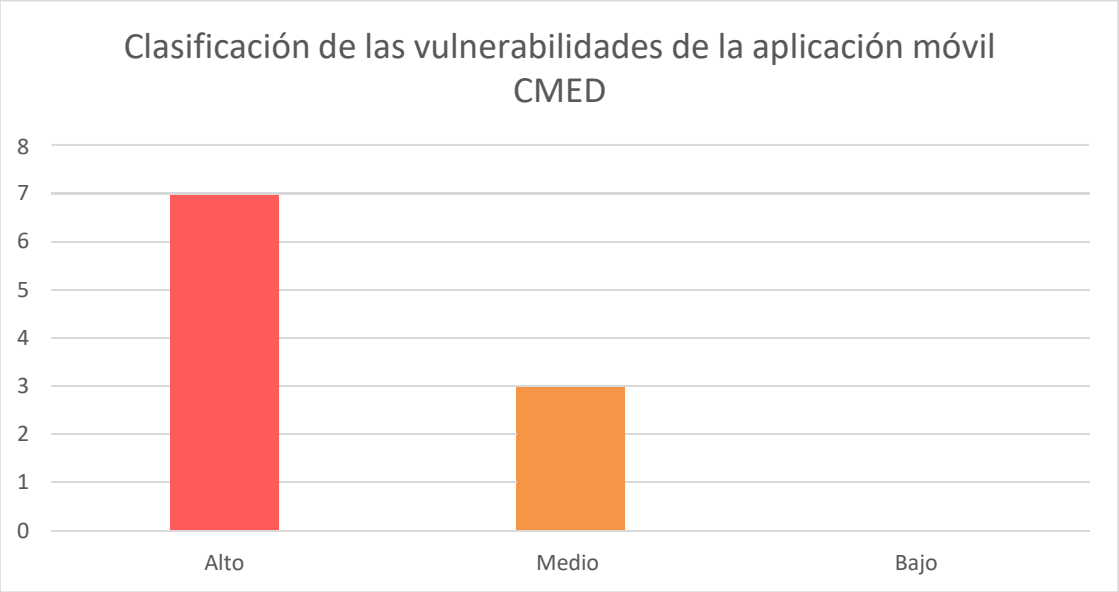


Gráfico 6-3: Numero de vulnerabilidades por clasificación.
Realizado por: Mesias Francisco, 2023

La representación gráfica de barras de las vulnerabilidades identificadas en la aplicación móvil CMED destaca una predominancia de amenazas clasificadas como alto riesgo. Del total de vulnerabilidades detectadas, el 70% (7 amenazas) están clasificadas como de alto riesgo. Estas amenazas, si se explotan, podrían causar un daño considerable en la aplicación, siendo también fácilmente reproducibles y explotables, y afectando a un amplio segmento de usuarios.

Por otro lado, se identificaron 3 amenazas de riesgo medio, que representan el 30% del total de vulnerabilidades encontradas. Aunque estas vulnerabilidades requieren condiciones más específicas para ser explotadas y no tienen un impacto tan generalizado como las de alto riesgo, es fundamental prestarles la debida atención y remediarlas adecuadamente.

3.4 Implementación de buenas practicas

Con el propósito de establecer una metodología efectiva y estructurada para la remediación de las vulnerabilidades identificadas en la aplicación móvil, es fundamental categorizar y entender la naturaleza de cada una de estas fallas. El Estándar de Verificación de Seguridad de Aplicaciones Móviles (MASVS) proporciona un marco de referencia sólido para esto,

definiendo áreas de seguridad específicas que abordan problemas comunes en el desarrollo y operación de aplicaciones móviles.

En la **Tabla 19-3** se presentará una clasificación de las vulnerabilidades detectadas, organizadas de acuerdo con las categorías propuestas por MASVS. Esta clasificación no solo nos permite comprender mejor el espectro de riesgos a los que se enfrenta la aplicación, sino que también sirve como guía para aplicar buenas prácticas específicas, facilitando la corrección y mitigación efectiva de cada vulnerabilidad.

Tabla 19-3: Clasificacion MASVS de las vulnerabilidades encontradas

Clasificación MASVS	**Vulnerabilidad**
MASVS-ALMACENAMIENTO	android.permission.READ_EXTERNAL_STORAGE
	Android permite copia de seguridad
MASVS-CRIPTOGRAFÍA	The App uses an insecure Random Number Generator
	Certificate algorithm vulnerable to hash collision
MASVS-AUTENTICACIÓN Y AUTORIZACIÓN	Autenticación Biométrica
MASVS-RED DE COMUNICACIÓN	Transmisión insegura de datos vulnerables
MASVS-PLATAFORMA	android.permission.REQUEST_INSTALL_PACKAGES
	App can be installed on a vulnerable Android version
MASVS-CÓDIGO	Application signed with debug certificate
	Application vulnerable to Janus Vulnerability
MASVS-RESILENCIA	No hay vulnerabilidades relacionadas

Realizado por: Mesias Francisco, 2023.

Con las vulnerabilidades clasificadas, se procedió a elaborar un plan de remediación detallado basado en las recomendaciones y buenas prácticas establecidas por MASVS y documentación oficial, este plan se centrará en abordar cada vulnerabilidad según su categoría.

3.4.1 MASVS-ALMACENAMIENTO

3.4.1.1 Android.permission_REQUEST_EXTERNAL_STORAGE

El proyecto MASVS enfatiza la importancia de solicitar cada permiso de manera explícita en el momento en que sea requerido por la funcionalidad de la aplicación. Esto debe hacerse en lugar de confiar en la concesión de permisos en grupo. Por otro lado, según la documentación (Huawei developer 2023) se debe declarar permisos en el AndroidManifest.xml. Si un permiso no es necesario para el correcto funcionamiento de la app, se debe considerar retirarlo. Además, para

garantizar una mayor seguridad y transparencia hacia el usuario se recomienda que las aplicaciones soliciten permisos de manera individual, y sólo cuando sean absolutamente esenciales para la tarea que el usuario está realizando en ese momento.

```
1  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:tools="http://schemas.android.com/tools"
3      package="com.example.cmed_app">
4     <!-- CORRECIÓN VULNERABILIDAD -->
5     <uses-permission
6      android:name="android.permission.WRITE_EXTERNAL_STORAGE" tools:node="remove"/>
```

Ilustración 11-3: Android.permission_REQUEST_EXTERNAL_STORAGE implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Con estas recomendaciones se llevó a cabo una revisión minuciosa de todas las dependencias en el archivo pubspec.yaml. Esta revisión tenía como objetivo determinar si alguna dependencia requería el permiso WRITE_EXTERNAL. Una vez que se confirmó que ninguna dependencia lo necesitaba, se decidió retirar este permiso del AndroidManifest.xml utilizando toold:node="remove". Como puede ser verificada en las líneas 5-6 de la **Ilustración 11-3.**

3.4.1.2 Application data can be backed up

Siguiendo las directrices del proyecto MASVS en cuanto a la gestión y almacenamiento de datos confidenciales en aplicaciones móviles, es esencial garantizar que esta información esté adecuadamente protegida y almacenada. Las fugas de datos pueden ocurrir por el uso incorrecto de ciertas APIs o características del sistema, como en el caso de la función de respaldo. Para abordar este riesgo, se sugiere modificar el archivo AndroidManifest.xml, específicamente el atributo android:allowBackup, asignando su valor a "falso". (Android dev 2023).

```
20      <application
21          android:allowBackup="false"
```

Ilustración 12-3: Android permite copias de seguridad implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Siguiendo estas recomendaciones, en la aplicación móvil CMED, se ha revisado y modificado el archivo AndroidManifest.xml como se puede observar en la **Ilustración 12-3**. Se estableció el valor de respaldo en "falso", garantizando así que la información de la aplicación no sea expuesta inapropiadamente

3.4.2 MASVS-CRIPTOGRAFIA

3.4.2.1 The app uses an insecure random number generator

Siguiendo la premisa que MASVS establece sobre la necesidad de una criptografía sólida, la gestión adecuada de números y claves aleatorias es fundamental para la seguridad. Cualquier fallo en esta área puede comprometer la integridad y confidencialidad de los datos, incluso si se emplean algoritmos robustos. De acuerdo (Flutter dev 2023), la función Random.secure() es altamente recomendada por su seguridad criptográfica.

```
557            await AndroidAlarmManager.oneShotAt(
558                DateTime(anio, mes, dia, 11, 59).add(Duration(days: i)),
559                Random.secure().nextInt(pow(2, 31) as int),
```

Ilustración 13-3: Generador de números inseguros implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Como se ilustra en la **Ilustración 13-3**, en la aplicación móvil CMED se ha implementado la función Random.secure(), siguiendo las buenas prácticas recomendadas. Esto asegura que cualquier número aleatorio generado por la aplicación sea adecuado para propósitos criptográficos, contribuyendo a la seguridad global del software.

3.4.2.2 Certificate algorithm vulnerable to hash collision

Dentro del marco de una criptografía bien implementada que MASVS propone, es esencial actualizar y abstenerse de usar algoritmos de hash obsoletos y vulnerables. Es conocido que algoritmos como MD5 y SHA-1 han presentado vulnerabilidades frente a ataques. (Bhargavan y Leurent 2016, pp. 5-6).

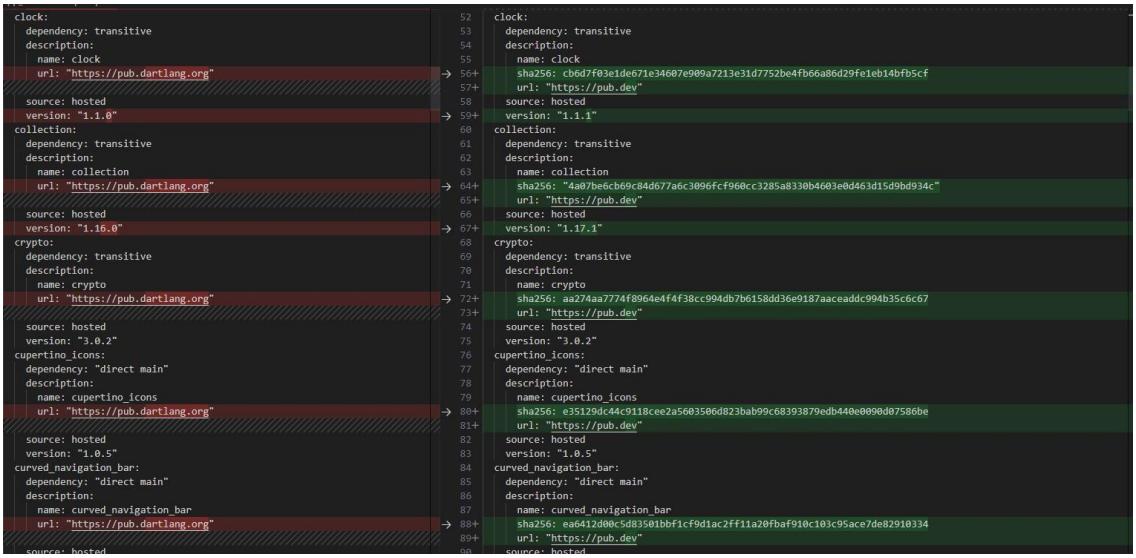


Ilustración 14-3: Aplicación firmada con certificado de depuración implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Como se muestra en la **Ilustración 14-3**, se implementó el algoritmo SHA256withRSA, que ofrece una mayor seguridad en comparación con SHA1. Esta implementación ayuda a proteger la aplicación contra ataques de colisión y garantiza la autenticidad e integridad de esta.

3.4.3 MASVS-AUNTETICACION Y AUTORIZACION

3.4.3.1 Autenticacion Biometrica

```
class Authentication {
  static final _auth = LocalAuthentication();
  static Future<bool> canAuhenticate() async =>
      await _auth.canCheckBiometrics || await _auth.isDeviceSupported();

  static Future<bool> authentication() async {
    try {
      if (!await canAuhenticate()) return false;
      return await _auth.authenticate(localizedReason: "dentro de la app");
    } catch (e) {
      print('error $e');
      return false;
    }
  }
}
```

Ilustración 15-3: Autenticación biométrica de datos implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Como se muestra en la **Ilustración 15-3**, se implementó la autenticación biométrica en la aplicación móvil CMED. Esta implementación se realizó mediante la instalación de paquetes específicos, permitiendo su utilización en el código de la aplicación.

3.4.4 MASVS-RED DE COMUNICACION

3.4.4.1 Transmisicion insegura de datos

Dado que los dispositivos móviles son inherentemente vulnerables debido a su portabilidad, es esencial garantizar que los datos transmitidos entre estos y otros sistemas se realicen de manera segura. MASVS, en su categoría red, subraya la necesidad de que las aplicaciones usen criptografía de acuerdo con las mejores prácticas de la industria, particularmente al tratarse de datos en tránsito. Estándares externos como NIST.SP.800-175B y NIST.SP.800-57 destacan el protocolo TLS (Transport Layer Security) como un estándar de la industria para la transmisión segura.

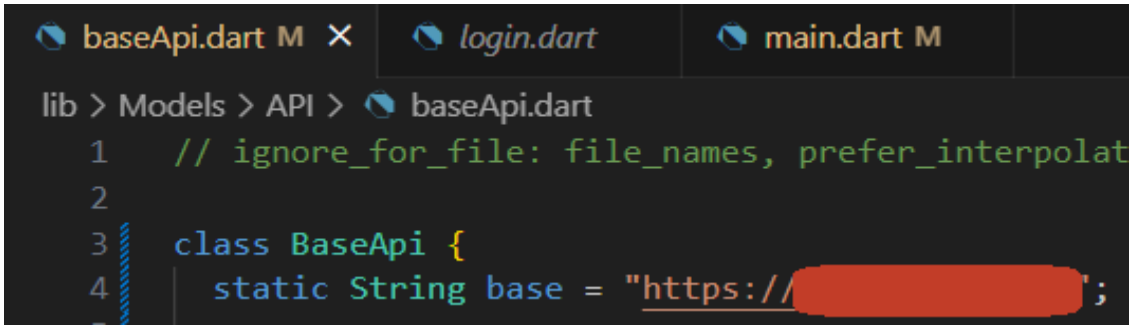


Ilustración 16-3: Transmisión insegura de datos implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

La **Ilustración 16-3** muestra que, en la aplicación móvil CMED, se ha implementado el protocolo HTTPS al conectar con la API, desplegando un servicio en la nube. Esta implementación asegura que la comunicación entre los puntos de la red sea segura y cifrada, alineándose con las recomendaciones de MASVS.

3.4.5 MASVS-PLATAFORMA

3.4.5.1 Android.permission_REQUEST_INSTALL_PACKAGES

En línea con la guía del proyecto MASVS, se recalca la importancia de solicitar permisos de forma explícita según su necesidad. Reforzando este enfoque, la documentación de Huawei Developer (2023) señala que los permisos deben declararse en el AndroidManifest.xml y sugiere

que, si un permiso no es esencial para el funcionamiento de la aplicación, debería considerarse su retirada. En pro de una mayor seguridad y transparencia, es aconsejable que las aplicaciones pidan permisos individualmente, solo cuando son imprescindibles para la acción que el usuario realiza en ese momento.

```
8    <uses-permission
9     android:name="android.permission.REQUEST_INSTALL_PACKAGES" tools:node="remove"/>
```

Ilustración 17-3: Android.permission_REQUEST_INSTALL_PACKAGES implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Siguiendo estas directrices, se realizó un análisis detallado de las dependencias en el archivo pubspec.yaml con el propósito de identificar si alguna requería el permiso INSTALL_PACKAGES. Tras confirmar que no era así, este permiso fue eliminado del AndroidManifest.xml usando toold:node="remove", como se evidencia en las líneas 8-9 de la **Ilustración 17-3**.

3.4.5.2 App can be installed on a vulnerable Android version

Para fortalecer la seguridad de la aplicación, es esencial ajustar las configuraciones de minSdk y declarar la versión objetivo (targetSdkVersion) en el archivo gradle. Esto previene que la aplicación pueda ser instalada en versiones de Android que ya no cuentan con soporte y que podrían estar expuestas a vulnerabilidades conocidas. (Flutter dev 2023)a.

```
defaultConfig {
    // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
    applicationId "com.example.cmed_app"
    // You can update the following values to match your application needs.
    // For more information, see: https://docs.flutter.dev/deployment/android#reviewing-the-build-configuration.
    minSdkVersion 26
    targetSdkVersion 33
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
    multiDexEnabled true
}
```

Ilustración 18-3: App can be installed on a vulnerable Android version implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

La **Ilustración 18-3** muestra la declaración del minSdkVersion, ajustado en consideración a la vulnerabilidad Janus previamente discutida. Se estableció un minSdkVersion de 26 para maximizar el alcance de usuarios compatibles, asegurando que no estén expuestos a la

vulnerabilidad de Janus. Adicionalmente, se definió el targetSdkVersion en 33, correspondiente a Android 13, la versión más reciente de Android disponible.

3.4.6 MASVS-CODIGO

3.4.6.1 Application signed with debug certificate

De acuerdo con la documentación (Flutter dev 2023), es esencial firmar la aplicación utilizando claves de implementación y carga. Esta práctica puede llevarse a cabo a través de herramientas como Android Studio o mediante la línea de comandos. Es más, para maximizar la seguridad, se sugiere la firma automática gestionada en la nube.

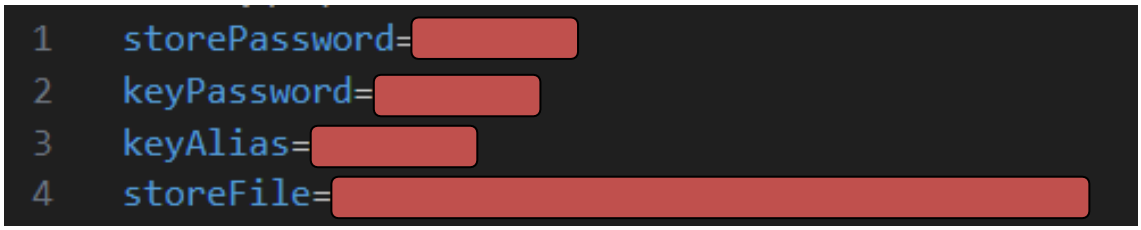


Ilustración 19-3: Aplicación firmada con certificado de depuración implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Como se observa en la **Ilustración 19-3** Se procedió a implementar la firma con un certificado de lanzamiento en lugar de un certificado de depuración. Se genero un almacén de claves (key.properties) a partir de este archivo se procedió a configurar el "signingConfigs" en el archivo "build.gradle".

