Análisis de seguridad de aplicaciones móviles wearables para sistema operativo Android

Saul Isui Lugo Martinez[1]

Instituto Nacional de Astrofísica Óptica y Electrónica, Tonantzintla (INAOE), (CONAHCYT) 72840, Puebla, México

[1]Instituto Tecnológico de Pachuca

Resumen

Este trabajo se enfoca en analizar el creciente papel de las aplicaciones móviles y la importancia de proteger la privacidad de los usuarios, explora la sinergia entre las aplicaciones móviles y los dispositivos wearables, se lleva a cabo un análisis de seguridad en una aplicación certificada y autenticada, donde su código está comprometido mediante técnicas de inyección de código malicioso con el objetivo de detectarlo y evaluar su alcance. El análisis propuesto se realiza mediante técnicas de evaluación estática. Los resultados obtenidos muestran que al inyectar código malicioso podría generar riesgos en la seguridad de la información del cualquier usuario.

Palabras clave: Wear, Android, Seguridad, Análisis estático.

Abstract

This work focuses on analyzing the growing role of mobile applications and the importance of protecting users' privacy. It explores the synergy between mobile applications and wearable devices. A security analysis is carried out on a certified and authenticated application, where its code is compromised using malicious code injection techniques in order to detect and assess its scope. The proposed analysis is conducted using static evaluation techniques. The obtained results demonstrate that injecting malicious code could pose risks to the information security of any user.

Keywords: Wear, Android, Security, Static Analysis.

1. Introducción al análisis de dispositivos móviles

En el mundo actual, las aplicaciones móviles desempeñan un papel central en nuestras rutinas diarias, brindándonos una amplia gama de servicios y funciones.[3] Estos avances tecnológicos nos permiten monitorear y gestionar diversas actividades, como ubicación, contacto, etc.,[4] Sin embargo, con el aumento en la cantidad de datos personales que se manejan a través de estas aplicaciones, surge la preocupación por la privacidad y la seguridad de los usuarios. La protección de la información personal se convierte en una prioridad clave para garantizar la confianza y el uso responsable de las aplicaciones móviles. En este contexto, también emergen los dispositivos wearables, que amplían aún más las capacidades de estas aplicaciones y su interacción con los usuarios [3]. Pero no todo es ventaja, por desgracia los dispositivos wearables, tienen brechas de seguridad, así como el software que controla a estos dispositivos.

2. Trabajos relacionados

Los trabajos relacionados desempeñan un papel crucial en el proceso de la investigación científica al proporcionar el contexto, justificación y apoyo necesario para el nuevo estudio. A continuación se resumen algunos trabajos relacionados con esta investigación.

2.1. Detecting Wearable App Permission Mismatches: A Case Study on Android Wear

El uso de dispositivos wearables está en aumento, impulsando la popularidad de las aplicaciones vestibles que se ejecutan en ellos. Estas aplicaciones están estrechamente vinculadas con las aplicaciones móviles en el dispositivo acompañante, y los permisos requeridos deben ser solicitados tanto en la versión móvil como en la wearable. Sin embargo, algunos desarrollos pueden presentar un desajuste de permisos, llevando a errores y bloqueos en las aplicaciones vestibles. En este trabajo, se propone una técnica para detectar estos desajustes y se realiza un estudio de caso en 2,409 aplicaciones gratuitas de Android Wear, identificando 73 aplicaciones vestibles con este problema. Esta investigación resalta la importancia de garantizar una integración adecuada entre aplicaciones móviles y vestibles para una experiencia óptima del usuario.[4] pero tiene la desventaja que no se tiene un conocimiento certero si estas anomalías son causadas por algún tipo de malware.

2.2. Wearable technology devices security and privacy vulnerability analysis

Este estudio se enfoca en la Tecnología wearable, que ofrece información y facilita el acceso a dispositivos principales. La seguridad y la privacidad aún presentan vulnerabilidades. Se destaca la crítica cuestión de autenticación debido a la limitada capacidad de procesamiento en estos dispositivos. Este artículo proporciona una visión general de las vulnerabilidades de seguridad y privacidad en los dispositivos wearable.[2] si bien resalta estas vulnerabilidades no hay una metodología o pasos para realizar el hallazgo de las mismas.

