ANÁLISIS DE SEGURIDAD DE LA APLICACIÓN PRISM PARA SISTEMAS OPERATIVOS ANDROID, PROPIEDAD DE LA EMPRESA BARCODEAPPS, TOMANDO COMO REFERENCIA EL TOP 10 DE RIESGOS ESTABLECIDOS EN EL PROYECTO DE SEGURIDAD MÓVIL - OWASP.

DAYCITH JOSÉ GONZÁLEZ RODRÍGUEZ
CÓDIGO: 1.102.812.148

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
COROZAL
2017

ANÁLISIS DE SEGURIDAD DE LA APLICACIÓN PRISM PARA SISTEMAS OPERATIVOS ANDROID, PROPIEDAD DE LA EMPRESA BARCODEAPPS, TOMANDO COMO REFERENCIA EL TOP 10 DE RIESGOS ESTABLECIDOS EN EL PROYECTO DE SEGURIDAD MÓVIL - OWASP.

DAYCITH JOSÉ GONZÁLEZ RODRÍGUEZ
CÓDIGO: 1.102.812.148

Trabajo de grado como requisito para optar al título de:
Especialista en Seguridad Informática

Director:
Ing. Julio Alberto Vargas Fernández

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA
ESCUELA DE CIENCIAS BÁSICAS, TECNOLOGÍA E INGENIERÍA
ESPECIALIZACIÓN EN SEGURIDAD INFORMÁTICA
COROZAL
2017

Nota de aceptación:

_____

_____

_____

_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Corozal, 13 de diciembre de 2017

Principalmente a Dios, a quien le debo todo lo que soy y lo que tengo. A mis padres, mis hermanos, y mi esposa, que han tenido una gran paciencia y me han brindado su apoyo en todo momento. A mi hija, porque ha sido mi gran inspiración.

Daycith José González Rodríguez

AGRADECIMIENTOS

Gracias a Dios por darme la oportunidad de realizar con éxito este proceso, al personal BarcodeApps., especialmente a **MAURICIO FRANCO,** su Vicepresidente, por su apoyo incondicional desde el primer momento; a mi asesor de proceso, el Ingeniero **JULIO ALBERTO VARGAS FERNÁNDEZ,** Especialista en Seguridad Informática y Docente en la Universidad Nacional Abierta y A Distancia UNAD, por el tiempo, la dedicación y orientación en la elaboración del proyecto.

CONTENIDO

pág.

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

LISTA DE ANEXOS

pág.

TÍTULO

ANÁLISIS DE SEGURIDAD DE LA APLICACIÓN PRISM PARA SISTEMAS OPERATIVOS ANDROID, PROPIEDAD DE LA EMPRESA BARCODEAPPS, TOMANDO COMO REFERENCIA EL TOP 10 DE RIESGOS ESTABLECIDOS EN EL PROYECTO DE SEGURIDAD MÓVIL - OWASP.

INTRODUCCIÓN

En el presente documento se muestra el proyecto desarrollado como requisito para obtener el Título de Especialista en Seguridad Informática en la Universidad Nacional Abierta y A Distancia – UNAD. El trabajo consiste en llevar a cabo un análisis de Seguridad de la aplicación PRISM para Android, que es comercializada por la compañía BarcodeApps. Dicha aplicación permite al personal de ventas de sus clientes mostrar catálogos de productos y capturar pedidos mientras se encuentran cubriendo una ruta de ventas o durante una feria comercial.

Inicialmente se estudiará cada uno de los riesgos establecidos en el OWASP Mobile Security Project, identificando sus "rasgos" característicos, la facilidad de explotación, y el impacto que dicha explotación pueden tener a nivel técnico y de negocio; posteriormente se construye un plan de trabajo que consiste en realizar un análisis estático del código de la aplicación y un análisis dinámico de la aplicación y el sistema de Backend con el que ésta interactúa, utilizando una serie de herramientas especializadas para cada propósito.