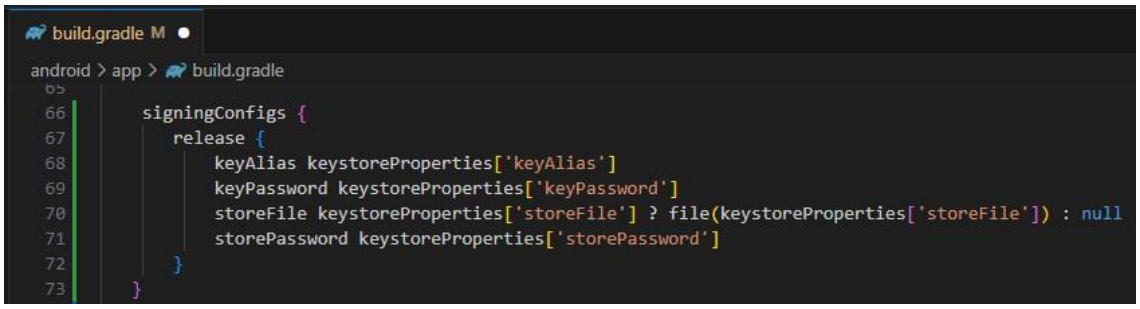


Ilustración 20-3: Aplicación firmada con certificado de depuración implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Como se observa en la **Ilustración 20-3** la parte de implementación del archivo build.gradle toda la información lo recoje del almacen de claves generado anteriormente haciendo que la aplicación ya tenga una firma que no sea de depuración.

3.4.6.2 Application vulnerable to Janus vulnerability

Para abordar esta vulnerabilidad, es crucial limitar la instalación de la aplicación a versiones específicas de Android. Kal (2021) recomienda no solo establecer dichos límites, sino también educar a los usuarios sobre los peligros de utilizar versiones antiguas de Android y fuentes de aplicaciones no confiables.

```
defaultConfig {
    // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
    applicationId "com.example.cmed_app"
    // You can update the following values to match your application needs.
    // For more information, see: https://docs.flutter.dev/deployment/android#reviewing-the-build-configuration.
    minSdkVersion 26
    targetSdkVersion 33
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
    multiDexEnabled true
}
```

Ilustración 21-3: Aplicación Janus vulnerabilidad implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Se limito la instalación de la aplicación movil que se puede observar en la **Ilustración 21-3** a una versión de SDK mayor o igual a 26 lo que siginficia que la aplicación solo se puede intallar en dispositivos Android que poseen Android evitando que se intalen en versiones done la vulnerabilidad de Janus este sin soporte.

CAPÍTULO IV

4 ANALISIS E INTERPRETACION DE LOS RESULTADOS

Este capítulo se centra en la evaluación de los resultados derivados de las pruebas de penetración realizadas a la aplicación móvil CMED. Utilizando el modelo DREAD, se evalúa laseguridad de la aplicación. A través de este análisis, se mide y clasifica cada vulnerabilidad identificada, proporcionando una puntuación total que indica el nivel de riesgo asociado de las dos etapas de pruebas de penetración que se ha realizado a la aplicación. Finalmente, mediante técnicas de estadística inferencial, como el análisis matemático, se busca entender el impacto que ha tenido la corrección de estas vulnerabilidades en la puntuación total de riesgo.

4.1 Análisis descriptivo de la seguridad

Se llevaron a cabo dos etapas de evaluación para obtener los datos de las puntuaciones de las diez vulnerabilidades identificadas en la aplicación móvil. La primera etapa consistió en la evaluación inicial de cada vulnerabilidad utilizando el modelo DREAD para antes de aplicar las correcciones correspondientes. La segunda etapa de evaluación se realizó después de haber implementado las correcciones necesarias a las vulnerabilidades previamente identificadas.

4.1.1 *Evaluación de las vulnerabilidades*

La primera prueba de penetración se realizó antes de las correcciones y la segunda después de las correcciones donde se tomaron la muestra en cada etapa de las pruebas de penetración. En la Tabla 1-4 se detallan los datos obtenidos en las pruebas de penetración, antes y después de las correcciones, y sus respectivas puntuaciones medias.

Tabla 1-4: Clasificacion MASVS de las vulnerabilidades encontradas

Vulnerabilidad	1ra prueba de penetración		2da prueba de penetración	
	Total	Clasificación	Total	Clasificación
Autenticación Biométrica	12	Alto	5	Bajo
android.permission.REQUEST_INSTALL_PACKAGES	13	Alto	5	Bajo
android.permission.READ_EXTERNAL_STORAGE	12	Alto	5	Bajo
Application vulnerable to Janus Vulnerability	11	Medio	5	Bajo
Application signed with debug certificate	11	Medio	5	Bajo

Certificate algorithm vulnerable to hash collision	11	Medio	5	Bajo
App can be installed on a vulnerable Android version	12	Alto	5	Bajo
Application Data can be Backed up	12	Alto	5	Bajo
The App uses an insecure Random Number Generator	13	Alto	5	Bajo
Transmisión insegura de datos vulnerables	15	Alto	5	Bajo
Media	12	Alto	5	Bajo

Realizado por: Mesias Francisco, 2023

Los resultados de las pruebas de penetración permitieron conocer que, en promedio, cada vulnerabilidad tenía una puntuación en promedio de 12 antes de la corrección, mientras que después de la corrección, la puntuación se redujo a 5. Esta reducción representó una disminución del 50% en el riesgo medio de las vulnerabilidades individual de la aplicación móvil CMED.

Para evaluar el impacto que tuvieron las correcciones en el riesgo de las vulnerabilidades, se consideró la puntuación de 15 como el 100% del riesgo. En base a esto, se obtuvo el porcentaje de riesgo que representa la puntuación media antes y después de las correcciones y con la diferencia de estas proporciones se determinó el porcentaje de disminución logrado en el riesgo de cada total de vulnerabilidades.

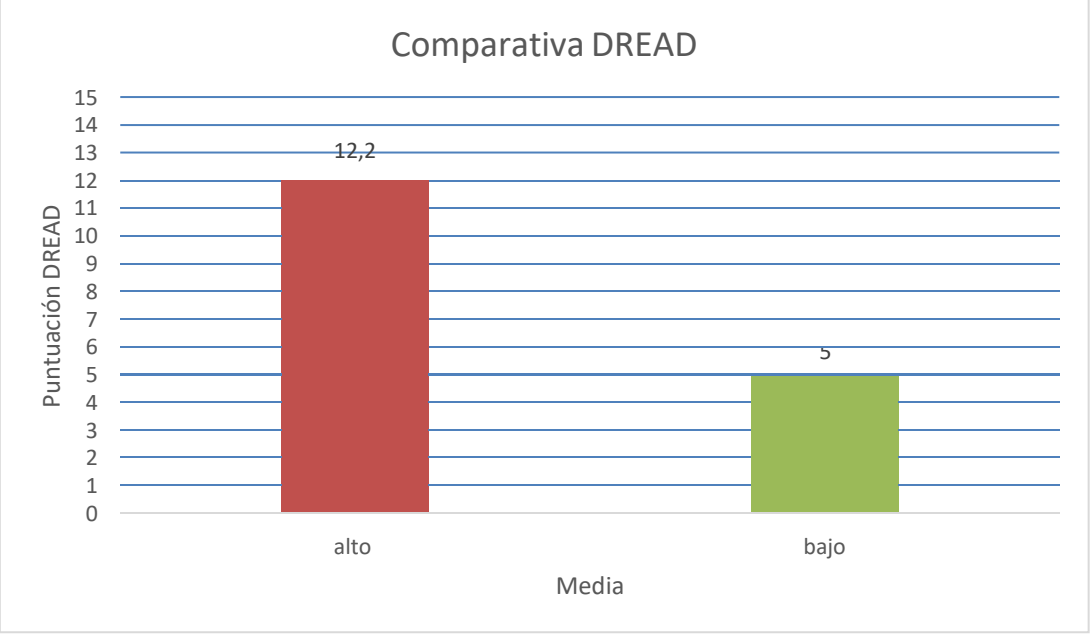


Gráfico 1-4: Numero de vulnerabilidades por clasificación.
Realizado por: Mesias Francisco, 2023

Como se puede observar en el **Grafico 1-4** las vulnerabilidades tenían una puntuación media de 12.2, luego de aplicar las correcciones pertinentes, la puntuación media se redujo a 5, lo que

evidencia un significativo mejoramiento en la seguridad de la aplicación. En términos porcentuales el riesgo inicial representaba aproximadamente un 81.33% del riesgo total. Sin embargo, después de las correcciones, ese riesgo disminuyó al 33.33%. Esto indica una reducción de riesgo del 48% respecto al estado inicial de la aplicación.

4.2 Análisis matemático de la seguridad de la aplicación móvil CMED

Al estudiar las representaciones visuales, como el **Gráfico 1-4**, se destaca de inmediato que las vulnerabilidades, en su estado original, reflejaban una puntuación promedio de 12.2. Posterior a la intervención y aplicación de las correcciones pertinentes, se observó una drástica reducción en esta puntuación, situándose en 5. Este decremento no solo es una manifestación cuantitativa, sino que también resalta un avance significativo en la robustez y fortaleza de la seguridad implementada en la aplicación móvil. Desde una perspectiva aritmética, el riesgo inicial estaba configurado con un porcentaje de 81.33% del total posible. Sin embargo, con las acciones correctivas, este porcentaje descendió al 33.33%. Estos valores, en su comparativa, nos arrojan una contundente reducción del riesgo del 48% en contraposición al estado primordial de la aplicación.

4.3 Conclusión matemática

Las hipótesis planteadas para el proceso de seguridad son:

- $H0$ = La seguridad de la aplicación móvil CMED no experimenta una mejora significativa después de la implementación de las correcciones basadas en las pruebasde pentesting.
- $H1$ = La seguridad de la aplicación móvil CMED experimenta una mejora significativa después de la implementación de las correcciones basadas en las pruebas de pentesting.

Para la validación de estas hipótesis, es esencial efectuar comparaciones aritméticas. Mediante la diferenciación entre los porcentajes medios antes y después de las pruebas de penetración, obtenemos una medida precisa del porcentaje de mejora, como se ilustra en el **Gráfico 2-4**.

1era Prueba de penetración	2da Prueba de penetración	Mejoras en la seguridad
81.33%	33.33%	48%

Gráfico 2-4: Numero de vulnerabilidades por clasificación.
Realizado por: Mesias Francisco, 2023

La evidencia matemática es inequívoca. Existe una mejora significativa en la seguridad de la aplicación móvil CMED, que se traduce en una reducción del 48% en relación con el estado inicial de vulnerabilidades. Por ende, no podemos más que refutar la hipótesis nula (H0) y adoptar la hipótesis alternativa (H1). Esta inferencia matemática robustece la premisa de que la seguridad de la aplicación móvil CMED ha sido fortalecida considerablemente después de accionar las correcciones basadas en las buenas prácticas y en las pruebas de pentesting.

CAPÍTULO V

5 CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Para el análisis estático, MobSF demostró ser superior a Androbugs debido a su accesibilidad para el usuario, la detallada generación de informes y su habilidad paraprocesar tanto archivos binarios como código fuente. En cuanto al análisis dinámico, el estudio favoreció a BurpSuite sobre ZAP. Debido a las mejores capacidades para funcionar como un proxy interceptador y permitir una personalización amplia influyeron en la decisión. En lo referente a las herramientas de inyección SQL, SQLMap se destacó sobre Havij por su habilidad avanzada en la detección de vulnerabilidades y en el empleo de diversas técnicas de explotación. Su alcance y especialización en comparación con Havij fueron considerables.
- Tras aplicar cinco fases de la Guía de Pruebas de Seguridad de Aplicaciones Móviles (MASTG) se encontraron y confirmaron diez vulnerabilidades de nivel alto y medio. Para una valoración más precisa de su gravedad, se aplicó el método de evaluación DREAD. Esta evaluación permitió categorizar las vulnerabilidades en niveles altos y medios, permitiendo una priorización más eficaz de las acciones correctivas.
- El uso del Estándar de Verificación de Seguridad de Aplicaciones Móviles (MASVS) ha permitido clasificar las vulnerabilidades de forma clara y eficaz, proporcionando una guía útil para abordar y corregir cada una de ellas. De esta manera, se han establecido soluciones específicas para cada categoría de vulnerabilidad identificada, lo que ha permitido una gestión y resolución de problemas más eficiente.
- Al llevar a cabo un análisis de penetración y posteriormente implementar las correcciones recomendadas, se ha conseguido una significativa reducción del nivel de riesgo asociado a cada vulnerabilidad. De acuerdo con el modelo DREAD, antes de las correcciones, las vulnerabilidades identificadas en la aplicación presentaban una puntuación promedio de 12.2, lo que representa aproximadamente un 81.33% del riesgo total. Después de la implementación de las correcciones, la puntuación promedio disminuyó a 5, representando ahora un 33.33% del riesgo total. Esto indica una reducción de riesgo del 48% respecto al estado inicial de la aplicación, lo que sugiere que las estrategias de mitigación propuestas fueron efectivas en la mejora general de la seguridad de la aplicación.

5.2 Recomendaciones

- Se recomienda adoptar MobSF para el análisis estático, BurpSuite para el análisis dinámico y SQLMap para pruebas de inyección SQL, dada su eficacia y características superiores identificadas en este estudio.
- Tras utilizar la Guía de Pruebas de Seguridad de Aplicaciones Móviles de OWASP, se recomienda implementar soluciones para las vulnerabilidades identificadas, priorizando las que fueron categorizadas de nivel alto y medio según el método de evaluación DREAD. Particularmente, se deberían abordar los problemas relacionados con los permisos otorgados a la aplicación móvil y la utilización de HTTPS para la transmisión de datos para mejorar la seguridad de la aplicación.
- Considerando la utilidad del Estándar de Verificación de Seguridad de Aplicaciones Móviles para clasificar y abordar las vulnerabilidades, se recomienda seguir utilizando este estándar en futuros proyectos de desarrollo y seguridad de aplicaciones. De esta manera, se puede garantizar un enfoque sistemático y basado en estándares para mejorar la seguridadde las aplicaciones.
- Basándose en la reducción significativa del nivel de riesgo que se logró tras la implementación de las correcciones sugeridas, se recomienda llevar a cabo pruebas de penetración y evaluaciones de vulnerabilidad de manera regular en todas las aplicaciones móviles desarrollada s. Esto permitirá a los desarrolladores identificar y corregir vulnerabilidades de manera oportuna, mejorando la seguridad de la aplicación y reduciendo el riesgo para los usuarios.

GLOSARIO

CMED : Centro de especialidades médicas la “Dolorosa”

MAS: OWASP Mobile Application Security

MASVS: estándar de seguridad para aplicaciones móviles

MASTG: Guía de pruebas exhaustiva

Pentesting: Pruebas de penetración

Pentester: Auditor de seguridad

BIBLIOGRAFÍA

ACOSTA SANTANA, J.J., 2022. Pentesting en entornos controlados [en línea]. 2022. [Consulta: 29 marzo 2023]. Disponible en: https://riull.ull.es/xmlui/handle/915/28744. Aceptado: 06 julio 2022.

ÁLZATE, W.A., ROMAÑA, C.A. y BARCO, Y.A., 2015. Factores y causas de la fuga de información sensibles en el sector empresarial. *Cuaderno activa*. Vol. 7, pp. 67- 73.

ANDROID DEV, 2020a. Storage updates in Android 11. *Android Developers* en línea. 2020. Recuperado a partir de: https://developer.android.com/about/versions/11/privacy/storage [accedido 29 julio 2023].

ANDROID DEV, 2020b. Recomendaciones sobre seguridad de apps | Desarrolladores de Android. *Recomendaciones sobre seguridad de apps* en línea. 2020. Recuperado a partir de: https://developer.android.com/topic/security/best-practices?hl=es-419 [accedido 29 julio 2023].

ANDROID DEV, 2023. Back up user data with Auto Backup. *Copia de seguridad de los datos del usuario con Copia de seguridad automática* en línea. Recuperado a partir de: https://developer.android.com/guide/topics/data/autobackup [accedido 31 julio 2023].

ANTONISHYN, M., 2020. Mobile applications vulnerabilities testing model. *Collection «Information Technology and Security»*. Vol. 8, número 1, pp. 49-57. DOI 10.20535/2411-1031.2020.8.1.218003.

ARGUDO MURILLO, J.A., 2019. Estudio de la afectación a la privacidad de los usuarios que utilizan aplicaciones Android desarrolladas para instituciones públicas del Ecuador en línea. bachelorThesis. Quito, 2019. Recuperado a partir de : http://bibdigital.epn.edu.ec/handle/15000/20159 [accedido 26 julio 2023]. Accepted: 2019-04-02T20:35:16Z

ARKIN, B., STENDER, S. y MCGRAW, G., 2005. Software penetration testing [en línea]. IEEE Security & Privacy. Enero 2005. Vol. 3, no. 1, pp. 84–87. [Consulta: 21 mayo 2023]. DOI 10.1109/MSP.2005.23.

ASAMBLEA CONSTITUYENTE, 2012. Constitución de la República del Ecuador [en línea]. Artículo 66, numeral 19. Quito: Gobierno del Ecuador. [Consulta: 22 mayo 2023]. Disponible en: https://educacion.gob.ec/wp-content/uploads/downloads/2012/08/Constitucion.pdf.

ASHARI, I. F. A., AFFANDI, M., PUTRA, H. T. y NUR, M. T., 2023. Security Audit for Vulnerability Detection and Mitigation of UPT Integrated Laboratory (ILab) ITERA Website Based on OWASP Zed Attack Proxy (ZAP). Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi). Vol. 7, número 1, pp. 24-34. DOI 10.35870/jtik.v7i1.657.

ASTÉLY, A. y EKROTH, J., 2022. Penetration Testing Ten Popular Swedish Android Applications [en línea]. S.l.: KTH Royal Institute of Technology. [Consulta: 22 mayo 2023]. Disponible en: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-320375.

BISHOP, M., 2007. About Penetration Testing [en línea]. IEEE Security & Privacy. Noviembre 2007. Vol. 5, no. 6, pp. 84–87. [Consulta: 20 de mayo 2023]. DOI 10.1109/MSP.2007.159.

BOLIVAR, T. N., DELGADO, M. S. A., FLORES, Y. A. T. y ALMONTE, F. U. R., 2021 Análisis y optimización del proceso de validación de ataques de secuencia de comandos en sitios cruzados (XSS) empleando Burp Suite para evadir medidas de seguridad - ProQuest. [en línea]. Disponible en: https://www.proquest.com/openview/07932f5ab68509ac8b29e51c76926097/1?pq-origsite=gscholar&cbl=1006393 [Consulta: 26 julio 2023].