3. Metodología

El Marco de Ciberseguridad del NIST (National Institute of Standards and Technology) es un conjunto de directrices y mejores prácticas diseñadas para ayudar a las organizaciones a mejorar su postura de ciberseguridad.[5] Se basa en una serie de estándares, marcos y controles que permiten a las organizaciones evaluar, gestionar y mitigar los riesgos de ciberseguridad de manera efectiva. El marco se compone de cinco funciones principales: identificar, proteger, detectar, responder y recuperar, las cuales guían a las organizaciones a desarrollar y mejorar sus capacidades de ciberseguridad.[6]

3.1. NIST como metodología para el análisis de seguridad

En este contexto, este trabajo se centra en el análisis de seguridad de aplicaciones tipo wearable en el sistema operativo Android. En particular, se busca abordar los desafíos únicos que presentan las aplicaciones wearables en términos de seguridad, y cómo el Marco de Ciberseguridad del NIST puede ser un recurso valioso para guiar este análisis.

El análisis de seguridad abarca tanto el enfoque estático como el dinámico. El análisis estático se centra en examinar el código y la estructura de la aplicación sin ejecutarla, en un entorno controlado.

3.2. Entorno de pruebas

Para abordar los desafíos y garantizar la integridad y seguridad de las aplicaciones wearable, se requiere un entorno de pruebas controlado y exhaustivo. En este contexto, el presente estudio propone un entorno de pruebas que integra herramientas como Kali Linux, Ubuntu, Mobile Security Framework (MobSF) y Metasploit, para realizar un análisis de seguridad detallado y eficaz de aplicaciones tipo wear para Android. El enfoque de este entorno de pruebas controlado está en línea con las pautas del Marco de Ciberseguridad del Instituto Nacional de Estándares y Tecnología (NIST), que proporciona una base sólida para la evaluación y mitigación de riesgos en sistemas informáticos. Al utilizar el marco NIST como guía, se asegura un análisis holístico y basado en las mejores prácticas, garantizando así la confiabilidad de los resultados y la protección de la privacidad del usuario.

3.2.1. Kali Linux, MobSF y Android Studio

Kali Linux es una distribución de Linux basada en Debian GNU/Linux, diseñada específicamente para realizar pruebas de penetración y auditorías de seguridad [7], Mobile Security Framework (MobSF) es una plataforma de código abierto diseñada para realizar análisis de seguridad en aplicaciones móviles [8], incluidas las aplicaciones para dispositivos Android. Proporciona una amplia gama de funcionalidades, como análisis estático de aplicaciones, detección de vulnerabilidades y escaneo de malware. Android Studio es un entorno de desarrollo integrado (IDE) oficial para la plataforma Android.[9] Kali Linux proporcionaría un entorno de pruebas controlado y especializado para realizar pruebas de penetración y análisis de seguridad, MobSF fue utilizado para evaluar y detectar posibles vulnerabilidades y riesgos de seguridad en las aplicaciones móviles, incluyendo las aplicaciones tipo wearable. Finalmente, Android Studio fue utilizado para la emulación de distintos dispositivos Android, lo que permitió abordar las vulnerabilidades identificadas.

3.3. Malware en dispositivos y aplicaciones móviles

El malware móvil es una amenaza creciente dirigida a dispositivos móviles para acceder a datos privados.[11] Aunque no es tan común como el malware dirigido a estaciones de trabajo, representa un peligro significativo, especialmente debido al uso cada vez más frecuente de dispositivos personales para acceder a redes corporativas y personales. Existen varios tipos populares de malware para dispositivos móviles que buscan aprovechar características específicas de los smartphones o vulnerabilidades en las tabletas o dispositivos wearable.[12] Véase tabla 1.