Antes de llevar a cabo el análisis de seguridad, fue necesario presentar el plan de trabajo y solicitar aprobación por parte de la empresa, que muy amablemente brindó dicha aprobación a través de un documento escrito, el cual se adjunta en el Anexo B. En el documento se evidencia el consentimiento para realizar las pruebas de penetración, además de las condiciones y el periodo de tiempo en el que se autorizan dichas pruebas.

Una vez identificados los posibles problemas de seguridad de la aplicación y el servidor con el que interactúa, se plasman en un documento con las evidencias correspondientes, consignando además las recomendaciones y medidas que deben ser tomadas por parte del equipo de desarrollo e infraestructura de la empresa.

1. PLANTEAMIENTO DEL PROBLEMA

En la actualidad el mercado de dispositivos móviles ha tenido un crecimiento exponencial ofreciendo mejores características y funcionalidades, con tecnología que cada vez más se asemeja a la de un ordenador, y con precios asequibles para usuarios y empresas[1]. La facilidad de adquisición de dispositivos móviles de gama media y alta, acompañado de planes de datos, permiten a las personas acceder a información en internet y realizar actividades de su trabajo o entretenimiento. La potencia y portabilidad de los dispositivos móviles también permiten a las empresas de tecnología desarrollar soluciones de software que pueden ser instaladas en estos dispositivos, y mejorar los procesos que actualmente se realizan de una forma manual, o utilizando dispositivos y tecnologías menos flexibles. Esto sin duda es un gran beneficio para personas y empresas, pero también implica nuevos retos de seguridad; los delincuentes informáticos han desarrollado técnicas para cometer delitos y acceder a información confidencial de las personas y empresas.

Como ejemplo de empresa que ha aprovechado el avance de tecnología móvil se encuentra La compañía Canadiense BarcodeApps, la cual ofrece soluciones personalizadas a pequeñas y medianas empresas, permitiéndoles una fácil y efectiva gestión de sus ventas e inventarios. Una de las más importantes es la aplicación de nombre “Prism”, cuya versión web se desarrolló en el año 2005, y desde entonces ha venido mejorando cada día, hasta el punto de estar disponible para iPhone, iPad, y dispositivos Android desde su versión 4.0 (ICE CREAM SANDWICH).

La aplicación está diseñada para que el personal de ventas de pequeñas y medianas empresas[2] (a los que BarcodeApps denomina Manufacturers) pueda ofrecer un completo catálogo de productos y capturar pedidos mientras cubren una ruta de venta o si se encuentran en una feria comercial. La aplicación puede llevar a cabo la mayoría de funcionalidades sin necesidad de estar conectada a internet. Los pedidos pueden ser capturados y almacenados en el dispositivo, para luego ser enviados a un servidor central, donde se procesan y se integran con algunos de los

[1] INTERNATIONAL DATA CORPORTATION. Smartphone Volumes Expected to Rebound in 2017 with a Five-Year Growth Rate of 3.8%, Driving Annual Shipments to 1.53 Billion by 2021, According to IDC. 2017. [En línea][Citado el 21 de octubre, 2017]. Disponible en internet: https://www.idc.com/getdoc.jsp?containerId=prUS42334717

[2] BARCODEAPPS. Prism Features and Glance. [En línea][Citado el 21 de octubre, 2017]. Disponible en internet: http://prism.basisinventory.com/

paquetes de contabilidad más populares en ese país, donde se sigue con los procesos contables y comerciales.

La aplicación trabaja principalmente con dos tipos de usuarios: Sales Rep (vendedores) y Customers (clientes), quienes pueden acceder a funcionalidades específicas y visualizar información de acuerdo a su rol y al grado de segmentación establecido por la compañía distribuidora. Los vendedores pueden ver información consolidada de pedidos y una mayor cantidad de productos que los clientes, además de poder realizar pedidos para un determinado cliente, mientras que este último sólo puede ver cierta cantidad de productos y realizar pedidos a nombre propio.