BORJA SALTOS, T.R., 2021. *Análisis de seguridad de aplicaciones Android en base al top 10 de OWASP 2016*. Online. Quito, 2021. [Accessed 1 May 2023]. Retrieved from: http://bibdigital.epn.edu.ec/handle/15000/21383Accepted: 2021-02-11T17:17:24Z.

BHARGAVAN y LEURENT, 2016. Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH. En: *Proceedings 2016 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society. 2016. ISBN 978-1-891562-41-9. DOI 10.14722/ndss.2016.23418.

CASTILLO ENRÍQUEZ, A.S., HIDALGO GUIJARRO, J.V., y GUANO CÁRDENAS, C.A., 2022. Pruebas de penetración para la seguridad informática al servidor web del laboratorio de ciberseguridad en la Universidad Politécnica Estatal del Carchi, 2021. 2022-07-07. Vol. Vol. 17 Núm. 2 (2022): Revista SATHIRI: Sembrador Vol. 17, N° 2 (Julio-diciembre). DOI https://doi.org/10.32645/13906925.1138.

CHEN, S., SU, T., FAN, L., MENG, G., XUE, M., LIU, Y. y XU, L., 2018. Are mobile banking apps secure? What can be improved? En: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York, NY, USA: Association for Computing Machinery. 26 octubre 2018. pp. 797–802. ISBN 978-1-4503-5573-5. DOI 10.1145/3236024.3275523. [Consulta: 3 abril 2023].

CHIROU, A., 2023. *Seguridad Informática y Cloud* en línea. achirou.com. [Consulta: 24 abril 2023] Recuperado a partir de: https://achirou.com/libro_seguridad.pdf

CIAMPA, A., VISAGGIO, C. A. y DI PENTA, M., 2010. A heuristic-based approach for detecting SQL-injection vulnerabilities in web applications. En: Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, pp. 43-49. New York, NY, USA: Association for Computing Machinery. 2 mayo 2010. SESS '10. ISBN 978-1-60558-965-7. DOI 10.1145/1809100.1809107.

CONKLIN, L., 2023. Threat Modeling Process | OWASP Foundation. en línea. 2023. Recuperado a partir de: https://owasp.org/www-community/Threat_Modeling_Process [accedido 29 julio 2023].

DUCUARA CRUZ, A.Y. y MOYA MOLANO, J.A., 2017. Manual de buenas prácticas sobre la seguridad de la información sensible de la entidad del DANE. [en línea]. Recuperado a partir de: https://alejandria.poligran.edu.co/handle/10823/997 [accedido 10 agosto 2023]. Accepted: 2017-08-04T01:36:52Z

DAMELE, BERNARDO y STAMPAR, MIROSLAV, 2023. sqlmap: automatic SQL injection and database takeover tool. [en línea]. Recuperado a partir de: https://sqlmap.org/ [accedido 10 agosto 2023].

DEHAVEN, K., 2019. Mobile Application Vulnerabilities and Threats [en línea]. S.l.: PT Security. p. 9. [Consulta: 22 mayo 2023]. Disponible en: https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Mobile-Application-Vulnerabilities-and-Threats-2019-eng.pdf.

DELÍA, L., GALDÁMEZ, N., THOMAS, P. y PESADO, P., 2013. Un análisis experimental de tipo de aplicaciones para dispositivos móviles. En: XVIII Congreso Argentino de Ciencias de la Computación en línea. Octubre 2013. Recuperado a partir de: http://sedici.unlp.edu.ar/handle/10915/32397 [Consulta: 23 abril 2023].

DOSS, C., YUAN, X., CHENNAKESHVA, V., ABERNATHY, A. y FORD, K., 2019. Modules for Teaching Secure in Android Application Development. Journal of The Colloquium for Information Systems Security Education. Vol. 6, número 2, pp. 1-13.

EKENBLAD, J. y STEFAN, A.G., 2022. *Security evaluation of ten Swedish mobile applications* en línea. Recuperado a partir de : https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-319794 [accedido 29 julio 2023].

ENISA, 2020. Data Breach [en línea]. S.l.: ENISA, p. 4. [Consulta: 22 mayo 2023]. Disponible en: https://www.enisa.europa.eu/publications/report-files/ETL-translations/es/etl2020-data-breach-ebook-en-es.pdf.

FIGUEIREDO, D.A., MARQUES, R.M., MARTINS, H.C., y ARAMUNI, J.P., 2020. Vulnerabilidades em redes Wi-Fi de instituições de ensino superior: um estudo de múltiplos casos. Research, Society and Development. Vol. 9, número 2, pp. e178921979-e178921979. DOI 10.33448/rsd-v9i2.1979.

FLUTTER DEV, 2019a. FAQ [en línea]. [Consulta: 24 abril 2023]. Disponible en: https://esflutter.dev/docs/resources/faq.

FLUTTER DEV, 2019b. Platform channels. [en línea]. [Consulta: 24 abril 2023]. Disponible en: https://esflutter.dev/docs/development/platform-integration/platform-channels.

FLUTTER DEV, 2023a. Random.secure constructor - Random - dart:math library - Dart API. Random.secure constructor en línea. 12 julio 2023. Recuperado a partir de: https://api.flutter.dev/flutter/dart-math/Random/Random.secure.html [accedido 31 julio 2023].

FLUTTER DEV, 2023b. website/src/reference/security-false-positives.md at main · flutter/website. GitHub en línea. 2023. Recuperado a partir de: https://github.com/flutter/website/blob/main/src/reference/security-false-positives.md [accedido 29 julio 2023].

GARG, S. y BALIYAN, N., 2021. Comparative analysis of Android and iOS from security viewpoint. *Computer Science Review*. Vol. 40, p. 100372. DOI 10.1016/j.cosrev.2021.100372.

GAMIDO, HEIDILYN y GAMIDO, MARLON, 2019. Comparative Review of the Features of Automated Software Testing Tools. International Journal of Electrical and Computer Engineering. Vol. 9, pp. 4473-4478. DOI 10.11591/ijece.v9i5.pp4473-4478.

GEER, D. y HARTHORNE, J., 2002. Penetration testing: a duet. En: 18th Annual Computer Security Applications Conference, 2002. Proceedings. Diciembre 2002. pp. 185–195. DOI 10.1109/CSAC.2002.1176290.

GÓMEZ ZÚÑIGA, J.G. y SUQUILANDA RODRÍGUEZ, Á.A., 2022. Aplicación de una metodología de seguridad para el desarrollo seguro de aplicaciones móviles de pagos en línea usando herramientas OPEN SOURCE. en línea. Thesis . Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas. Carrera de Ingeniería en Networking y Telecomunicaciones. Recuperado a partir de: http://repositorio.ug.edu.ec/handle/redug/59789 [accedido 26 julio 2023]. Accepted: 2022-05-15T00:40:31Z

GOOGLE PLAY DEVELOPER, 2023. Uso del permiso REQUEST_INSTALL_PACKAGES - Ayuda de Play Console. en línea. 2023. Recuperado a partir de: https://support.google.com/googleplay/android-developer/answer/12085295?hl=es-419 [accedido 29 julio 2023].

HE, D., NAVEED, M., GUNTER, C.A. y NAHRSTEDT, K., 2014. Security Concerns in Android mHealth Apps. AMIA Annu Symp Proc [en línea], vol. 2014, pp. 645-654. [Consulta: 22 mayo 2023]. PMC4419898. PMID: 25954370. Disponible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4419898/pdf/1969977.pdf.

HUAWEI DEV, 2023. Adding Permissions-Development Preparations-Android Phone App Development-Wear Engine. *Adding Permissions* en línea. 17 julio 2023. Recuperado a partirde : https://developer.huawei.com/consumer/en/doc/development/connectivity-Guides/adding-permissions-0000001064141094 [accedido 31 julio 2023].

IBM CORPORATION, 2022. Mobile security. [en línea]. [Consulta: 6 mayo 2023]. Disponible en: https://www.ibm.com/es-es/topics/mobile-security.

INTERPOL, 2020. Los ciberdelincuentes secuestran datos de las instituciones esenciales de atención médica. [en línea]. [Consulta: 6 mayo 2023]. Disponible en: https://www.interpol.int/es/Noticias-y-acontecimientos/Noticias/2020/Los-ciberdelincuentes-secuestran-datos-de-las-instituciones-esenciales-de-atencion-medica.

KAL, 2021. Exploiting Apps vulnerable to Janus (CVE-2017–13156). *Mobis3c* [en línea]. [consulta: 12 junio 2023]. Disponible en: https://medium.com/mobis3c/exploiting-apps-vulnerable-to-janus-cve-2017-13156-8d52c983b4e0.

KNORR, K. y ASPINALL, D., 2015. Security testing for Android mHealth apps. In: *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 1–8. April 2015. DOI 10.1109/ICSTW.2015.7107459.

LAKE, 2018. Picking your compileSdkVersion, minSdkVersion, targetSdkVersion. *Android Developers* en línea. 24 agosto 2018. Recuperado a partir de : https://medium.com/androiddevelopers/picking-your-compilesdkversion-minsdkversion-targetsdkversion-a098a0341ebd [accedido 29 julio 2023].

LIN, Yu-Cheng, 2023. *AndroBugs Framework* [logiciel] en línea. 24 julio 2023. [accedido 26 julio 2023]. Recuperado a partir de: https://github.com/AndroBugs/AndroBugs_Framework [accedido 26 julio 2023].

MARTORELL, R., y HUACARÁN, F., 2013. EL VALOR DE LA INFORMACIÓN CIUDADANA. SEGURIDAD INFORMÁTICA Y GESTIÓN MUNICIPAL. *Revista Iberoamericana de Estudios Municipales; No. 7 (2013); 10-26.* DOI 10.32457/riem.vi7.400. Accepted: 2022-10-28T02:00:11Z

MAINKA, C., MLADENOV, V., GUENTHER, T. y SCHWENK, J., 2015. Automatic recognition, processing and attacking of single sign-on protocols with burp suite. Gesellschaft für Informatik e.V. [en línea]. ISBN 978-3-88579-645-9. Disponible en: http://dl.gi.de/handle/20.500.12116/1977. [Consulta: 3 abril 2023].

MD ZAUR REZA, J., 2023. *Havij* [logiciel] en línea. 24 julio 2023. Recuperado a partir de: https://github.com/rezaJOY/Havij [accedido 27 julio 2023].

MICROSOFT, 2018. Documento informativo sobre seguridad de Microsoft 2880823. *Documento informativo sobre seguridad de Microsoft 2880823* en línea. 29 noviembre 2018. Recuperado a partir de: https://learn.microsoft.com/es-es/security-updates/securityinformationdocuments/2013/2880823 [accedido 29 julio 2023].

MICROSOFT, 2023a. Threat modeling for drivers - Windows drivers. en línea. 1 marzo 2023. Recuperado a partir de: https://learn.microsoft.com/en-us/windows-hardware/drivers/driversecurity/threat-modeling-for-drivers [accedido 28 julio 2023].

MICROSOFT, 2023b. Reglas de seguridad (análisis de código) - .NET. *Reglas de seguridad* en línea. 10 mayo 2023. Recuperado a partir de: https://learn.microsoft.com/es-es/dotnet/fundamentals/code-analysis/quality-rules/security-warnings [accedido 29 julio 2023].

MOBILE SECURITY FRAMEWORK, 2021. Mobile Security Framework - MobSF Documentation. Mobile Security Framework - MobSF Documentation en línea. 2021. Recuperado a partir de: https://mobsf.github.io/docs/ [accedido 26 julio 2023].

MUELLER, B., SCHLEIER, S., WILLEMSEN, J., y HOLGUERA, C., 2023. OWASP Mobile Application Security: MASTG. Version v1.6.0 [en línea]. [Consulta: 20 mayo 2023]. Disponible en: https://github.com/OWASP/owasp-mastg/releases/latest/download/OWASP_MASTG.pdf.

NAGPAL, B., SINGH, N., CHAUHAN, N. y PANESAR, A., 2015. Tool based implementation of SQL injection for penetration testing. En: Communication & Automation International Conference on Computing, pp. 746-749. mayo 2015. DOI 10.1109/CCAA.2015.7148509.

NAPOLI, M.L., 2019. Beginning Flutter: A Hands On Guide to App Development. Indianapolis, IN: s.n. ISBN 978-1-119-55082-2.

NURINDAHSARI, F. y ZEN, B.P., 2021. ANALISIS STATIK KEAMANAN APLIKASI VIDEO STREAMING BERBASIS ANDROID MENGGUNAKAN MOBILE SECURITY FRAMEWORK (MOBSF). Cyber Security dan Forensik Digital. Vol. 4, no. 2, pp. 63–80. DOI 10.14421/csecurity.2021.4.2.3373.

OJAGBULE, O., WIMMER, H. y HADDAD, R. J., 2018. Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP. En: SoutheastCon 2018, pp. 1-7. abril 2018. DOI 10.1109/SECON.2018.8479130.

OLIVARES, E. y EDUARDO, G., 2017. Driving directions, live traffic & road conditions updates. *Modelo de amenazas, una ténica de análisis y gestión de riesgo asociados a software y aplicaciones* en línea. Recuperado a partir de: https://wwww.waze.com/live-map/directions?zoom=15&lat=4.63208&lon=-74.06539&pin=1 [accedido 30 julio 2023].

QUISAGUANO, L. R., PALLASCO, M. S., ANDALUZ, A. A., MARTÍNEZ, M. N., y CORRALES, S. H., 2022. Desarrollo Hibrido con Flutter. Revista Latina, 6(4). https://doi.org/10.37811/cl_rcm.v6i4.2959.

RIERA, J.,2023. Implementación de una aplicación móvil nativa para el control de prescripciones médicas del Centro Médico de Especialidades "La Dolorosa". Escuela Superior Politécnica de Chimborazo [Consulta: 23 mayo 2023].

ROMERO CASTRO, M.I., FIGUEROA MORÁN, G.L., VERA NAVARRETE, D.S., ÁLAVA CRUZATTY, J.E., PARRALES ANZÚLES, G.R., ÁLAVA MERO, C.J., MURILLO QUIMIZ, Á.L. y CASTILLO MERINO, M.A., 2018. Introducción a la seguridad informática y el análisis de vulnerabilidades [en línea]. 1. S.l.: Editorial Científica 3Ciencias. [Consulta: 13 mayo 2023]. ISBN 978-84-949306-1-4. Disponible en: https://www.3ciencias.com/wp-content/uploads/2018/10/Seguridad-inform%C3%A1tica.pdf.

SCHLEIER, S., HOLGUERA, C., BECKERS, J., MÜLLER, B. y WILLEMSEN, J., 2023. Release v2.0.0 · OWASP/owasp-masvs. [en línea]. Disponible en: https://github.com/OWASP/owasp-masvs/releases/tag/v2.0.0 [Consulta: 27 julio 2023].

THE OWASP FOUNDATION, 2020. Web Security Testing Guide. 4.2. S.l.: The OWASP Foundation.

THOMAS, P.J., DELÍA, L.N., CORBALÁN, L.C., CÁSERES, G., FERNÁNDEZ SOSA, J., TESONE, F., CUITIÑO, A. y PESADO, P.M., 2018. Tendencias en el desarrollo de aplicaciones para dispositivos móviles [en línea]. En: XX Workshop de Investigadores enCiencias de la Computación (WICC 2018, Universidad Nacional del Nordeste). 2018. ISBN 978-987-3619-27-4. Disponible en: http://sedici.unlp.edu.ar/handle/10915/67726. [Consulta: 28 abril 2023].

VARGAS BIANCHI, L., 2003. Procesamiento de información y familiaridad de marca. [en línea]. [consulta: 10 agosto 2023]. Disponible en: https://www.raco.cat/index.php/Analisi/article/view/15147.

WILSON, A., 2019. LO ESENCIAL DEL HACKEO: la guía para principiantes sobre hackeo ético y pruebas de penetración. [en línea]. ADIDAS WILSON. ISBN 978-1071508701.

ZED ATTACK PROXY, 2023. OWASP ZAP – Getting Started. en línea. 2023. Recuperado a partir de: https://www.zaproxy.org/docs/ [accedido 27 julio 2023].

ZHANG, L., TAAL, A., CUSHING, R., DE LAAT, C. y GROSSO, P., 2022. A risk-level assessment system based on the STRIDE/DREAD model for digital data marketplaces. International Journal of Information Security. Vol. 21, número 3, pp. 509-525. DOI 10.1007/s10207-021-00566-3.

ANEXO A: MANUAL TÉCNICO

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA SOFTWARE

ANÁLISIS DE VULNERABILIDADES MEDIANTE PENTESTING A LA APLICACIÓN MÓVIL DESARROLLADA PARA EL CENTRO MÉDICO DE ESPECIALIDADES “LA DOLOROSA”

MANUAL TÉCNICO

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO DE SOFTWARE

AUTOR

FRANCISCO XAVIER MESIAS FLORES

Riobamba – Ecuador

2023

1. Estudio de Factibilidad

Este estudio tiene como objetivo el determinar aspectos técnicos, operacionales y económicos con el objetivo de analizar la viabilidad del proyecto.

1.1. Factibilidad Técnica

La factibilidad técnica tiene el propósito de identificar la viabilidad del proyecto mediante un análisis de los recursos que conforman la realización del proyecto como pueden ser recursos hardware, software y humanos. La **Tabla 1-1** muestra los recursos hardware que son usados en el desarrollo del proyecto.

Tabla 1-1: Recursos Hardware existentes

Cantidad	Descripción	Observación
1	Laptop Asus ROG ZEPHYRUS Procesador: i7-12700H RAM: 16GB RAM, Disco: SSD 1.5 TB Tarjeta Gráfica: RTX 3060	Necesaria para realizar las pruebas de penetración.

Realizado por: Francisco Mesias 2023

A continuación, en la Tabla 1-2 se observa los recursos de software existentes.

Tabla 2-1: Recursos Software existentes

Cantidad	Descripción	Observación
1	Sistema Operativo Windows 11	Sistema operativo de paga
1	Visual Studio Code	Software Libre

Realizado por: Francisco Mesias 2023

En la **Tabla 3-1** se observa los recursos de software requeridos para realizar el trabajo de integración curricular.

Tabla 2-1: Recursos Software requeridos

Cantidad	Descripción	Observación
1	MobSF	Software Libre
1	BurpSuite	Software de Paga/Software Community
1	SQLMap	Software Libre
1	Railway	Software de Paga

Realizado por: Francisco Mesias 2023

En la tabla 4-1 se puede evidenciar los recursos humanos presentes para realizar las pruebas de penetración que se requieren para la identificación de vulnerabilidades.