Tabla 1 Malware más popular en aplicaciones y dispotivos móviles 2023 por Kaspersky[12]

Malware más popular en dispositivos y aplicaciones Móvil	
Malware bancario	Diseñado para comprometer a usuarios que realizan operaciones bancarias desde sus dispositivos móviles, recopilando detalles de inicio de sesión y contraseñas para enviarlos a servidores de control

Ransomware Móvil	Encripta datos importantes del usuario, como documentos, fotos y videos, y exige un rescate en Bitcoin para desbloquearlos. Si no se paga el rescate a tiempo, los archivos son eliminados o permanecen inaccesibles.
Spyware Móvil	Monitorea la actividad del usuario, registra la ubicación y roba información crítica, como nombres de usuario y contraseñas para cuentas de correo electrónico o sitios de comercio electrónico.
Malware MMS	Explota la comunicación basada en texto para enviar malware a través de mensajes de texto, incluso sin que el usuario los abra o reconozca, permitiendo a los hackers acceder al dispositivo
Adware Móvil	Va más allá de los molestos anuncios emergentes y recopilación de datos, algún adware ahora puede infectar y controlar el dispositivo, permitiendo robar información persona
Troyanos SMS	Infecta dispositivos móviles enviando mensajes SMS a números de tarifas premium, acumulando altas facturas telefónicas

3.3.1. Aplicaciones legitimas con código malicioso

Por otro lado, las aplicaciones legítimas con código malicioso inyectado son aplicaciones aparentemente legítimas y funcionales que han sido alteradas o manipuladas para incluir código malicioso.[13] Estas aplicaciones pueden pasar desapercibidas en las tiendas de aplicaciones y engañar a los usuarios para que las descarguen y las instalen en sus dispositivos.[14] Una vez instaladas, estas aplicaciones pueden realizar actividades maliciosas en segundo plano sin el conocimiento del usuario, como robar datos personales, enviar mensajes o realizar acciones no autorizadas.

3.3.2. Inyección de código malintencionado con Metasploit

Metasploit es una potente y conocida herramienta de código abierto utilizada en el ámbito de la seguridad informática y pruebas de penetración. Permite a los profesionales de la seguridad identificar debilidades en sistemas y aplicaciones, ayudando a mejorar la protección y mitigar riesgos.[15]

La inyección de código malicioso en aplicaciones legítimas es una técnica que consiste en modificar el código de una aplicación legítima para incluir código malicioso sin que el usuario o el desarrollador original sean conscientes de ello. Esto se hace con el fin de explotar vulnerabilidades y realizar acciones no autorizadas en el dispositivo donde se instala la aplicación.

En un entorno de pruebas controlado utilizando Kali Linux, se realizó un ejemplo con la aplicación Google Fit. Primero, se generó un archivo APK malicioso utilizando el payload de Metasploit (msfvenom), herramienta que permite generar código malicioso personalizado para explotar vulnerabilidades específicas. En este caso, se generó un payload que crea una conexión inversa a un servidor (LHOST) en la dirección IP 192.0.0.0 y el puerto 4545 para establecer una comunicación maliciosa con el dispositivo. Véase figura 1.

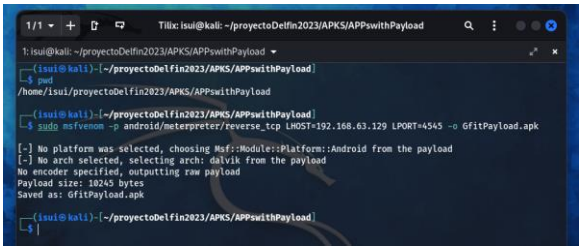

Figura 1 Creación de archivo APK con código malintencionado

Luego, se procedió a descompilar tanto la aplicación maliciosa (apk generada con el payload) como la aplicación legítima de Google Fit. La descompilación permite obtener el código fuente y los recursos de una aplicación, lo que facilita la manipulación de los archivos. A continuación, se combinaron ambas aplicaciones modificando el código de la aplicación legítima para incluir el código malicioso del payload. De esta manera, se creó una nueva APK que aparentemente es la aplicación original, pero en realidad contiene el código malicioso.