Si bien es cierto que la aplicación ha cumplido con los requerimientos funcionales a lo largo de los últimos 10 años, la naturaleza de la misma, los planes de integración con pasarelas de pagos, la expansión del negocio hacia otros mercados y países, y la legislación para protección de datos existente en estos países, como la Ley 1581 en Colombia; han hecho que la empresa se plantee algunas inquietudes en cuanto al nivel de seguridad de la aplicación y el sistema de backend con el que ésta interactúa. Una de las principales necesidades de la empresa BarcodeApps es conocer si las credenciales de los usuarios se están almacenando adecuadamente en el dispositivo y si existe algún riesgo de robo de identidad al producirse una pérdida del dispositivo en el que se ha instalado la aplicación, teniendo en cuenta que en caso de no contar con internet, el proceso de autenticación de usuarios y la captura de pedidos se puede realizar en base al sistema de almacenamiento local, que en este caso se trata de una base de datos SQLITE.

El otro interrogante tiene que ver con la necesidad de saber si los procesos de comunicación con los Web Services son lo suficientemente seguros. A pesar de que todo el proceso de facturación de los pedidos que se toman a través de la aplicación es responsabilidad de las compañías clientes de BarcodeApps, que generalmente se apoyan en su propia logística y software específicos para este propósito; dentro de los planes de BarcodeApps se encuentra la inclusión de sistemas de pagos que permitan capturar y facturar los pedidos, lo que implica que se tenga que transmitir información mucho más sensible, como información de tarjetas de crédito.

1.2 FORMULACIÓN DEL PROBLEMA

¿Cómo el realizar un análisis de seguridad a la aplicación PRISM para Sistemas Operativos Android, propiedad de la empresa BarcodeApps, y el generar las respectivas recomendaciones de solución para que la empresa se encargue de aplicarlas, pueden aportar al fortalecimiento de la seguridad informática de la compañía, para que esta ofrezca a sus clientes un mayor nivel de confianza en su aplicación?

2. JUSTIFICACIÓN

Como se mencionó anteriormente, uno de los productos más importantes de BarcodeApps es Prism en sus versiones web y móvil, y la compañía ya ha tomado conciencia del hecho que la aplicación además de cumplir con los requerimientos funcionales, también debe cumplir con unas medidas de seguridad mínimas. Debido a la normatividad que se ha desarrollado para la protección de datos personales, los planes de integración con pasarelas de pagos, y los propósitos de expansión del negocio a países como Colombia, se hace necesario realizar un análisis de la aplicación y del sistema de backend del cual depende.

Al ser una aplicación que captura información de compradores, es necesario que a dicha información se le dé un manejo adecuado, no solo para mantener la integridad de la misma información, sino para cumplir con normas que exigen a las empresas manejar y proteger los datos de los consumidores, entre las que se destaca la Ley 1581 del 2012[3]. Específicamente la aplicación debe tratar y proteger adecuadamente los datos que son almacenados en el dispositivo y que son intercambiados directamente con el sistema de backend alojado en un servidor en la nube, e indirectamente con otros sistemas, como es el caso de algunos paquetes contables. Existe la posibilidad de poner en riesgo la integridad, no sólo de los datos gestionados dentro de la aplicación, sino de la información manipulada en estos sistemas externos, y que en teoría pueden ser considerados como ajenos a la aplicación móvil.

Como un aliciente más, se encuentra la posibilidad de integración con pasarelas de pagos, lo que implica el manejo de información mucho más delicada que la que se maneja actualmente. En estos momentos al capturar los pedidos de un cliente, se requiere información de la orden y datos básicos de la persona, como lo es el nombre, dirección de residencia y entrega, correo electrónico, entre otros; pero al incluir un sistema de pagos obviamente se necesitará el manejo de información mucho más delicada, como un número de tarjeta de crédito. Adicionalmente, estos sistemas de pago tienen unas condiciones y políticas de seguridad que deben ser cumplidos por la aplicación y el sistema de backend, por lo que la empresa debe tomar las medidas necesarias antes de aspirar a realizar una integración con estos sistemas.