Tabla 4-1: Recursos Humanos requeridos

Cantidad	Función	Descripción	Observación
1	Pentester	Estudios de tercer nivel	Analista de seguridad de aplicaciones

Realizado por: Francisco Mesias 2023

1.2. Factibilidad Económica

Con la factibilidad económica se busca obtener la viabilidad que tendrá el desarrollo de este proyecto frente a los costos técnicos de hardware, software y otras necesidades al desarrollo del proyecto. La **Tabla 5-1** muestra los costos de desarrollo para la realización del trabajo de integración curricular.

Tabla 5-1: Costos del desarrollo del proyecto

Detalle	Cantidad	Valor Unitario	Costo Total
Laptop Asus ROG ZEPHYRUS Procesador: i7-12700H RAM: 16GB RAM, Disco: SSD 1.5 TB Tarjeta Gráfica: RTX 3060	1	$ 1,800.00	$ 1,800.00
Sistema Operativo Windows 11	1	$ 79.99	$ 79.99
COSTO TOTAL			$ 1,879.9

Realizado por: Francisco Mesias, 2023.

En la **Tabla 6-1** se indican los costos de los recursos humanos necesarios para el trabajo de integración curricular.

Tabla 6-1: Costos del desarrollo del proyecto

Detalle	Personal	Mes	Valor Unitario	Costo Total
Pentester	1	4	$ 1,000	$ 4,000
COSTO TOTAL				$ 4,000

Realizado por: Francisco Mesias, 2023.

1.3. Factibilidad Operativa

El desarrollo de este trabajo de integración curricular dará como resultado la corrección e implementación de medidas de seguridad para la aplicación móvil del centro médico “La Dolorosa” mediante pentesting, influyendo en la seguridad de la aplicación debido a que maneja información sensible de los pacientes. Esto demuestra que el desarrollo de este trabajo tiene factibilidad operacional, ya que, se mejorara el nivel de seguridad de la aplicación móvil.

2. ANÁLISIS DE RIESGO

Para cada peligro detectado en el avance del proyecto, hay un rango de posibilidad de incidencia que varía entre el 10% y el 100%, aludiendo a la chance de que dicho peligro ocurra.

La **Tabla 1-2** muestra las métricas para la categorización de los riesgos identificados.

Tabla 1-2: Rango de métricas

Probabilidad	**Porcentaje**
Baja	10%-30%
Media	31%-50%
Alta	51%-100%

Realizado por: Francisco Mesias 2023

En la **Tabla 2-2** refleja las cifras relacionadas con el efecto que cada peligro podría causar en el progreso del proyecto, teniendo en cuenta el lapso de demora que generaría en el itinerario de construcción del producto.

Tabla 2-2. Impacto de progreso

Impacto	**Impacto Técnico**	**Tiempo**	**Valor**
Bajo	Leve	Máximo 1 semana	1
Medio	Moderado	2 a 3 semanas	2
Alto	Critico	Mayor a 3 semanas	3

Realizado por: Francisco Mesias 2023

2.1. Análisis cuantitativo de riesgos y priorización

La clasificación de riesgos facilitará la obtención del valor de la variable de exposición al peligro, con el propósito de asignar una prioridad de riesgo, tal como se indica en la **Tabla 3-2**. Dicha variable resulta de multiplicar la posibilidad de incidencia con el valor temporal.

Tabla 3-2. Clasificación de riesgos

Rango de valores	**Prioridad**	**Color**
0.1 hasta 0.4	Baja	Amarillo
0.41 hasta 1.5	Media	Naranja
1.51 en adelante	Alta	Rojo

Realizado por: Francisco Mesias 2023

Tomando como referencia la **Tabla 3-2**, se avanza en la realización de análisis cuantitativo de cada peligro detectado, dicho procedimiento se ilustra en la **Tabla 4-2**.

Tabla 4-2. Análisis cuantitativo de los riesgos detectados

ID	Probabilidad		Impacto			Resultado	Prioridad
	Probabilidad	Valor	Impacto	Semanas	Valor		

RG01	Media	31%	Media	1.5	2	0.41	Baja
RG02	Baja	15%	Media	2	2	1.8	Alta
RG03	Baja	15%	Media	1.5	2	1.7	Alta
RG04	Baja	20%	Baja	1	1	0.3	Baja
RG05	Baja	17%	Baja	1	1	0.8	Media
RG06	Baja	18%	Baja	1	1	0.9	Media
RG07	Baja	18%	Baja	1	1	0.3	Baja

Realizado por: Francisco Mesias 2023

Con los resultados obtenidos que se pueden observar en la **Tabla 4-2**, se obtiene que existen dos riesgos de alta prioridad las cuales son RG02 y RG03. Para la prioridad media se obtiene dos riesgos que se representan con el identificador RG05 y RG06. Finalmente, para la prioridad Baja se obtienen 2 identificadores RG01, RG04 y RG01.

2.2. Hoja de gestión de riegos

En esta sección se encuentran las hojas de gestión de riesgos permitiendo realizar el seguimiento, control y el reporte de los riesgos identificados para este proyecto.

Tabla 5-2. Hoja de gestión de riesgos RG01

Riesgo RG01		
Identificador del riesgo: RG01		Fecha: 3/04/2023
Probabilidad: Media	Impacto: Media	Prioridad: Baja
Descripción: Fallos de seguridad no detectados durante el pentesting		
Refinamiento: • Causas 1. Falta de experiencia con las herramientas de pentesting utilizadas 2. Complejidad o falta de documentación de la aplicación móvil. • Consecuencias 1. Vulnerabilidades no detectadas pueden ser explotadas en el futuro. 2. Retrasos en la planificación debido a la necesidad de repetir el pentesting.		
Reducción: • Entrenamiento exhaustivo en las herramientas de pentesting. • Utilizar una metodología de pentesting establecida como OWASP.		
Supervisión • Revisiones continuas de los resultados del pentesting.		
Gestión • Realización de pruebas de pentesting. • Mantenimiento de un registro detallado de todas las pruebas realizadas y sus resultados.		
Estado Actual: ▪ Fase de reducción iniciada ▪ Fase de supervisión iniciada ▪ Gestionando el riesgo		
Encargados: • Francisco Mesias		

Realizado por: Francisco Mesias 2023

Tabla 6-2. Hoja de gestión de riesgos RG02

Riesgo RG02		
Identificador del riesgo: RG02		Fecha: 3/04/2023
Probabilidad: Baja	Impacto: Media	Prioridad: Alta

Descripcion: Falla en la corrección de vulnerabilidades identificadas
Refinamiento: • Causas 1. Falta de conocimiento o experiencia en la corrección de ciertas vulnerabilidades. 2. Limitaciones de la plataforma de desarrollo • Consecuencias 1. Las vulnerabilidades al volverlas al pasar por la herramienta siguen existiendo. 2. La corrección de la vulnerabilidad puede introducir nuevas vulnerabilidades.
Reducción: • Adquirir formación adicional en la corrección de vulnerabilidades. • Consultar la documentación de Flutter y buscar ayuda de la comunidad si es necesario.
Supervisión • Revisiones regulares del código y nuevas pruebas de pentesting después de cada corrección de vulnerabilidad.
Gestión • Mantenimiento de un registro detallado de todas las correcciones realizadas. • Pruebas exhaustivas de la funcionalidad de la aplicación después de cada corrección.
Estado Actual: ▪ Fase de reducción iniciada ▪ Fase de supervisión iniciada ▪ Gestionando el riesgo
Encargados: • Francisco Mesias

Realizado por: Francisco Mesias 2023

Tabla 7-2. Hoja de gestión de riesgos RG03

Riesgo RG03		
Identificador del riesgo: RG03		Fecha: 3/04/2023
Probabilidad: Baja	Impacto: Media	Prioridad: Alta
Descripcion: Riesgo de herramientas de pentesting ineficaces		
Refinamiento: • Causas 1. Elección de herramientas de pentesting inadecuadas. 2. Uso incorrecto de las herramientas de pentesting • Consecuencias 1. Resultados de pentesting inexactos o engañosos. 2. Posibles vulnerabilidades sin detectar.		
Reducción: • Investigación y selección cuidadosa de las herramientas de pentesting • Capacitación adecuada en el uso de las herramientas de pentesting		
Supervisión • Monitoreo constante de las pruebas de pentesting • Revisión regular de las actualizaciones y mejoras de las herramientas de pentesting		
Gestión • Asegurar una respuesta rápida y una revisión de las herramientas si se detecta una ineficacia		
Estado Actual: ▪ Fase de reducción iniciada ▪ Fase de supervisión iniciada ▪ Gestionando el riesgo		
Encargados: • Francisco Mesias		

Realizado por: Mesias, Francisco 2023

Tabla 8-2. Hoja de gestión de riesgos RG04

Riesgo RG04		
Identificador del riesgo: RG04		Fecha: 3/04/2023
Probabilidad: Baja	Impacto: Baja	Prioridad: Baja

Descripción: No realizar pruebas de pentesting suficientes
Refinamiento: • Causas 1. Falta de tiempo o recursos para realizar pruebas suficientes. 2. Subestimar la necesidad de pruebas exhaustivas. • Consecuencias 1. Vulnerabilidades no detectadas. 2. La aplicación queda expuesta a posibles ataques.
Reducción: • Planificación adecuada para permitir pruebas de pentesting suficientes • Comprender la importancia de las pruebas exhaustivas y asignar recursos en consecuencia
Supervisión • Seguimiento constante del progreso de las pruebas de pentesting • Revisión regular de los resultados de las pruebas
Gestión • Asegurar una rápida reacción en caso de que las pruebas sean insuficientes
Estado Actual: ▪ Fase de reducción iniciada ▪ Fase de supervisión iniciada ▪ Gestionando el riesgo
Encargados: • Francisco Mesias

Realizado por: Mesias, Francisco 2023

Tabla 9-2. Hoja de gestión de riesgos RG05

Riesgo RG05		
Identificador del riesgo: RG05		Fecha: 3/04/2023
Probabilidad: Baja	Impacto: Baja	Prioridad: Media
Descripción: Riesgo de falsos positivos en las pruebas de pentesting		
Refinamiento: • Causas 1. Herramientas de pentesting ineficaces o mal configuradas. 2. Interpretación incorrecta de los resultados de las pruebas • Consecuencias 1. Pérdida de tiempo y recursos en la investigación de falsos positivos 2. Posible desviación del enfoque de las verdaderas vulnerabilidades		
Reducción: • Uso correcto y eficaz de las herramientas de pentesting. • Capacitación adecuada en la interpretación de los resultados de las pruebas.		
Supervisión • Revisión cuidadosa de los resultados de las pruebas de pentesting. • Confirmación de los resultados antes de tomar medidas.		
Gestión • Rápida reevaluación y ajuste de las estrategias de pruebas si se detectan falsos positivos		
Estado Actual: ▪ Fase de reducción iniciada ▪ Fase de supervisión iniciada ▪ Gestionando el riesgo		
Encargados: • Francisco Mesias		

Realizado por: Mesias, Francisco 2023

Tabla 10-2. Hoja de gestión de riesgos RG06

Riesgo RG06		
Identificador del riesgo: RG06		Fecha: 3/04/2023
Probabilidad: Baja	Impacto: Baja	Prioridad: Media
Descripción: Riesgo de no mantenerse al día con las nuevas amenazas de seguridad		

Refinamiento: • Causas 1. Falta de tiempo o recursos para la investigación continua. 2. Subestimar la rapidez con la que evolucionan las amenazas de seguridad • Consecuencias 1. La aplicación puede quedar expuesta a nuevas amenazas 2. Posibles brechas de seguridad y pérdida de datos
Reducción: • Asignar tiempo y recursos para mantenerse al día con las últimas amenazas de seguridad. • Comprender la importancia de la evolución constante en el campo de la seguridad informática
Supervisión • Mantener una vigilancia constante de las últimas noticias y actualizaciones en seguridad informática • Revisar regularmente las estrategias de seguridad y las herramientas utilizadas
Gestión • Rápida reevaluación y ajuste de las estrategias de pruebas si se detectan falsos positivos
Estado Actual: ▪ Fase de reducción iniciada ▪ Fase de supervisión iniciada ▪ Gestionando el riesgo
Encargados: • Francisco Mesias

Realizado por: Mesias, Francisco 2023

Tabla 11-2. Hoja de gestión de riesgos RG07

Riesgo RG07		
Identificador del riesgo: RG07		Fecha: 3/04/2023
Probabilidad: Baja	Impacto: Media	Prioridad: Media
Descripción: Riesgo de no abordar adecuadamente todas las vulnerabilidades		
Refinamiento: • Causas 1. Falta de tiempo o recursos para analizar y solucionar todas las vulnerabilidades encontradas. 2. Interpretación incorrecta del informe de vulnerabilidades • Consecuencias 1. La aplicación sigue siendo susceptible a ataques y brechas de seguridad 2. Pérdida de confianza por parte de los usuarios y posibles sanciones legales		
Reducción: • Asignar tiempo y recursos adecuados para abordar las vulnerabilidades encontradas. • Capacitación adecuada en la interpretación de informes de vulnerabilidades y técnicas de remediación		
Supervisión • Seguimiento de las vulnerabilidades solucionadas y pendientes • Revisión periódica de las medidas de seguridad implementadas		
Gestión • Establecer un plan de acción para abordar las vulnerabilidades encontradas y garantizar su ejecución • Evaluar periódicamente el estado de seguridad de la aplicación y adaptar las estrategias en consecuencia		
Estado Actual: ▪ Fase de reducción iniciada ▪ Fase de supervisión iniciada ▪ Gestionando el riesgo		
Encargados: • Francisco Mesias		

Realizado por: Mesias, Francisco 2023

3. PLANIFICACIÓN DEL PROYECTO

En este apartado se delineará la etapa de preparación del proyecto centrado en la integración curricular. El objetivo es determinar el/los participantes encargados de realizar las pruebas de penetración y el análisis vulnerabilidades a la aplicación móvil del centro médico la “Dolorosa”.

3.1. Roles de Estudio

En la tabla 1-3 se observa el encargado que va a realizar el análisis de vulnerabilidad y las pruebas de penetración.

Tabla 1-3. Roles de Estudio

Integrante	Rol	Contacto
Francisco Xavier Mesias Flores	Pentester	f.mesias99@gmail.com

Realizado por: Francisco Mesias 2023

3.2. Cronograma de actividades

En el apartado siguiente se detallan las fases del proyecto, organizadas según un programa de actividades. Estas actividades corresponden a las distintas etapas de la tesis y se han llevado a cabo en el periodo que se muestra en la **Ilustración 1-3**.

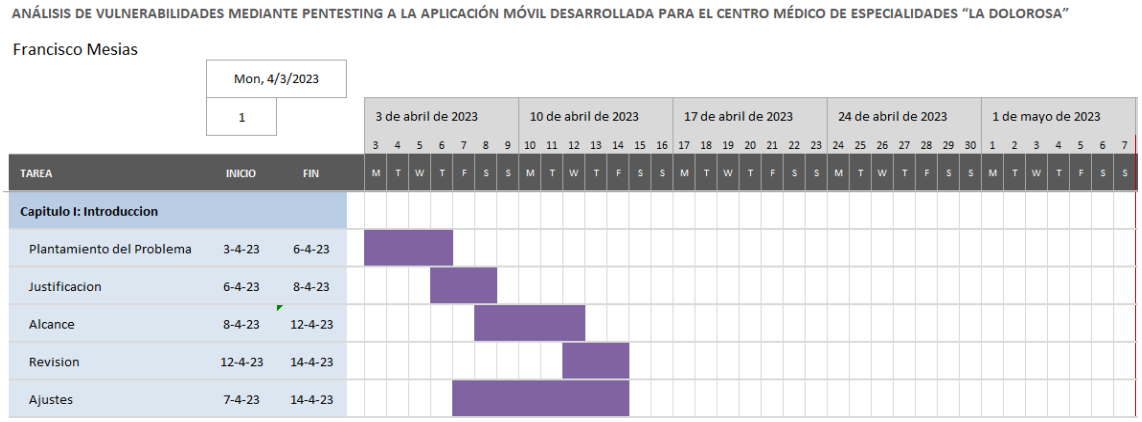


Ilustración 1-3: Cronograma de actividades fase introducción
Realizado por: Mesias Francisco, 2023

En la **Ilustracion 2-3** se observa el tiempo de desarrollo de la fase investigacion del proyecto de investigacion donde se completo la fase de investigacion de las herramientas de pentesting.

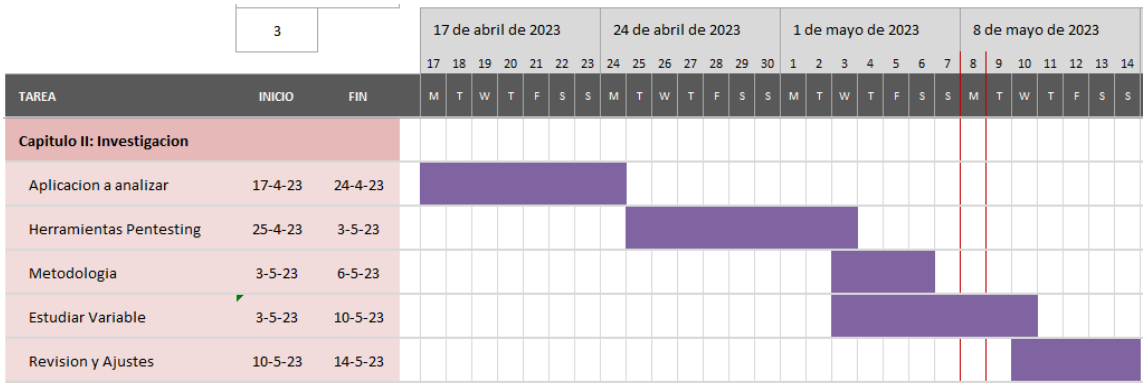


Ilustración 2-3: Cronograma de actividades fase Investigación
Realizado por: Mesias Francisco, 2023

En la **Ilustracion 3-3** se observa el tiempo de desarrollo de la fase metodologia del proyecto de investigacion donde se completo la fase de investigacion de las herramientas de pentesting.