Es importante destacar que esta actividad se realizó con fines científicos, éticos y legales, en un entorno de pruebas controlado. El objetivo es evaluar la seguridad de las aplicaciones y entender cómo los ciberdelincuentes podrían intentar inyectar código malicioso en aplicaciones legítimas. Sin embargo, es fundamental resaltar que realizar esta práctica sin autorización o para fines ilegales es un delito y está prohibido por la ley.

3.4. Análisis estático

El análisis estático consiste en examinar el código de una aplicación sin ejecutarla.[16] Se busca detectar posibles llamadas de función sospechosas o permisos maliciosos. Para este trabajo se utilizó Mobile Security Framework. La información obtenida en el análisis

estático, como firmas de malware, puede ser utilizada para mejorar la detección de software malicioso basado en firmas.

Paso 1: Preparación del entorno

Instalar MobSF, tener acceso a la APK que se desea analizar. Verificar que la versión de MobSF esté actualizada y que todas las dependencias estén configuradas correctamente.

Paso 2: Inicio del análisis con MobSF

Ejecutar MobSF y cargar la APK objetivo en la interfaz. El análisis comenzará automáticamente y MobSF realizará una serie de pasos para extraer información relevante de la APK.

Paso 3: Extracción y descompresión de la APK

MobSF extraerá los contenidos de la APK y descomprimirá los archivos necesarios para su análisis estático. Esta etapa es fundamental para acceder a los recursos internos de la aplicación, como el manifiesto de Android y los archivos de código compilados (dex).

Paso 4: Análisis del manifiesto de Android

MobSF examinará el manifiesto de Android para identificar permisos requeridos, actividades, servicios, proveedores de contenido y receptores de difusión. En este paso, se busca identificar comportamientos sospechosos o permisos innecesarios que puedan indicar la presencia de un payload malicioso.

Paso 5: Análisis del código fuente y archivos dex

En esta etapa, MobSF llevará a cabo un análisis estático del código fuente de la APK y los archivos dex generados después de la compilación. El objetivo es buscar patrones o cadenas que puedan indicar la presencia de código malicioso, como la carga de un payload desde Metasploit.

Paso 6: Identificación de bibliotecas de terceros

MobSF buscará bibliotecas de terceros integradas en la APK y proporcionará información sobre sus versiones y posibles problemas de seguridad conocidos. Si el payload se ha incorporado mediante una biblioteca de Metasploit, este paso puede ayudar a identificarlo.

Paso 7: Búsqueda de firmas de Metasploit

En este paso, MobSF buscará firmas y patrones específicos asociados a Metasploit. Estas firmas podrían incluir cadenas de texto, nombres de clases o métodos que indican la presencia de código de Metasploit en la APK.

Paso 8: Análisis de recursos y activos

MobSF examinará los recursos utilizados por la APK, como imágenes, archivos de configuración y texto, en busca de cualquier indicio de un payload cargado con Metasploit.

Si el payload previamente cargado con Metasploit es detectado durante el análisis, el informe de MobSF proporcionará información sobre su ubicación en el código y sus características específicas. El analista podrá entonces tomar acciones adecuadas, como eliminar el payload malicioso y mejorar la seguridad de la aplicación, antes de su distribución y uso por los usuarios finales. Es importante recordar que el análisis estático es una de las muchas técnicas utilizadas en la evaluación de la seguridad de aplicaciones, y se debe combinar con otras técnicas, como el análisis dinámico, para obtener una imagen completa de la seguridad de la APK.

4. Resultados

A continuación se presentan los resultados del análisis estático:

Identificación del Payload: El análisis estático detecto la presencia de cadenas de texto o patrones específicos que indican actividades sospechas dichas de la aplicación.

```
<uses-sdk android:minSdkVersion="10" android:targetSdkVersion="17" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.SET_WALLPAPER" />
<uses-permission android:name="android.permission.READ_CALL_LOG" />
<uses-permission android:name="android.permission.WRITE_CALL_LOG" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
<uses-feature android:name="android.hardware.microphone" />
```

Figura 2 Permisos sopechosos encontrados

Detección de clases y métodos asociados: El análisis estático revela la existencia de clases o métodos relacionados con el payload de Metasploit, como aquellos utilizados para iniciar o manejar la sesión del /meterpreter/ dentro de la APK.

```
<data android:scheme="metasploit" android:host="my_host" />
<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.BROWSABLE" />
<action android:name="android.intent.action.VIEW" />
```

Figura 3 Detección de Metasploit en el Manifiesto de Android

Referencias a Metasploit Framework: El análisis estático descubrió referencias directas o indirectas a las bibliotecas y funciones específicas de Metasploit Framework utilizadas para cargar el payload /meterpreter/ en la APK.