[3] MINISTERIO DE COMERCIO, INDUSTRIA Y TURISMO. Decreto Número 1377 de 2013. 2013. [En línea][Citado el 21 de octubre, 2017]. Disponible en: https://www.mintic.gov.co/portal/604/articles-4274_documento.pdf

Otro hecho a resaltar, es que la empresa tiene dentro de sus planes una expansión hacia nuevos mercados y países, lo que conllevará a una mayor cantidad de información a manejar y a un mayor conocimiento de la empresa y de la herramienta por parte del público, entre los cuales pueden encontrarse personas y entidades (por ejemplo, la competencia) con malas intenciones.

Por todo lo mencionado hasta el momento, se puede decir que es de vital utilidad e importancia realizar un análisis del estado de la aplicación Prism para Android y establecer el nivel de seguridad que esta puede brindar actualmente, para que posteriormente se tomen las medidas de seguridad pertinentes. Se ha decido utilizar una metodología basada en el proyecto de Seguridad Mobile OWASP, por ser uno de los más reconocidos a nivel internacional y que se encuentra en constante actualización, definiendo periódicamente el listado de riesgos más significativos y que podrían estar presentes en la aplicación Prism para Android.

El OWASP Mobile Security Project, no solo tiene en cuenta el funcionamiento y nivel de protección ofrecido por la aplicación móvil, también considera los riesgos implícitos en los canales de distribución de la misma y el alojamiento de datos en la nube y los sistemas externos que de alguna manera interactúan con dicha aplicación, lo que permite planear y ejecutar un análisis completo de todo el sistema que compone la aplicación.

3. OBJETIVOS

3.1 GENERAL

Realizar un análisis de seguridad de la aplicación Prism para Sistemas Operativos Android, propiedad de la empresa BarcodeApps, tomando como referencia el Top 10 de riesgos establecidos en el Proyecto de seguridad Móvil OWASP (OWASP Mobile Security Project).

3.2 ESPECÍFICOS

- Identificar los componentes que conforman la arquitectura del Sistema Operativo Android.

- Identificar los principales vectores de ataques, el nivel de explotación, y los posibles impactos de cada uno de los riesgos establecidos en el Top 10 de riesgos del OWASP Mobile Security Project.

- Definir las herramientas a utilizar en el proceso de identificación de vulnerabilidades en la aplicación.

- Realizar el análisis de la aplicación, utilizando las herramientas definidas para cada uno de los propósitos.

- Presentar el estado de seguridad de la aplicación evaluada, junto a las recomendaciones y medidas de seguridad que se deben tomar.

4. MARCO REFERENCIAL

4.1 REFERENTE TEÓRICO

Con el objetivo de definir estándares de desarrollo de código seguro varias compañías de todo el mundo, como Siemens en Alemania, Tata en India, Nomura Research en Japón, CIBC en Londres y las americanas Kaiser Permanente, Boeing, Cisco, Symantec, Intel y American Express, se unieron a esta iniciativa. El proyecto pretende no sólo mejorar los procesos en cada una de las empresas sino el desarrollo de software en sí. Todas tienen como patrón común que sus desarrollos dependen de la calidad del software, lo que permitirá además que los clientes reciban productos seguros, ya sean desarrollados por terceros o propios[4].

A este grupo de empresas se unieron también desarrolladores de otras organizaciones y universidades, como Amazon, Secure Compass, Universidad Carnegie Mellon y el equipo OWASP, para dar lugar al "Concilio de Programación segura" que, desde el 2007, se reúne para definir estándares y documentos que indiquen las habilidades necesarias para escribir código seguro[5].

Los estándares y documentos generados por este grupo de organizaciones serán los que marcarán el camino para el proceso de investigación que se realizará. Pero sin duda alguna, el referente más importante será el equipo OWASP, pues es la comunidad que más se ha enfocado en la seguridad de aplicaciones web y móviles. Entre sus publicaciones más destacadas se encuentra OWASP Mobile Security Project – Top Ten Mobile Risks, el cual es un documento en el que periódicamente se especifican los diez riesgos de seguridad más importantes que afectan las aplicaciones móviles.