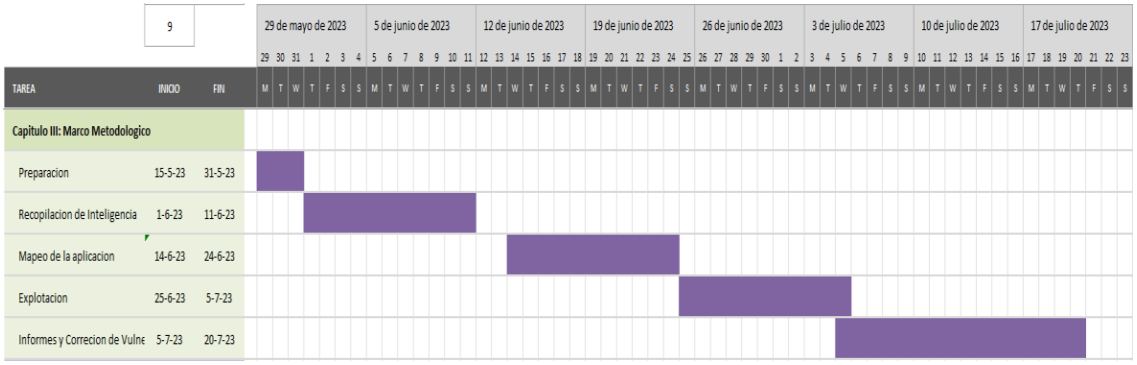


Ilustración 3-3: Cronograma de actividades fase Investigación
Realizado por: Mesias Francisco, 2023

En la **Ilustracion 4-3** se observa el tiempo de desarrollo de la fase resultados del proyecto de investigacion donde se completo la fase de investigacion de las herramientas de pentesting.

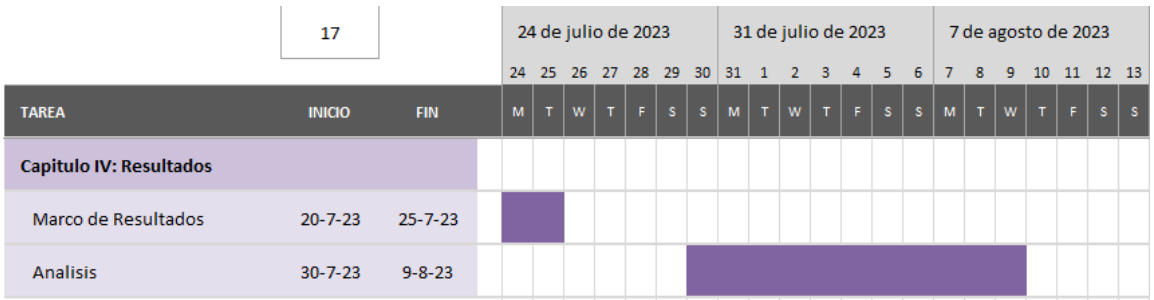


Ilustración 4-3: Cronograma de actividades fase Investigación
Realizado por: Mesias Francisco

4. USO DE LAS HERRAMIENTAS

En este apartado se indicará el uso de las herramientas utilizadas durante el trabajo de integración curricular.

4.1. Mobile Security Framework (MobSF)

MobSF.live, un servicio en la nube al que se puede acceder directamente desde el navegador. Esta herramienta permite analizar aplicaciones en las que se desea llevar a cabo una auditoría de seguridad.

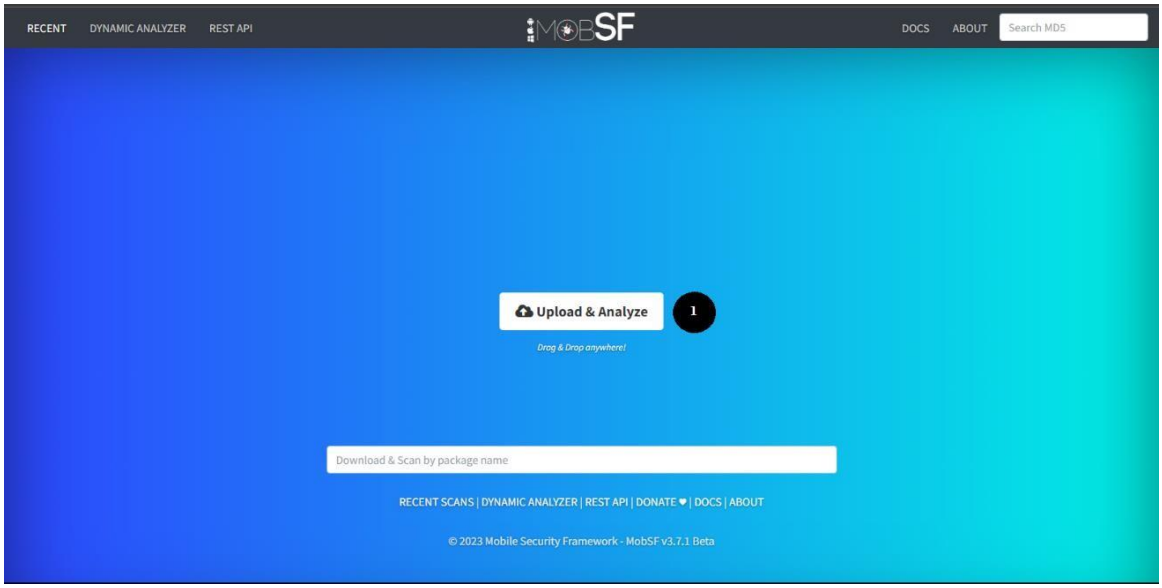


Ilustración 1-4. Inicio de la herramienta MobSF.live y carga de la aplicación a analizar
Fuente: (Mobile Security Framework, 2023)

Como se muestra en la **Ilustración 1-4**, se encuentra la interfaz inicial de la herramienta MobSF. Aquí, la carga del archivo de la aplicación varía según el tipo de esta. Dado que la aplicación que se analiza fue desarrollada en Flutter para Android, se procede a cargar el archivo .apk correspondiente.

Ilustración 2-4. Resumen de la seguridad de la aplicación - MobSF
Fuente: (Mobile Security Framework, 2023)

HIGH: 2 | WARNING: 1 | INFO: 1

TITLE	SEVERITY	DESCRIPTION
Signed Application	info	Application is signed with a code signing certificate
Application vulnerable to Janus Vulnerability	warning	Application is signed with v1 signature scheme, making it vulnerable to Janus vulnerability on Android 5.0-8.0, if signed only with v1 signature scheme. Applications running on Android 5.0-7.0 signed with v1, and v2/v3 scheme is also vulnerable.
Application signed with debug certificate	high	Application signed with a debug certificate. Production application must not be shipped with a debug certificate.

Ilustración 3-4. Informacion de las vulnerabilidades encontradas en la aplicación - MobSF
Fuente: (Mobile Security Framework, 2023)

Tras esperar el tiempo necesario para que MobSF analice la aplicación, se nos presenta una nueva pantalla que muestra un resumen de los hallazgos. Esto se puede observar en la **Ilustración 2-4**. En la sección siguiente de la página web, se detallan específicamente cada una de las vulnerabilidades encontradas, acompañadas de una breve descripción. Esto último puede apreciarse en la **Ilustración 3-4**.

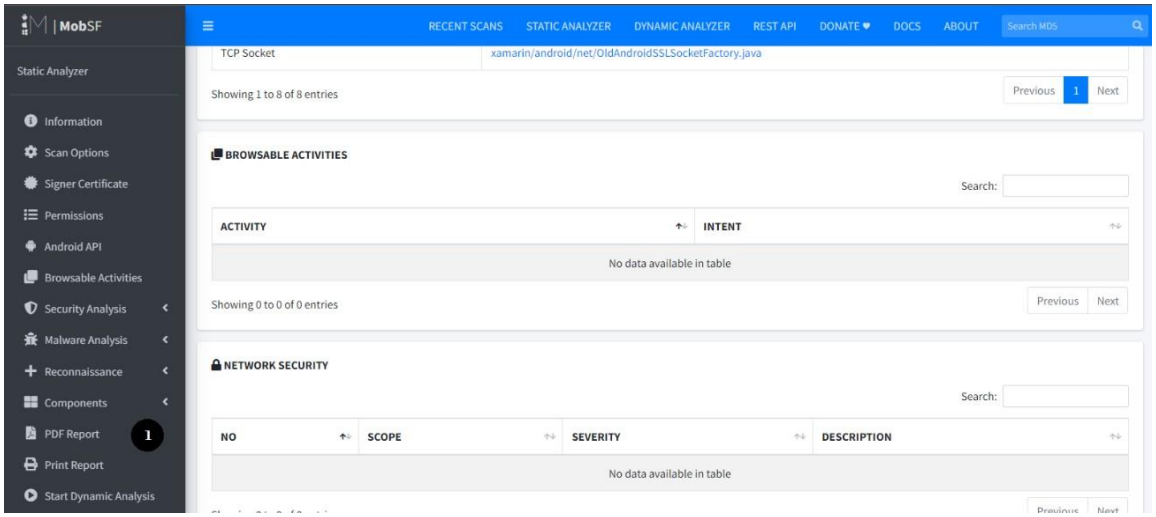


Ilustración 4-4. Generar Reporte - MobSF
Fuente: (Mobile Security Framework, 2023)

Para generar informes con MobSF, se debe navegar a la barra lateral izquierda y seleccionar la opción correspondiente al formato en que se desea descargar los reportes como se observa en la **Ilustración 4-4**.

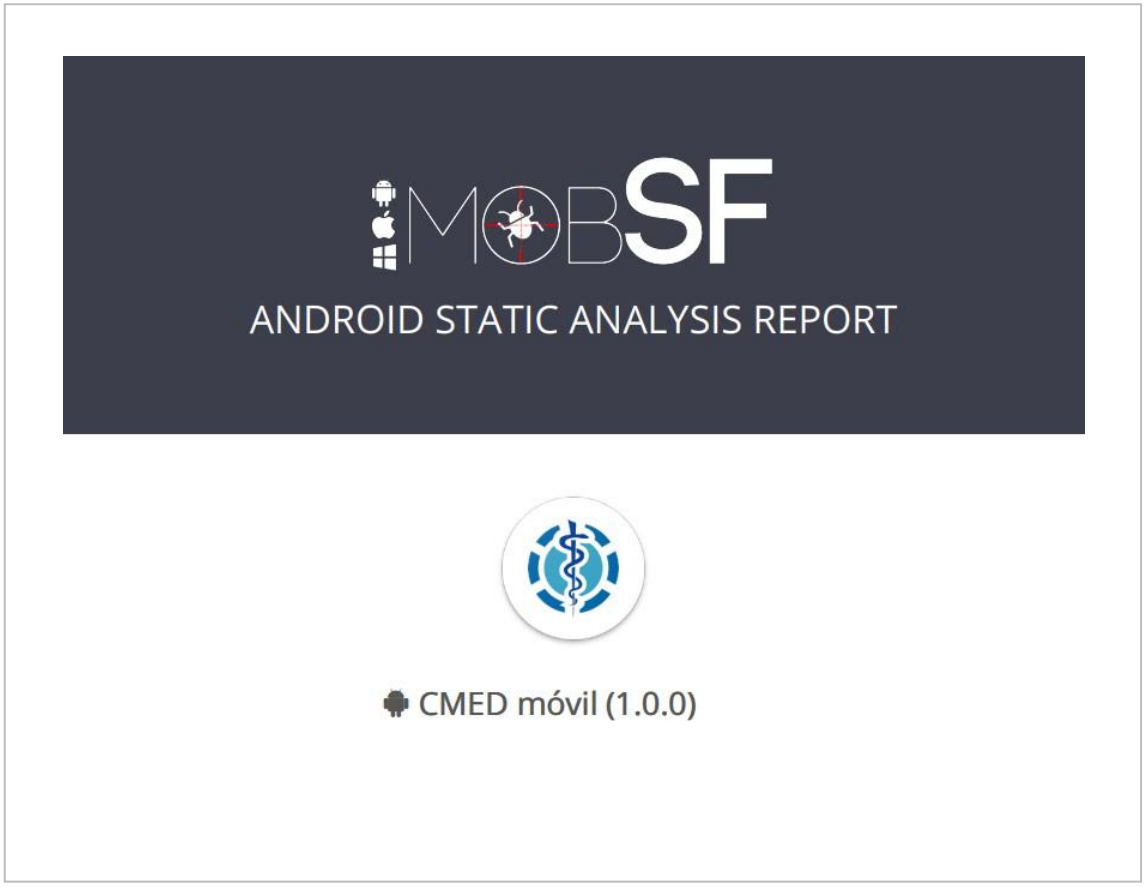


Ilustración 5-4. Reporte generado - MobSF
Fuente: (Mobile Security Framework, 2023)

En la **Ilustración 5-4** se muestra el informe generado por la herramienta MobSF en formato PDF. Este informe incluye diversos detalles, desde el nombre de la aplicación analizada hasta la calificación de seguridad que se le ha asignado.

4.2. Burp Suite

Para utilizar la versión Community Edition de Burp Suite, es necesario descargar e instalar el programa correspondiente. A continuación, se detalla el procedimiento para el uso de esta herramienta como un proxy HTTP(S).

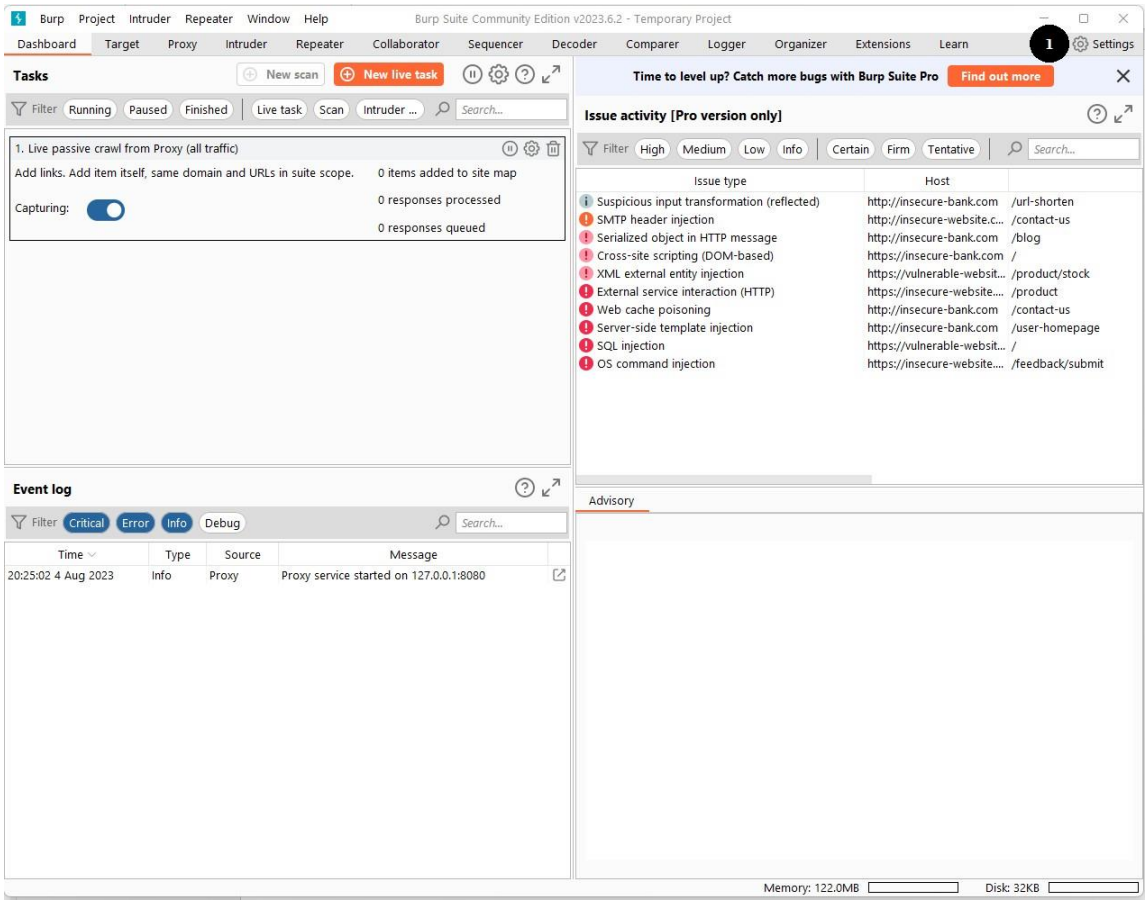


Ilustración 6-4. Pantalla principal – Burp Suite
Fuente: (Mobile Security Framework, 2023)

La **Ilustración 6-4** muestra la pantalla principal de Burp Suite, en la que se pueden apreciar las diversas funcionalidades que ofrece la herramienta. Para configurar esta aplicación como un

proxy, un aspecto clave en este proyecto de integración curricular, es necesario dirigirse a la sección de configuraciones.