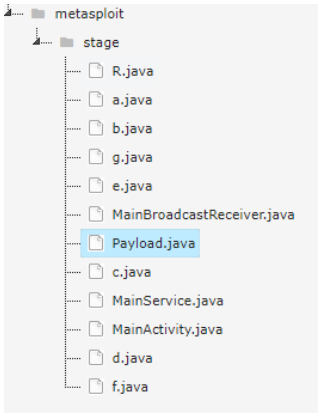

Figura 4 Payload cargado en el código fuente Java de la aplicación

Actividades sospechosas: El análisis revelo actividades sospechosas dentro de la APK, como la creación de procesos o cambios en la configuración que sugieran la presencia de un payload malicioso.

Inyección de código malicioso: El análisis revelo la inserción de código malicioso en ciertas partes de la APK, lo que indica la inclusión del payload /meterpreter/ en el proceso de compilación.

File: Payload.smali

```
1.  .class public Lcom/metasploit/stage/Payload;
2.  .super Ljava/lang/Object;
3.
```

Figura 5 Inyección de código malicioso en el directorio Smali

Es importante tener en cuenta que estos resultados solo indican la presencia del payload /meterpreter/ en la APK y no necesariamente confirman su intención maliciosa. Es responsabilidad del analista evaluar cuidadosamente el contexto y las implicaciones de estos hallazgos para determinar si se trata de una carga útil con fines legítimos o si es un indicio de una actividad maliciosa. En cualquier caso, la detección de un payload de Metasploit /meterpreter/ durante el análisis estático es un indicador significativo que requiere una acción inmediata para garantizar la seguridad y la integridad de la aplicación analizada.

5. Conclusiones

Los hallazgos de esta investigación demuestran la importancia crítica de realizar análisis de seguridad en aplicaciones móviles, especialmente considerando el creciente papel de las apps móviles y su creciente sinergia con dispositivos portátiles. Nuestro enfoque en proteger la privacidad del usuario resalta la necesidad de implementar medidas de seguridad sólidas para salvaguardar la información sensible.

A través del análisis estático de una aplicación certificada y autenticada, se detecto la presencia de código malicioso inyectado mediante técnicas de Metasploit. Este descubrimiento pone de relieve las vulnerabilidades potenciales que existen en las aplicaciones móviles, incluso aquellas que han pasado por procesos de certificación y autenticación.

La evaluación del código comprometido reveló riesgos alarmantes para la seguridad de la información de cualquier usuario. El código malicioso inyectado podría permitir el acceso no autorizado, la violación de datos y comprometer la integridad de la información del usuario. Estos hallazgos subrayan la urgencia de implementar medidas de seguridad rigurosas durante el desarrollo de aplicaciones y de llevar a cabo un monitoreo continuo a lo largo del ciclo de vida de la app. En conclusión, nuestra investigación resalta la naturaleza imperativa de priorizar la seguridad en el desarrollo de aplicaciones móviles. Al enfatizar la protección de la privacidad, realizar análisis de seguridad y aplicar medidas sólidas contra la inyección de código, se asegura la seguridad y confidencialidad de los datos de los usuarios en el mundo cada vez más interconectado de las aplicaciones móviles y los dispositivos portátiles. Mediante la comprensión de los riesgos y la implementación de estrategias preventivas, los desarrolladores y usuarios pueden adoptar con confianza los beneficios de la tecnología móvil sin comprometer la seguridad.