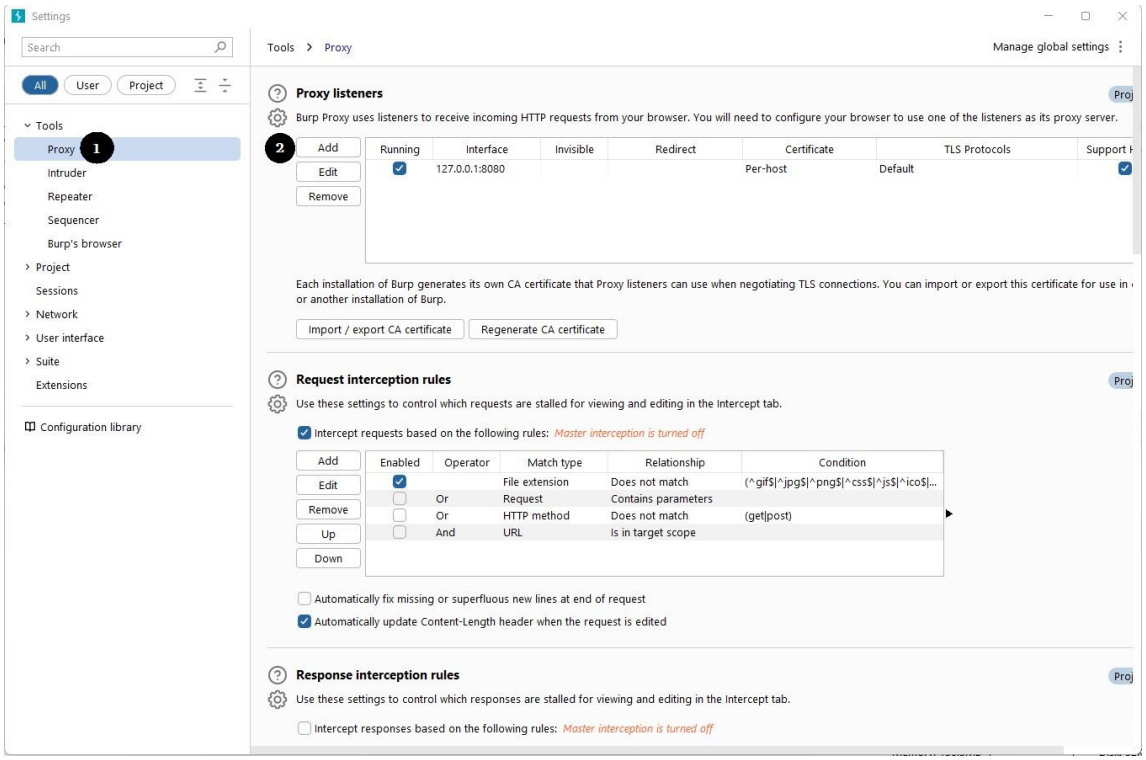


Ilustración 7-4. Configuración del proxy – Burp Suite
Fuente: (Mobile Security Framework, 2023)

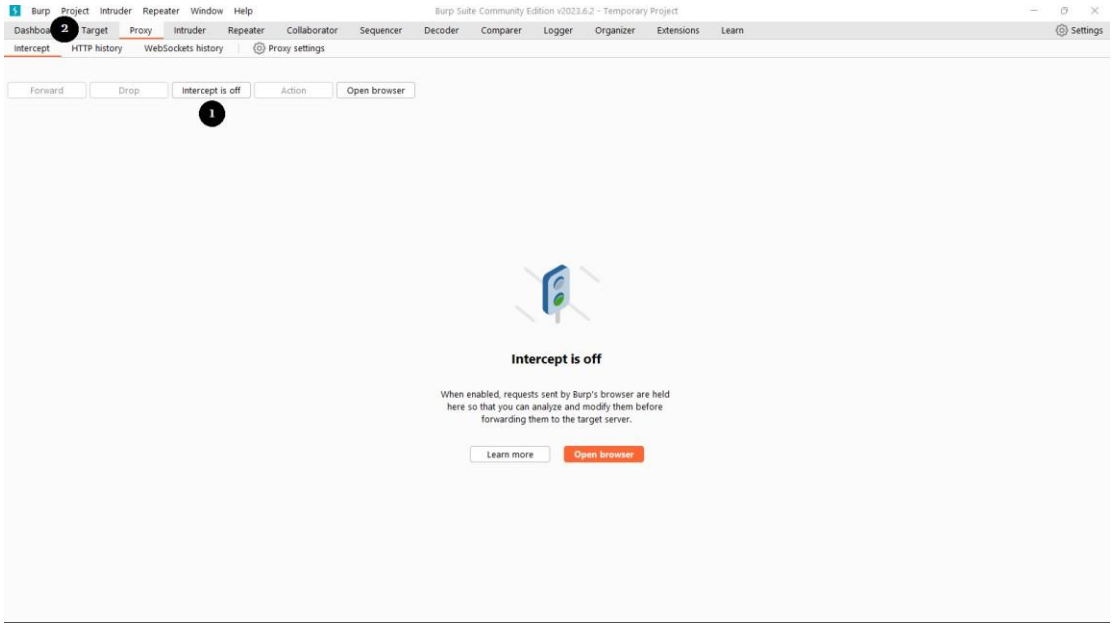


Ilustración 8-4. Encender Proxy– Burp Suite
Fuente: (Mobile Security Framework, 2023)

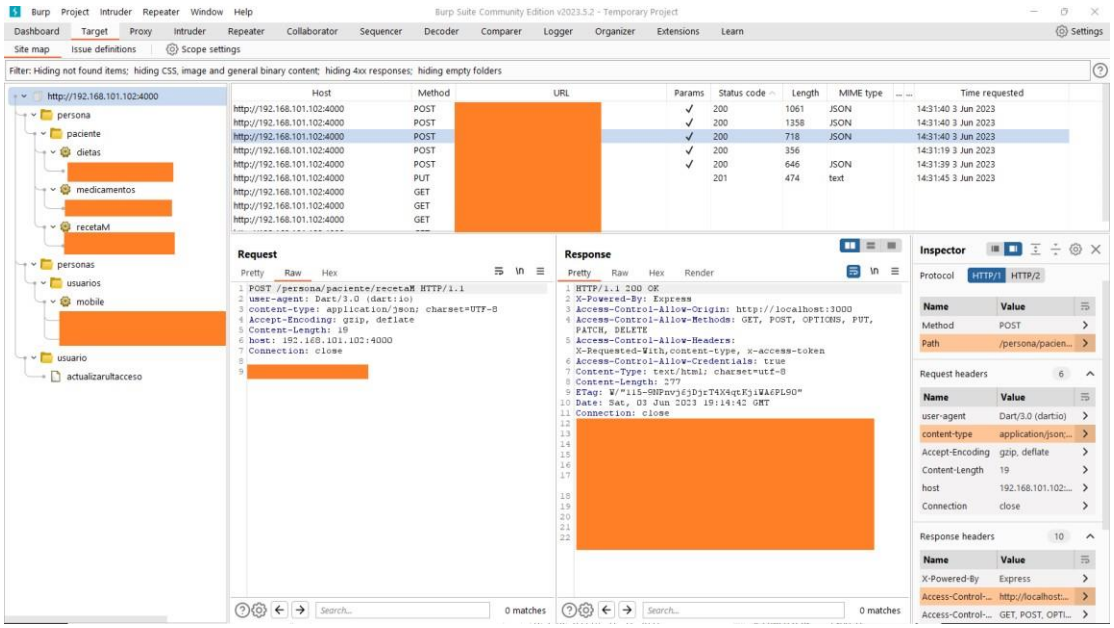


Ilustración 9-4. Visualización de datos interceptados– Burp Suite
Fuente: (Mobile Security Framework, 2023)

Como se puede obervar en las **Ilustraciones 7,8,9-4** es el proceso de la visualización de los paquetes interceptados y el resultado que nos arroja la herramienta Burp Suite.

4.3. SQLMap

Para descargar SQLMap se debe dirigir a la pagina oficial para descargar el archivo correspondiente, además se debe tener descargado Python para poder usar esta herramienta. A continuación, se procederá a explicar como y que comandos se uso para este trabajo de integración curricular.

Ilustración 10-4. Pantalla inicial – SQLMap
Fuente: (Mobile Security Framework, 2023)

La ilustración 10-4 muestra la pantalla inicial de la herramienta SQLMap, una herramienta utilizada en pruebas de penetración. Para iniciar un análisis con esta herramienta, se puede optar por introducir la información requerida en un archivo separado. De esta manera, SQLMap tomará automáticamente los datos desde dicho archivo, simplificando así el proceso.

Ilustración 11-4. Pantalla inicial – SQLMap
Fuente: (Mobile Security Framework, 2023)

En la **Ilustración 11-4**, se demuestra la aplicación de los niveles más elevados disponibles en la herramienta SQLMap para pruebas de penetración, utilizando los comandos 'level=5' y 'risk=3'. El parámetro 'level' se refiere al nivel de intensidad de la prueba, en una escala de 1 a 5, donde 5 es el más intenso. El parámetro 'risk' indica el nivel de riesgo de la prueba, en una escala de 1 a 3, donde 3 es el más arriesgado. Estos comandos están diseñados para evaluar la seguridad de la base de datos de la aplicación de manera exhaustiva y agresiva. Una vez aplicados, la herramienta automatiza los ataques, permitiendo observar tanto la información que arroja como el tiempo que toma en realizar cada prueba.

5. PRUEBAS DE PENETRACIÓN

En esta sección se detallan las vulnerabilidades detectadas a través de las pruebas de penetración llevadas a cabo con las herramientas MobSF, Burpsuite y SQLMap. Además, se presentan las correcciones implementadas para cada una de estas vulnerabilidades, ofreciendo una visión integral de los métodos empleados tanto para identificar los problemas como para resolverlos.

5.1. Vulnerabilidades encontradas

A continuación, en la **Tabla 1-5** se muestran las vulnerabilidades encontradas en la aplicación móvil CMED.

Tabla 1-5: Vulnerabilidades detectadas

Vulnerabilidad
Autenticacion Biometrica
android.permission.REQUEST_INSTALL_PACKAGES
android.permission.READ_EXTERNAL_STORAGE
Aplicacion vulnerable a la vulnerabilidad Janus
Aplicación firmada con certificado de depuración
Algoritmo de certificado vulnerable a la colisión de hash
La aplicación se puede instalar en una versión vulnerable de Android

Aplicación permite copia de seguridad
Generador de numeros inseguros
Transmisión insegura de datos sensibles

Realizado por: Mesias Francisco, 2023.

5.2. Vulnerabilidades corregidas

Una vez identificadas las vulnerabilidades utilizando las herramientas MobSF, Burpsuite y SQLMap, se procede a detallar y corregir cada una de las vulnerabilidades, las cuales se indican a continuación.

- **Autenticacion Biometrica**

```dart
import 'package:flutter/material.dart';
import 'package:local_auth/local_auth.dart';
import 'package:local_auth_android/local_auth_android.dart';

class Authentication {
  static final _auth = LocalAuthentication();
  static Future<bool> canAuhenticate() async =>
      await _auth.canCheckBiometrics || await _auth.isDeviceSupported();

  static Future<bool> authentication() async {
    try {
      if (!await canAuhenticate()) return false;
      return await _auth.authenticate(localizedReason: "dentro de la app");
    } catch (e) {
      print('error $e');
      return false;
    }
  }
}
```

Ilustración 1-5. Autenticacion biométrica corrección vulnerabilidad
Realizado por: Mesias Francisco, 2023

Como se muestra en la **Ilustración 1-5**, se implementó la autenticación biométrica en laaplicación móvil CMED. Esta implementación se realizó mediante la instalación de paquetes específicos, permitiendo su utilización en el código de la aplicación.

- **Android.permission.REQUEST_INSTALL_PACKAGES**

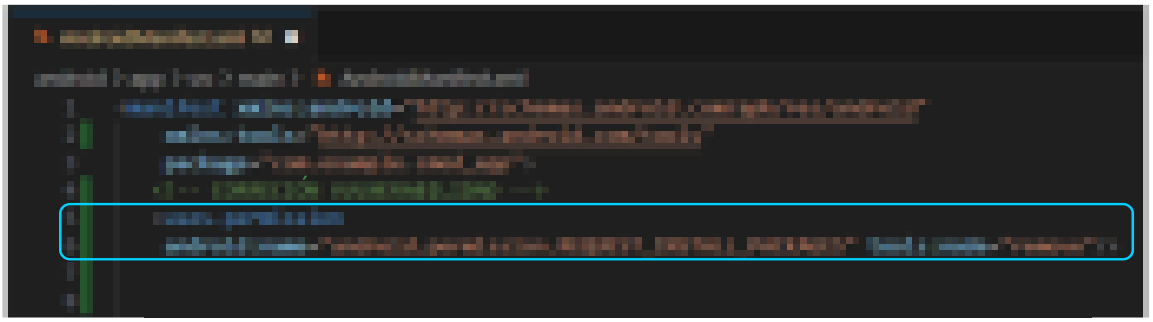


Ilustración 2-5: Android.permission_REQUEST_INSTALL_PACKAGES correction vulnerabilidad
Realizado por: Mesias Francisco, 2023

Siguiendo estas directrices, se realizó un análisis detallado de las dependencias en el archivo pubspec.yaml con el propósito de identificar si alguna requería el permiso INSTALL_PACKAGES. Tras confirmar que no era así, este permiso fue eliminado del AndroidManifest.xml usando toold:node="remove", como se evidencia en las líneas 5-6 de la **Ilustración 2-5**.

- **Android.permission.READ_EXTERNAL_STORAGE**

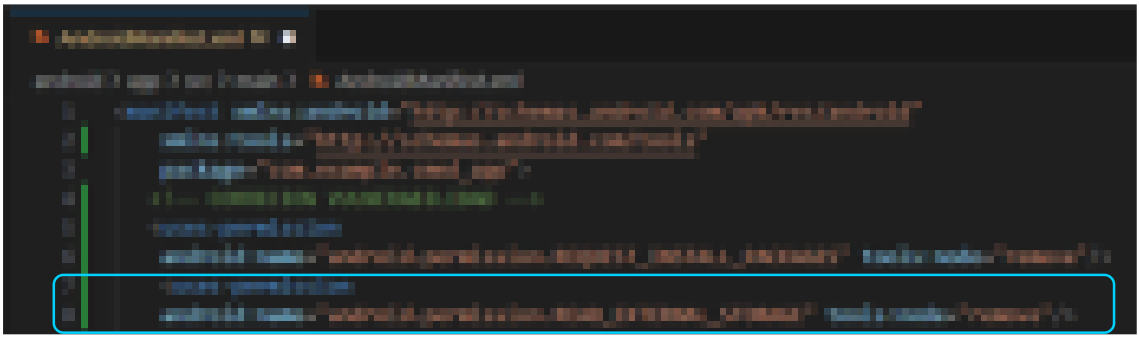


Ilustración 3-5. Android.permission.READ_EXTERNAL_STORAGE corrección vulnerabilidad
Realizado por: Mesias Francisco, 2023

Se llevó a cabo una revisión minuciosa de todas las dependencias en el archivo pubspec.yaml con el objetivo de determinar si alguna dependencia requería el permiso INSTALL_PACKAGES. Tras confirmar que ninguna dependencia lo necesitaba, se decidió retirar este permiso del AndroidManifest.xml utilizando tools:node="remove". Esta acción puede ser verificada en la **Ilustración 3-5**.

- **Aplicacion vulnerable a la vulnerabilidad Janus**

Ilustración 4-5: Vulnerabilidad Janus corrección vulnerabilidad
Realizado por: Mesias Francisco, 2023

Se limitó la instalación de la aplicación móvil, como se puede observar en la Ilustración 4-5, a una versión de SDK mayor o igual a 26. Esto significa que la aplicación solo se puede instalar en dispositivos Android que poseen esta versión o superior, evitando así que se instale en versiones donde la vulnerabilidad de Janus esté sin soporte.

- **Aplicación firmada con certificado de depuración**

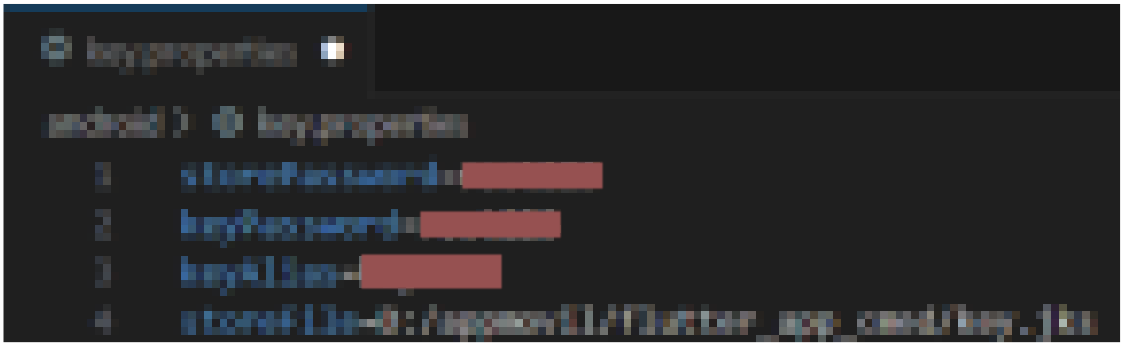


Ilustración 5-5: Aplicación firmada con certificado de depuración corrección vulnerabilidad
Realizado por: Mesias Francisco, 2023

Como se observa en la **Ilustración 5-5**, se procedió a implementar la firma con un certificado de lanzamiento, en lugar de utilizar un certificado de depuración. Para ello, se generó un almacén de claves (key.properties), y a partir de este archivo, se llevó a cabo la configuración del 'signingConfigs' en el archivo 'build.gradle'.

- **Algoritmo de certificado vulnerable a la colisión de hash**

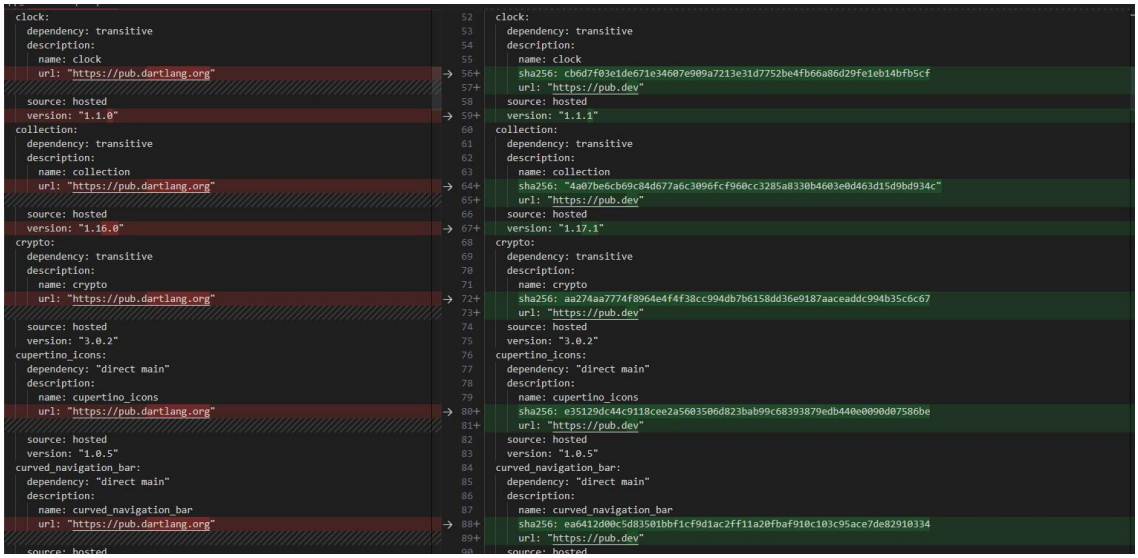


Ilustración 6-5: Colisión de hash vulnerabilidad corregida
Realizado por: Mesias Francisco, 2023

Como se ilustra en la **Ilustración 6-5**, se implementó el algoritmo SHA256withRSA en lugar del menos seguro SHA1. Esta implementación contribuye a proteger la aplicación contra ataques de colisión, asegurando así su autenticidad e integridad.

- **La aplicación se puede instalar en una versión vulnerable de Android**

Ilustración 7-5: App can be installed on a vulnerable Android version implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

La **Ilustración 7-8** muestra la configuración del minSdkVersion, ajustado en consideración a la vulnerabilidad Janus previamente mencionada. Se fijó un minSdkVersion de 26 con el objetivo de maximizar el alcance de usuarios compatibles, garantizando al mismo tiempo que no estén expuestos a la vulnerabilidad de Janus. Adicionalmente, se definió el targetSdkVersion en 33, que corresponde a Android 13, la versión más reciente de Android disponible hasta la fecha.

- **Aplicación permite copia de seguridad**

Ilustración 8-5. Aplicación permite copia de seguridad corrección vulnerabilidad
Realizado por: Mesias Francisco, 2023

Siguiendo las recomendaciones establecidas, se llevó a cabo en la aplicación móvil CMED una revisión y modificación del archivo AndroidManifest.xml, como se evidencia en la **Ilustración 8-5**. Se determinó el valor del atributo 'backup' como 'false', con el propósito de asegurar que la información de la aplicación no quede expuesta de manera inapropiada.

- **Generador de numeros inseguros**

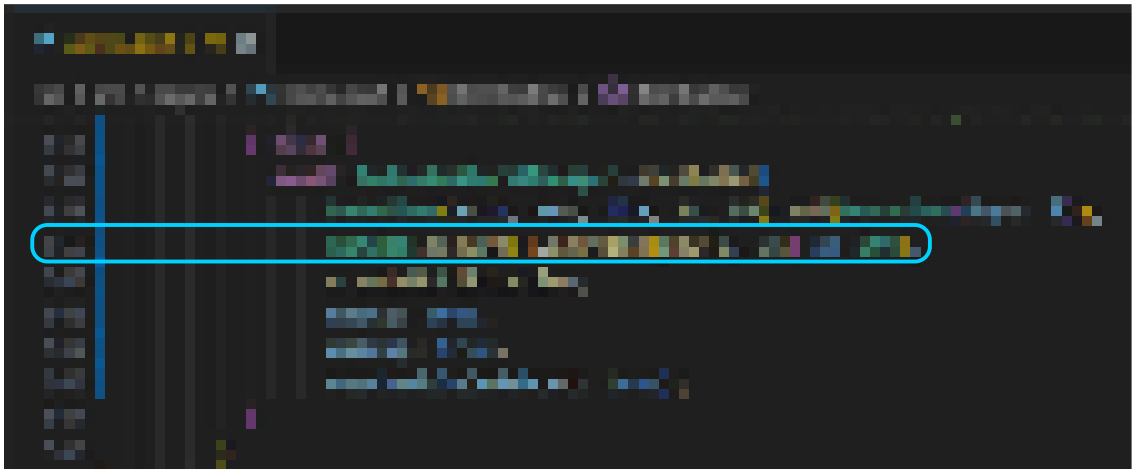


Ilustración 9-5: Android permite copias de seguridad implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

Como se ilustra en la **Ilustración 9-5**, en la aplicación móvil CMED se ha implementado la función Random.secure(). Esto asegura que cualquier número aleatorio generado por la aplicación sea adecuado para propósitos criptográficos, contribuyendo a la seguridad global del software.

- **Transmisión insegura de datos sensibles**

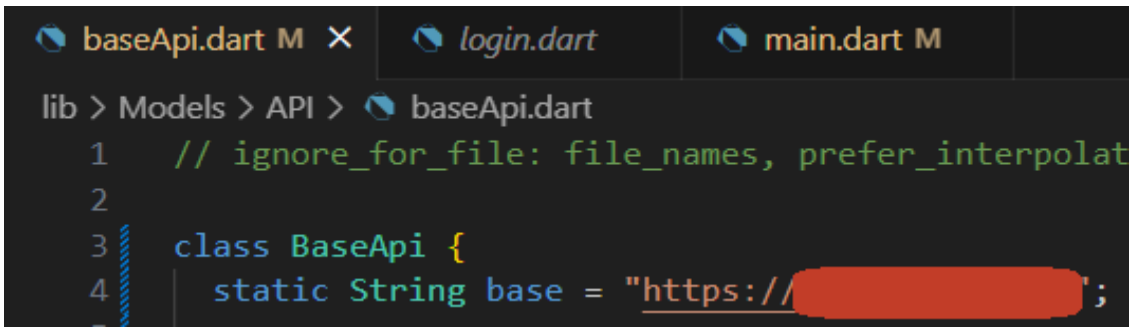


Ilustración 10-5: Transmisión insegura de datos implementación de buenas practicas
Realizado por: Mesias Francisco, 2023

La **Ilustración 10-5** demuestra que, en la aplicación móvil CMED, se ha implementado el protocolo HTTPS para la conexión con la API, la cual se despliega como un servicio en la nube. Esta implementación garantiza una comunicación segura y cifrada entre los nodos de la red.

5.3. Primera ejecución de las pruebas de penetración

Se muestran los reportes de la primera ejecución de las pruebas de penetración a la aplicación móvil CMED.

Ilustración 11-5: Primera ejecución - MobSF
Fuente: (Mobile Security Framework, 2023)

Como se puede observar en la **Ilustración 11-5** se puede ver el resultado de la primeraejecución de seguridad a la aplicación móvil CMED dándonos como resultado un riesgo medio en toda la aplicación.

5.4. Segunda ejecución de las pruebas de penetración

Se muestran los reportes de la primera ejecución de las pruebas de penetración a la aplicación móvil CMED.

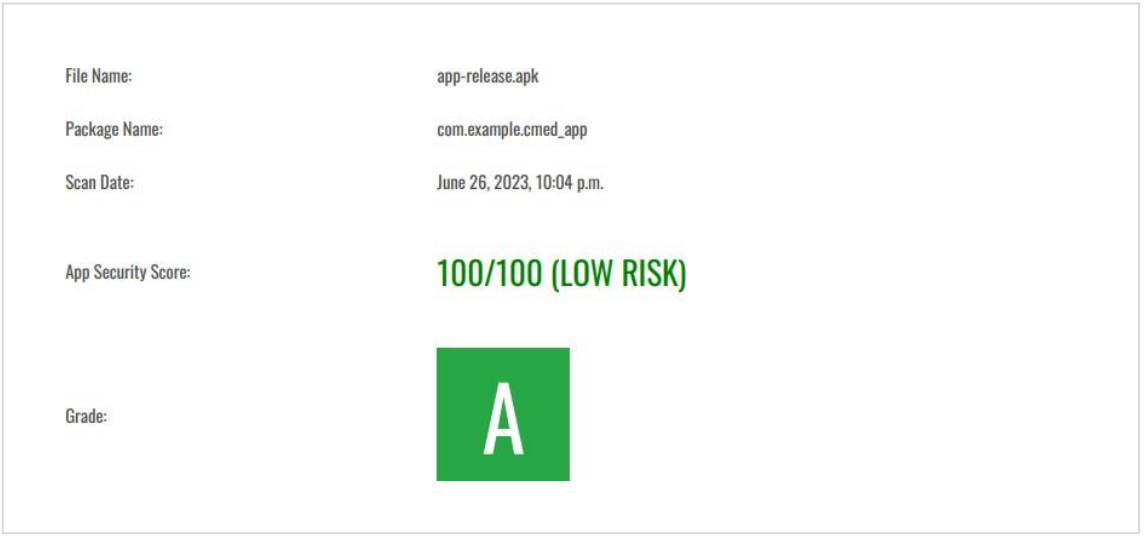


Ilustración 12-5: Segunda ejecución - MobSF
Fuente: (Mobile Security Framework, 2023)

Como se puede observar en la **Ilustración 12-5** se puede ver el resultado de la primeraejecución de seguridad a la aplicación móvil CMED dándonos como resultado un riesgo bajo entoda la aplicación.

ANEXO B: CONTRATO LEGAL

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA SOFTWARE

ANÁLISIS DE VULNERABILIDADES MEDIANTE PENTESTING A LA APLICACIÓN MÓVIL DESARROLLADA PARA EL CENTRO MÉDICO DE ESPECIALIDADES “LA DOLOROSA”

MANUAL TÉCNICO

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO DE SOFTWARE

AUTOR

FRANCISCO XAVIER MESIAS FLORES

Riobamba – Ecuador

2023

CONTRATO DE SERVICIOS DE PRUEBAS DE PENETRACIÓN

Quito, 01-04-2023

Francisco Xavier Mesias Flores con cedula: 1105640534 en adelante "Pentester"Y Jhony Ruperto Riera Ortiz con cedula: 1719624999 en adelante "Cliente"

1. DEFINICIONES

Prueba de Penetración (PenTest): Procedimiento de evaluación de la seguridad de un sistema informático o red que simula un ataque de un usuario malintencionado.

Objeto del Contrato: El Pentester se compromete a realizar una Prueba de Penetración sobre la aplicación móvil CMED del Cliente.

2. ALCANCE DE LOS SERVICIOS

El Pentester se compromete a realizar una Prueba de Penetración en la aplicación móvil del Cliente utilizando diversas herramientas y técnicas, incluyendo a MobSF, Burp Suite y Sqlmap. El objetivo de estas pruebas es identificar, analizar y proporcionar recomendaciones para corregir las vulnerabilidades encontradas en la aplicación móvil CMED antes de su despliegue por lo tanto el encargado legito de la aplicación es el Ing. Jony Ruperto Riera Ortiz con cedula 1719624999.

3. CONFIDENCIALIDAD

Toda la información a la que el Pentester tenga acceso durante la realización del trabajo se considerará confidencial y no se revelará a terceros sin el consentimiento por escrito del Cliente.

4. CONSENTIMIENTO

El Cliente otorga su consentimiento para que el Pentester realice la Prueba de Penetración en la aplicación. El Cliente entiende y acepta que la Prueba de Penetración puede interrumpir o degradar el desarrollo de la aplicación.

5. INDEMNIZACIÓN

El Cliente se compromete a indemnizar y mantener al Pentester libre de cualquier reclamación, daño, pérdida, responsabilidad, costo o gasto, incluidos los honorarios razonables de abogados, que resulten directa o indirectamente de las Pruebas de Penetración realizadas bajo este Contrato.

6. DURACIÓN

Este Contrato entrará en vigor en la fecha de firma y continuará hasta que se complete el servicio, a menos que se rescinda antes por cualquier motivo.

7. DERECHOS Y OBLIGACIONES

El Pentester se compromete a realizar el trabajo de manera profesional y ética. El Cliente se compromete a facilitar al Pentester toda la información necesaria para la realización de la Prueba de Penetración.

Francisco Xavier Mesias Flores
1105640534

Jhony Ruperto Riera Ortiz
1719624999

ANEXO C: INFORME DE VULNERABILIDADES

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA SOFTWARE

ANÁLISIS DE VULNERABILIDADES MEDIANTE PENTESTING A LA APLICACIÓN MÓVIL DESARROLLADA PARA EL CENTRO MÉDICO DE ESPECIALIDADES “LA DOLOROSA”

INFORME DE VULNERABILIDADES

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO DE SOFTWARE

AUTOR

FRANCISCO XAVIER MESIAS FLORES

Riobamba – Ecuador

2023

MOBSF

ANDROID STATIC ANALYSIS REPORT

CMED móvil (1.0.0)

File Name:	app-release.apk
Package Name:	com.example.cmed_app
Scan Date:	June 4, 2023, 1:48 a.m.
App Security Score:	46/100 (MEDIUM RISK)
Grade:	B

FINDINGS SEVERITY

HIGH	MEDIUM	INFO	SECURE	HOTSPOT
2	6	2	1	1

FILE INFORMATION

File Name: app-release.apk
Size: 25.0MB
MD5: bdf548245cd9b1095d2c24db97f4404f
SHA1: 0364b4d0e2ce784f64de1b2558e20130ce5b8c5b
SHA256: 1dc790ea722a0c285003469ebfc8324f34f37674e57cc037518a5f0b1e3a038d

APP INFORMATION

App Name: CMED móvil
Package Name: com.example.cmed_app
Main Activity: com.example.cmed_app.MainActivity
Target SDK: 33
Min SDK: 16
Max SDK:
Android Version Name: 1.0.0
Android Version Code: 1

APP COMPONENTS

Activities: 1
Services: 1
Receivers: 4
Providers: 1
Exported Activities: 0
Exported Services: 0
Exported Receivers: 0
Exported Providers: 0

CERTIFICATE INFORMATION

APK is signed
v1 signature: True
v2 signature: True
v3 signature: False
Found 1 unique certificates
Subject: CN=Android Debug, O=Android, C=US
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2023-05-25 03:39:17+00:00
Valid To: 2053-05-17 03:39:17+00:00
Issuer: CN=Android Debug, O=Android, C=US
Serial Number: 0x1
Hash Algorithm: sha1
md5: 23850b1c08009f7362cdf978a7b5f6cf
sha1: b8aae15374a4b487a22fbe790a570f7c725c8369
sha256: 9c78131eb7e2ca30bd9df4729ae7e99212d1682c0da11d9c929fe0982a100459
sha512: 68168aad7fa3d5bd7f0324aa4860fffd8d84b56636dc22fe929eb3b29b9be075248df7626cbcdce1c925533afc9aaadbcebf617462f45a1d39403f17e2ad1732
PublicKey Algorithm: rsa
Bit Size: 2048
Fingerprint: fd6d0d6e1f5d991e3f767d75a3a85d4458cd50f4cd7dcd488c47fc6c01f5fa5a

APPLICATION PERMISSIONS

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.RECEIVE_BOOT_COMPLETED	normal	automatically start at boot	Allows an application to start itself as soon as the system has finished booting. This can make it take longer to start the phone and allow the application to slow down the overall phone by always running.
android.permission.WAKE_LOCK	normal	prevent phone from sleeping	Allows an application to prevent the phone from going to sleep.
android.permission.SCHEDULE_EXACT_ALARM	normal		Allows an app to use exact alarm scheduling APIs to perform timing sensitive background work.
android.permission.REQUEST_INSTALL_PACKAGES	dangerous	Allows an application to request installing packages.	Malicious applications can use this to try and trick users into installing additional malicious packages.
android.permission.READ_EXTERNAL_STORAGE	dangerous	read external storage contents	Allows an application to read from external storage.
android.permission.VIBRATE	normal	control vibrator	Allows the application to control the vibrator.
android.permission.USE_FULL_SCREEN_INTENT	normal		Required for apps targeting Build.VERSION_CODES.Q that want to use notification full screen intents.
android.permission.POST_NOTIFICATIONS	unknown	Unknown permission	Unknown permission from android reference

APKID ANALYSIS

FILE	DETAILS
classes.dex	FINDINGS: Anti-VM Code — DETAILS: Build.FINGERPRINT check FINDINGS: Compiler — DETAILS: r8 without marker (suspicious)

NETWORK SECURITY

NO	SCOPE	SEVERITY	DESCRIPTION

CERTIFICATE ANALYSIS

HIGH: 2 | WARNING: 1 | INFO: 1

TITLE	SEVERITY	DESCRIPTION
Signed Application	info	Application is signed with a code signing certificate
Application vulnerable to Janus Vulnerability	warning	Application is signed with v1 signature scheme, making it vulnerable to Janus vulnerability on Android 5.0-8.0, if signed only with v1 signature scheme. Applications running on Android 5.0-7.0 signed with v1, and v2/v3 scheme is also vulnerable.
Application signed with debug certificate	high	Application signed with a debug certificate. Production application must not be shipped with a debug certificate.

TITLE	SEVERITY	DESCRIPTION
Certificate algorithm vulnerable to hash collision	high	Application is signed with SHA1withRSA. SHA1 hash algorithm is known to have collision issues.

MANIFEST ANALYSIS

HIGH: 0 | WARNING: 2 | INFO: 0 | SUPPRESSED: 0

NO	ISSUE	SEVERITY	DESCRIPTION
1	App can be installed on a vulnerable Android version [minSdk=16]	warning	This application can be installed on an older version of android that has multiple unfixed vulnerabilities. Support an Android version > 8, API 26 to receive reasonable security updates.
2	Application Data can be Backed up [android:allowBackup] flag is missing.	warning	The flag [android:allowBackup] should be set to false. By default it is set to true and allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.

CODE ANALYSIS

HIGH: 0 | WARNING: 3 | INFO: 2 | SECURE: 0 | SUPPRESSED: 0

NO	ISSUE	SEVERITY	STANDARDS	FILES
1	App can read/write to External Storage. Any App can read data written to External Storage.	warning	CWE: CWE-276: Incorrect Default Permissions OWASP Top 10: M2: Insecure Data Storage OWASP MASVS: MSTG-STORAGE-2	t1/g.java t1/i.java
2	The App logs information. Sensitive information should never be logged.	info	CWE: CWE-532: Insertion of Sensitive Information into Log File OWASP MASVS: MSTG-STORAGE-3	c1/b.java dev/fluttercommunity/plus/androidalarmmanager/AlarmService.java dev/fluttercommunity/plus/androidalarmmanager/RebootBroadcastReceiver.java dev/fluttercommunity/plus/androidalarmmanager/a.java dev/fluttercommunity/plus/androidalarmmanager/b.java g/a.java j/g.java j0/a.java k/c.java p/b.java p/e.java p/h.java p/l.java p/m.java p/q.java q/a.java q/b.java q/f.java r/c.java r/e.java r/f.java r/g.java r/j.java r/k.java s/a.java s/e.java t1/i.java

NO	ISSUE	SEVERITY	STANDARDS	FILES
				w/b.java y/a.java y/c0.java y/g.java y/i.java y/u.java y/v.java y/x.java z/b.java
3	Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	warning	CWE: CWE-312: Cleartext Storage of Sensitive Information OWASP Top 10: M9: Reverse Engineering OWASP MASVS: MSTG-STORAGE-14	com/dexterous/flutterlocalnotifications/FlutterLocalNotificationsPlugin.java com/dexterous/flutterlocalnotifications/models/NotificationDetails.java
4	The App uses an insecure Random Number Generator.	warning	CWE: CWE-330: Use of Insufficiently Random Values OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-6	i2/a.java i2/b.java j2/a.java
5	This App copies data to clipboard. Sensitive data should not be copied to clipboard as other applications can access it.	info	OWASP MASVS: MSTG-STORAGE-10	io/flutter/plugin/editing/b.java io/flutter/plugin/platform/c.java

SHARED LIBRARY BINARY ANALYSIS

NO	SHARED OBJECT	NX	STACK CANARY	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
1	lib/armeabi-v7a/libflutter.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	True info This shared object has a stack canary value added to the stack so that it will be overwritten by a stack buffer that overflows the return address. This allows detection of overflows by verifying the integrity of the canary before function return.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	False warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	True info Symbols are stripped.
2	lib/armeabi-v7a/libapp.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	False warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	True info Symbols are stripped.

NO	SHARED OBJECT	NX	STACK CANARY	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
3	lib/arm64-v8a/libflutter.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	True info The shared object has the following fortified functions: ['__vsnprintf_chk', '__read_chk', '__memcpy_chk', '__strcpy_chk', '__strlen_chk', '__memmove_chk']	True info Symbols are stripped.
4	lib/arm64-v8a/libapp.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	False warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	True info Symbols are stripped.

NO	SHARED OBJECT	NX	STACK CANARY	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
5	lib/x86_64/libflutter.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	True info The shared object has the following fortified functions: ['__vsnprintf_chk', '__read_chk', '__memcpy_chk', '__strcpy_chk', '__strlen_chk', '__memmove_chk']	True info Symbols are stripped.
6	lib/x86_64/libapp.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	False warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	True info Symbols are stripped.