6. Trabajo futuro

Análisis dinámico y sandboxing: Realizar un análisis dinámico exhaustivo de aplicaciones móviles en entornos controlados (sandboxing) para evaluar su comportamiento en tiempo de ejecución. Esto permitiría identificar comportamientos maliciosos o sospechosos que no se pueden detectar mediante análisis estático.

Detección de ataques en tiempo real: Desarrollar sistemas de detección de intrusiones en tiempo real para aplicaciones móviles, que alerten sobre posibles ataques o actividades anómalas mientras el usuario interactúa con la aplicación.

Agradecimientos

Deseo agradecer al Programa de Verano de la Investigación Científica y Tecnológica del Pacifico 2023 por la oportunidad de participar del programa Delfín, al Instituto Nacional de Astrofísica, Óptica y Electrónica, (INAOE) por darme la oportunidad de realizar mi estancia en sus instalaciones, al Verano de la Investigación Científica del INAOE (VICI), al

Instituto Tecnológico Nacional de México campus Pachuca, (TECNM/ITP) por respaldarme como mi institución de procedencia, al Dr. Alfonso Martinez Cruz, por guiarme en esta estancia con su conocimiento y atención, y a mi familia por su apoyo incondicial para realizar esta estancia científica.

Referencias

[1] S. Hutchinson, Investigating Wearable Fitness Applications: Data Privacy and Digital Forensics Analysis on Android. 2022.

[2] K. W. Ching y M. M. Singh, "Wearable technology devices security and privacy vulnerability analysis", Int. J. Netw. Secur. Appl., vol. 8, núm. 3, pp. 19–30, 2016.

[3] S. Mujahid, R. Abdalkareem, y E. Shihab, "Studying permission related issues in android wearable apps", en 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2018.

[4] S. Mujahid, "Detecting wearable app permission mismatches: a case study on Android wear", en Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, 2017.

[5] National Institute of Standards and Technology, "Framework for improving critical infrastructure cybersecurity, version 1.1", National Institute of Standards and Technology, Gaithersburg, MD, 2018.

[6] "Cybersecurity Framework | NIST", 2013.

[7] G. D. Singh, The Ultimate Kali Linux Book: Perform advanced penetration testing using Nmap, Metasploit, Aircrack-ng, and Empire, 2a ed. Birmingham, Inglaterra: Packt Publishing, 2022.

[8] "Mobile Security Framework - MobSF Documentation", Mobile Security Framework - MobSF Documentation. [En línea]. Disponible en: https://mobsf.github.io/docs/#/. [Consultado: 26-jul-2023].

[9] T. Hagos, Learn android studio 3: Efficient android app development, 1a ed. Berlín, Alemania: APress, 2018.

[10] April, "What is malware + how to prevent malware attacks", Norton.com. [En línea]. Disponible en: https://us.norton.com/blog/emerging-threats/malware. [Consultado: 26-jul-2023].

[11] K. Baker, "What is mobile malware? Types & prevention tips - CrowdStrike", crowdstrike.com, 16-jun-2019. [En línea]. Disponible en: https://www.crowdstrike.com/cybersecurity-101/malware/mobile-malware/. [Consultado: 26-jul-2023].

[12] "Mobile malware", usa.kaspersky.com, 19-abr-2023. [En línea]. Disponible en: https://usa.kaspersky.com/resource-center/threats/mobile-malware. [Consultado: 26-jul-2023].

[13] J. Doe, "Deceive the heavens to cross the sea", Threatfabric.com, 17-nov-2021. .

[14] J. Umawing, "Have you downloaded that Android malware from the Play Store lately?", Malwarebytes, 01-dic-2021. [En línea]. Disponible en: https://www.malwarebytes.com/blog/news/2021/12/have-you-downloaded-that-android-malware-from-the-play-store-lately. [Consultado: 26-jul-2023].

[15] O. Valea y C. Oprisa, "Towards pentesting automation using the metasploit framework", en 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), 2020.

[16] Y. Wang y Y. Alshboul, "Mobile security testing approaches and challenges", en 2015 First Conference on Mobile and Secure Services (MOBISECSERV), 2015.

Dr. Alfonso Martinez Cruz
Asesor de proyecto
amartinezc@inaoe.mx