NIAP ANALYSIS v1.3

NO	IDENTIFIER	REQUIREMENT	FEATURE	DESCRIPTION
1	FCS_RBG_EXT.1.1	Security Functional Requirements	Random Bit Generation Services	The application use no DRBG functionality for its cryptographic operations.
2	FCS_STO_EXT.1.1	Security Functional Requirements	Storage of Credentials	The application does not store any credentials to non-volatile memory.
3	FCS_CKM_EXT.1.1	Security Functional Requirements	Cryptographic Key Generation Services	The application generate no asymmetric cryptographic keys.
4	FDP_DEC_EXT.1.1	Security Functional Requirements	Access to Platform Resources	The application has access to no hardware resources.
5	FDP_DEC_EXT.1.2	Security Functional Requirements	Access to Platform Resources	The application has access to no sensitive information repositories.
6	FDP_NET_EXT.1.1	Security Functional Requirements	Network Communications	The application has no network communications.
7	FDP_DAR_EXT.1.1	Security Functional Requirements	Encryption Of Sensitive Application Data	The application does not encrypt files in non-volatile memory.
8	FMT_MEC_EXT.1.1	Security Functional Requirements	Supported Configuration Mechanism	The application invoke the mechanisms recommended by the platform vendor for storing and setting configuration options.
9	FTP_DIT_EXT.1.1	Security Functional Requirements	Protection of Data in Transit	The application does encrypt some transmitted data with HTTPS/TLS/SSH between itself and another trusted IT product.

DOMAIN MALWARE CHECK

DOMAIN	STATUS	GEOLOCATION
192.168.101.102	ok	**IP:** 192.168.101.102 **Country:** - **Region:** - **City:** - **Latitude:** 0.000000 **Longitude:** 0.000000 **View:** Google Map
cdn-icons-png.flaticon.com	ok	**IP:** 104.103.64.64 **Country:** Finland **Region:** Uusimaa **City:** Helsinki **Latitude:** 60.169521 **Longitude:** 24.935450 **View:** Google Map
docs.flutter.dev	ok	**IP:** 199.36.158.100 **Country:** United States of America **Region:** California **City:** Mountain View **Latitude:** 37.405991 **Longitude:** -122.078514 **View:** Google Map
api.flutter.dev	ok	**IP:** 199.36.158.100 **Country:** United States of America **Region:** California **City:** Mountain View **Latitude:** 37.405991 **Longitude:** -122.078514 **View:** Google Map

DOMAIN	STATUS	GEOLOCATION
www.w3.org	ok	**IP:** 104.18.22.19 **Country:** United States of America **Region:** California **City:** San Francisco **Latitude:** 37.775700 **Longitude:** -122.395203 **View:** Google Map
developer.android.com	ok	**IP:** 172.217.21.174 **Country:** United States of America **Region:** California **City:** Mountain View **Latitude:** 37.405991 **Longitude:** -122.078514 **View:** Google Map
github.com	ok	**IP:** 140.82.121.4 **Country:** United States of America **Region:** California **City:** San Francisco **Latitude:** 37.775700 **Longitude:** -122.395203 **View:** Google Map
xmlpull.org	ok	**IP:** 185.199.108.153 **Country:** United States of America **Region:** Pennsylvania **City:** California **Latitude:** 40.065632 **Longitude:** -79.891708 **View:** Google Map

DOMAIN	STATUS	GEOLOCATION
fonts.gstatic.com	ok	**IP:** 142.250.74.99 **Country:** United States of America **Region:** California **City:** Mountain View **Latitude:** 37.405991 **Longitude:** -122.078514 **View:** Google Map
flutter.dev	ok	**IP:** 199.36.158.100 **Country:** United States of America **Region:** California **City:** Mountain View **Latitude:** 37.405991 **Longitude:** -122.078514 **View:** Google Map
schemas.android.com	ok	No Geolocation information available.

EMAILS

EMAIL	FILE
_cookie@13463476.fromsetcoo authenticationscheme@13463476.fromstring _list@0150898.of _httpparser@13463476.responsepa _list@0150898.generate _typeerror@0150898._create _list@0150898._ofgrowabl _hashcollisionnode@416137193.fromcollis _list@0150898._ofefficie	

EMAIL	FILE
_double@0150898.fromintege _growablelist@0150898._literal3 _future@4048458.immediate _growablelist@0150898._literal _growablelist@0150898._ofother _link@14069316.fromrawpat _growablelist@0150898.withcapaci _timer@1026248._internal _growablelist@0150898._literal6 _growablelist@0150898._literal5 _rawsocket@14069316._readpipe _receiveportimpl@1026248.fromrawrec _colorfilter@15065589.mode _assetmanifestbin@735287047.fromstanda _list@0150898._ofarray _socket@14069316._readpipe _timer@1026248.periodic _growablelist@0150898._literal2 _bigintimpl@0150898.from _compressednode@416137193.single _list@0150898.empty _list@0150898._ofother _bytebuffer@7027147._new _directory@14069316.fromrawpat storationinformation@829124995.fromserial _invocationmirror@0150898._withtype ngstreamsubscription@4048458.zoned _assertionerror@0150898._create _rawsocket@14069316._writepipe _nativesocket@14069316.normal _colorfilter@15065589.lineartosr _filestream@14069316.forstdin _colorfilter@15065589.srgbtoline _uri@0150898.file _growablelist@0150898._literal1 _uri@0150898.directory _growablelist@0150898._literal8 _file@14069316.fromrawpat _growablelist@0150898._literal4 _growablelist@0150898._ofgrowabl	lib/armeabi-v7a/libapp.so

EMAIL	FILE
_growablelist@0150898.generate _uri@0150898.notsimple _growablelist@0150898._literal7 _growablelist@0150898._ofefficie _future@4048458.immediatee _nativesocket@14069316.pipe	
appro@openssl.org	lib/arm64-v8a/libflutter.so

MOBSF

ANDROID STATIC ANALYSIS REPORT

CMED móvil (2.0.0)

File Name:	app-release.apk
Package Name:	com.example.cmed_app
Scan Date:	June 26, 2023, 10:04 p.m.
App Security Score:	100/100 (LOW RISK)
Grade:	A

FINDINGS SEVERITY

HIGH	MEDIUM	INFO	SECURE	HOTSPOT
0	0	0	1	0

FILE INFORMATION

File Name: app-release.apk
Size: 52.19MB
MD5: 55645dbe6137a0a008b591ae8e231f84
SHA1: 72cf145919c3a0c351e71ce28c967689ccfaad96
SHA256: 8d010536819dc9730fbd07ab60110832c3a757b0b3cdb50da7e036ff16850cdc

APP INFORMATION

App Name: CMED móvil
Package Name: com.example.cmed_app
Main Activity: com.example.cmed_app.MainActivity
Target SDK: 33
Min SDK: 26
Max SDK:
Android Version Name: 1.0.0
Android Version Code: 1

APP COMPONENTS

Activities: 1
Services: 1
Receivers: 4
Providers: 1
Exported Activities: 0
Exported Services: 0
Exported Receivers: 0
Exported Providers: 0

CERTIFICATE INFORMATION

APK is signed
v1 signature: False
v2 signature: True
v3 signature: False
Found 1 unique certificates
Subject: C=Unknown, ST=Unknown, L=Unknown, O=Unknown, OU=Unknown, CN=Unknown
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2023-06-25 16:01:29+00:00
Valid To: 2050-11-10 16:01:29+00:00
Issuer: C=Unknown, ST=Unknown, L=Unknown, O=Unknown, OU=Unknown, CN=Unknown
Serial Number: 0x45ba308b2b28da49
Hash Algorithm: sha256
md5: 0c84d9790880cda7cf882e624a087402
sha1: 52a271272a9b43e305ae851ef22f9d65b3d2f06a
sha256: cfceafd3c081cc96320ad5ebd32eaf2a45bb901cea749f9e7b0001968178bfbd
sha512: 8ac8c99e8e42bf23398d401af0336bb7428749e8dab82c42d795b49aefb3f630dceaf6a14f5a0fd1d54c67480b55243353e1ffbd9d9d86b4bfa427ab196f9333
PublicKey Algorithm: rsa
Bit Size: 2048
Fingerprint: ff0ed58623eb8cb0f5240cc5f2a4f87a0bd9530fa2f7b6d7a45490ce3c06a660

APPLICATION PERMISSIONS

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.RECEIVE_BOOT_COMPLETED	normal	automatically start at boot	Allows an application to start itself as soon as the system has finished booting. This can make it take longer to start the phone and allow the application to slow down the overall phone by always running.
android.permission.WAKE_LOCK	normal	prevent phone from sleeping	Allows an application to prevent the phone from going to sleep.
android.permission.SCHEDULE_EXACT_ALARM	normal		Allows an app to use exact alarm scheduling APIs to perform timing sensitive background work.
android.permission.VIBRATE	normal	control vibrator	Allows the application to control the vibrator.
android.permission.USE_FULL_SCREEN_INTENT	normal		Required for apps targeting Build.VERSION_CODES.Q that want to use notification full screen intents.
android.permission.POST_NOTIFICATIONS	unknown	Unknown permission	Unknown permission from android reference

APKID ANALYSIS

FILE	DETAILS

FILE	DETAILS
classes.dex	FINDINGS: Anti-VM Code — DETAILS: Build.FINGERPRINT check FINDINGS: Compiler — DETAILS: dx

NETWORK SECURITY

NO	SCOPE	SEVERITY	DESCRIPTION

CERTIFICATE ANALYSIS

HIGH: 0 | WARNING: 0 | INFO: 1

TITLE	SEVERITY	DESCRIPTION
Signed Application	info	Application is signed with a code signing certificate

MANIFEST ANALYSIS

HIGH: 0 | WARNING: 0 | INFO: 0 | SUPPRESSED: 0

NO	ISSUE	SEVERITY	DESCRIPTION

</> CODE ANALYSIS

NO	ISSUE	SEVERITY	STANDARDS	FILES

SHARED LIBRARY BINARY ANALYSIS

NO	SHARED OBJECT	NX	STACK CANARY	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
1	lib/armeabi-v7a/libflutter.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	True info This shared object has a stack canary value added to the stack so that it will be overwritten by a stack buffer that overflows the return address. This allows detection of overflows by verifying the integrity of the canary before function return.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	False warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	True info Symbols are stripped.

NO	SHARED OBJECT	NX	STACK CANARY	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
2	lib/armeabi-v7a/libapp.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	False warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	True info Symbols are stripped.
3	lib/arm64-v8a/libflutter.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	True info The shared object has the following fortified functions: ['__vsnprintf_chk', '__read_chk', '__memcpy_chk', '__strcpy_chk', '__strlen_chk', '__memmove_chk']	True info Symbols are stripped.

NO	SHARED OBJECT	NX	STACK CANARY	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
4	lib/arm64-v8a/libapp.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	False warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	True info Symbols are stripped.
5	lib/x86_64/libflutter.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	True info The shared object has the following fortified functions: ['__vsnprintf_chk', '__read_chk', '__memcpy_chk', '__strcpy_chk', '__strlen_chk', '__memmove_chk']	True info Symbols are stripped.

NO	SHARED OBJECT	NX	STACK CANARY	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
6	lib/x86_64/libapp.so	True info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	False high This shared object does not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return address. Use the option -fstack-protector-all to enable stack canaries.	None info The shared object does not have run-time search path or RPATH set.	None info The shared object does not have RUNPATH set.	False warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	True info Symbols are stripped.

NIAP ANALYSIS v1.3

NO	IDENTIFIER	REQUIREMENT	FEATURE	DESCRIPTION
1	FCS_STO_EXT.1.1	Security Functional Requirements	Storage of Credentials	The application does not store any credentials to non-volatile memory.
2	FCS_CKM_EXT.1.1	Security Functional Requirements	Cryptographic Key Generation Services	The application generate no asymmetric cryptographic keys.
3	FDP_DEC_EXT.1.1	Security Functional Requirements	Access to Platform Resources	The application has access to no hardware resources.

NO	IDENTIFIER	REQUIREMENT	FEATURE	DESCRIPTION
4	FDP_DEC_EXT.1.2	Security Functional Requirements	Access to Platform Resources	The application has access to no sensitive information repositories.
5	FDP_NET_EXT.1.1	Security Functional Requirements	Network Communications	The application has no network communications.
6	FDP_DAR_EXT.1.1	Security Functional Requirements	Encryption Of Sensitive Application Data	The application does not encrypt files in non-volatile memory.
7	FTP_DIT_EXT.1.1	Security Functional Requirements	Protection of Data in Transit	The application does not encrypt any data in traffic or does not transmit any data between itself and another trusted IT product.

! OFAC SANCTIONED COUNTRIES

This app may communicate with the following OFAC sanctioned list of countries.

DOMAIN	COUNTRY/REGION

DOMAIN MALWARE CHECK

DOMAIN	STATUS	GEOLOCATION

DOMAIN	STATUS	GEOLOCATION
docs.flutter.dev	ok	**IP:** 199.36.158.100 **Country:** United States of America **Region:** California **City:** Mountain View **Latitude:** 37.405991 **Longitude:** -122.078514 **View:** Google Map
xmlpull.org	ok	**IP:** 185.199.109.153 **Country:** United States of America **Region:** Pennsylvania **City:** California **Latitude:** 40.065632 **Longitude:** -79.891708 **View:** Google Map
fonts.gstatic.com	ok	**IP:** 216.58.207.227 **Country:** United States of America **Region:** California **City:** Mountain View **Latitude:** 37.405991 **Longitude:** -122.078514 **View:** Google Map
www.w3.org	ok	**IP:** 104.18.23.19 **Country:** United States of America **Region:** California **City:** San Francisco **Latitude:** 37.775700 **Longitude:** -122.395203 **View:** Google Map

DOMAIN	STATUS	GEOLOCATION
cdn-icons-png.flaticon.com	ok	**IP:** 104.103.64.64 **Country:** Finland **Region:** Uusimaa **City:** Helsinki **Latitude:** 60.169521 **Longitude:** 24.935450 **View:** Google Map
github.com	ok	**IP:** 140.82.121.3 **Country:** United States of America **Region:** California **City:** San Francisco **Latitude:** 37.775700 **Longitude:** -122.395203 **View:** Google Map
api.flutter.dev	ok	**IP:** 199.36.158.100 **Country:** United States of America **Region:** California **City:** Mountain View **Latitude:** 37.405991 **Longitude:** -122.078514 **View:** Google Map
192.168.101.102	ok	**IP:** 192.168.101.102 **Country:** - **Region:** - **City:** - **Latitude:** 0.000000 **Longitude:** 0.000000 **View:** Google Map

DOMAIN	STATUS	GEOLOCATION
flutter.dev	ok	**IP:** 199.36.158.100 **Country:** United States of America **Region:** California **City:** Mountain View **Latitude:** 37.405991 **Longitude:** -122.078514 **View:** Google Map

EMAILS

EMAIL	FILE
_cookie@13463476.fromsetcoo authenticationscheme@13463476.fromstring _list@0150898.of _httpparser@13463476.responsepa _compressednode@417137193.single _list@0150898.generate _typeerror@0150898._create _hashcollisionnode@417137193.fromcollis _list@0150898._ofgrowabl _list@0150898._ofefficie _growablelist@0150898._ofarray _double@0150898.fromintege _growablelist@0150898._literal3 _future@4048458.immediate _growablelist@0150898._literal _growablelist@0150898._ofother _link@14069316.fromrawpat _growablelist@0150898.withcapaci _timer@1026248._internal _growablelist@0150898._literal6 _growablelist@0150898._literal5	

EMAIL	FILE
_rawsocket@14069316._readpipe _receiveportimpl@1026248.fromrawrec _colorfilter@15065589.mode _list@0150898._ofarray _socket@14069316._readpipe _timer@1026248.periodic _growablelist@0150898._literal2 _bigintimpl@0150898.from _list@0150898.empty _list@0150898._ofother _bytebuffer@7027147._new _directory@14069316.fromrawpat _invocationmirror@0150898._withtype ngstreamsubscription@4048458.zoned _assertionerror@0150898._create _rawsocket@14069316._writepipe _nativesocket@14069316.normal _colorfilter@15065589.lineartosr _filestream@14069316.forstdin _colorfilter@15065589.srgbtoline _uri@0150898.file _growablelist@0150898._literal1 _uri@0150898.directory storationinformation@830124995.fromserial _growablelist@0150898._literal8 _file@14069316.fromrawpat _growablelist@0150898._literal4 _growablelist@0150898._ofgrowabl _growablelist@0150898.of _growablelist@0150898.generate _uri@0150898.notsimple _growablelist@0150898._literal7 _assetmanifestbin@736287047.fromstanda _growablelist@0150898._ofefficie _future@4048458.immediatee _nativesocket@14069316.pipe	lib/armeabi-v7a/libapp.so
appro@openssl.org	lib/arm64-v8a/libflutter.so

Report Generated by - MobSF v3.6.8 Beta

Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis.

Quito 18 de agosto del 2023

Señores:

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

Ciudad:

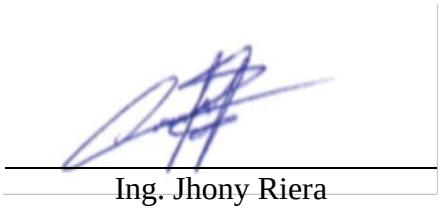
Riobamba

De mis consideraciones:

Por medio de la presente comunico que el estudiante, FRANCISCO XAVIER MESIAS FLORES, con CC: 110564053-4, han cumplido con el análisis y corrección de las vulnerabilidades en la aplicación CMED móvil en su totalidad en cumplimiento con su Trabajo de Integración Curricular: **ANÁLISIS DE VULNERABILIDADES MEDIANTE PENTESTING A LA APLICACIÓN MÓVIL DESARROLLADA PARA EL CENTRO MÉDICO DE ESPECIALIDADES "LA DOLOROSA"**

El producto software ha sido validado y está completo de acuerdo con las medidas de seguridad analizadas.

Atentamente,



Ing. Jhony Riera
Desarrollador móvil

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE

UNIDAD DE PROCESOS TÉCNICOS

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: 11/ 01 / 2024

INFORMACIÓN DEL AUTOR
Nombres – Apellidos: FRANCISCO XAVIER MESIAS FLORES
INFORMACIÓN INSTITUCIONAL
Facultad: INFORMÁTICA Y ELECTRÓNICA
Carrera: SOFTWARE
Título a optar: INGENIERIA DE SOFTWARE
f. Analista de Biblioteca responsable: Ing. Fernanda Arévalo M.