Teniendo en cuenta el listado final de establecidos en el Top 10 de riesgos del proyecto OWASP, se espera establecer un plan de trabajo en el que inicialmente se identifique el conjunto de herramientas que permitan realizar un análisis estático y

[4] CEBALLOS, Julián y MARULANDA, Cesar. Una Revisión de metodologías seguras en cada fase del ciclo de desarrollo de desarrollo de software. 2012. [En línea][Revisado el 25 de mayo, 2016] Disponible en internet: http://web.usbmed.edu.co/usbmed/fing/v3n1/v3n1a2.pdf
[5] Ibíd, p. 18.

dinámico de la aplicación, para luego tratar de identificar la presencia de alguno de estos riesgos, y finalmente establecer el estado de la aplicación en cuanto a seguridad, teniendo en cuenta el grado de facilidad de explotación y los impactos que esto pueda traer para el sistema o aplicación misma y para la organización.

4.2 REFERENTE CONCEPTUAL

4.2.1 Sistema Operativo Android. Android es el nombre del Sistema Operativo móvil propiedad de la compañía americana Google; está basado en Linux y tiene una sintaxis Java. La principal característica de este sistema operativo es que es completamente libre. Es decir, ni para programar en este sistema ni para incluirlo en un teléfono hay que pagar nada. Y esto lo hace muy popular entre fabricantes y desarrolladores, ya que los costes para lanzar un teléfono o una aplicación son muy bajos.

4.2.2 Actualizaciones de Android. Google trabaja constantemente en realizar actualizaciones para Android. Por lo menos una vez al año sale una nueva versión de dicho Sistema Operativo. Las versiones generalmente vienen con un código numérico y un nombre que hasta ahora ha sido tema relacionados con dulces y postres[6]:

Android 1.5 Cupcake	Android 4.1 Jelly Bean
Android 1.6 Donut	Android 4.2 Jelly Bean
Android 2.1 Eclair	Android 4.3 Jelly Bean
Android 2.2 Froyo	Android 4.4 KitKat
Android 2.3 Gingerbread	Android 5.0 Lollipop
Android 3.2 Honeycomb	Android 6.0 Marshmallow
Android 4.0 Ice Cream Sandwich	Android 7.0 Nougat

[6] ANDROID DEVELOPERS. Paneles de control. [En línea][Citado el 16 de mayo, 2016]. Disponible en https://developer.android.com/about/dashboards/index.html

Por su parte los fabricantes de dispositivos móviles suelen realizar modificaciones a las versiones de Android para cambiar aspectos gráficos o incluir sus propias aplicaciones o drivers específicos del dispositivo. Como ejemplo de esto se puede citar el Touchwiz UI de Samsung, el Timescape UI de Sony, el Motoblur de Motorola, o el LG Optimus UI de LG; los cuales son skins que permiten que el sistema operativo Android instalado en los dispositivos de estos fabricantes luzcan diferentes a los demás[7] [8].

La versión oficial de Android (la cual se utiliza como base) junto a las modificaciones realizadas por un fabricante es lo que se conoce como el PDA del dispositivo, este a su vez hace parte del firmware del dispositivo[9].

4.2.3 Arquitectura de Android. A continuación, los principales componentes de la arquitectura del sistema Operativo Android:

4.2.3.1 Kernel. El núcleo del sistema operativo Android está basado en el kernel de Linux, similar al que puede incluir cualquier distribución de Linux, como Ubuntu, solo que adaptado a las características del hardware en el que se ejecutará Android, es decir, para dispositivos móviles.[10]

El núcleo actúa como una capa de abstracción entre el hardware (HAL) y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores.[11]

[7] GOGI, Rahul. What are the main differences between stock Android and CyandohendMod. [En línea][Citado el 17 de mayo, 2016] Disponible en internet: https://www.quora.com/What-are-the-main-differences-between-stock-Android-and-CyanogenMod

[8] ANDROID EXPERTOS. ¿Android stock o modificado? ¿Cuáles son las diferencias?. 2014. [En línea][Citado el 16 de mayo, 2016]. http://www.androidexperto.com/aprender-android/android-stock-modificado-diferencias/

[9] ANDROIDPIT. ¿Qué es el número del firmware?. [En línea][Citado el 16 de mayo, 2016]. http://www.androidpit.es/que-es-numero-firmware

[10] INGENIERÍA COGNITIVA. Arquitectura general Android. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: http://ingenieriacognitiva.com/developer/cursos/AndroidBasico/chapters/c2.php

[11] Ibíd.

El kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos y elementos de comunicación.

4.2.3.2 Librerías. Son paquetes que proporcionan un conjunto de funcionalidades básicas y repetitivas, generalmente son desarrolladas por los fabricantes utilizando el lenguaje C/C++. Entre las librerías que habitualmente se incluyen en Android, se encuentran OpenGL (motor gráfico), Bibliotecas multimedia (formatos de audio, imagen y video), Webkit (navegador), SSL (cifrado de comunicaciones), FreeType (fuentes de texto), SQLite (base de datos), entre otras[12].

4.2.3.3 Entorno de ejecución. Al mismo nivel que las librerías se encuentran el entorno de Ejecución, cuyo componente principal es la máquina virtual Dalvik. Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación[13].

4.2.3.4 Framework de aplicaciones. En esta capa se provee a los desarrolladores un conjunto de servicios de alto nivel en forma de Clases Java, las cuales pueden ser incluidas en las aplicaciones que se desarrollen. Algunas de los servicios proporcionados en esta capa son el Gestor de Actividades, Proveedores de Contenidos, Administrador de Recursos, Administrador de Notificaciones, entre otros[14].

4.2.3.5 Aplicaciones. En esta última capa se encuentran las aplicaciones preinstaladas y las que el mismo usuario vaya instalando. Estas aplicaciones hacen uso de las API's y librerías proporcionadas en las demás capas[15].

[12] Ibíd.
[13] Ibíd.
[14] Ibíd.
[15] Ibíd.

4.2.4 Vulnerabilidad. Según las OWASP una vulnerabilidad no es más que la ausencia o debilidad de un control[16].

4.2.5 Amenaza. Evento cuya ocurrencia podría impactar en forma negativa en la organización (Las amenazas explotan o toman ventaja de las vulnerabilidades)[17].

4.2.6 Riesgo. Combinación de probabilidad de ocurrencia e impacto de una amenaza[18].

4.2.7 Análisis estático. Un análisis estático de Software es aquel que se lleva a cabo sin la ejecución de la aplicación que está siendo analizada. Generalmente el proceso se realiza sobre una versión del código fuente, utilizando una herramienta automática.

4.2.8 Análisis dinámico. El Análisis dinámico de software es un tipo de análisis que supone la ejecución del programa y observar su comportamiento.

Para que el análisis dinámico resulte efectivo el programa a ser analizado se debe ejecutar con los suficientes casos de prueba como para producir un comportamiento interesante, se pueden usar varias estrategias de pruebas de software para lograr esto, tales como cobertura de código o simplemente programas conocidos como fuzzers que ayudan a asegurar que una porción adecuada del conjunto de posibles comportamientos del programa ha sido observada. Otras herramientas en vez de probar casos de pruebas buscan a otros tipos de deficiencias en el software.

[16] OWASP. Category: Vulnerability. 2016. [En línea][Citado el 28 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Category:Vulnerability

[17] RACCIATI. Hernán. Modelado de Amenazas, una introducción. 2012. [En línea][Revisado el 22 de mayo, 2016] Disponible en internet https://www.owasp.org/images/8/8e/HRacciatti_ModeladodeAmenazas.pdf

[18] Ibíd.

4.2.9 Características de un Software Seguro. CEBALLOS y MARULANDA[19] definen algunas de las condiciones que debe cumplir un sistema seguro:

- Que sea poco vulnerable y libre de defectos tanto como sea posible[20].

- Que limite el resultado de los daños causados por cualquier ataque, asegurando que los efectos no se propaguen y que puedan ser recuperados tan rápido como sea posible[21].

- Que continúe operando correctamente en la presencia de la mayoría de ataques[22].

CEBALLOS *et al*[23] también plantean las propiedades que generalmente debe reflejar un software que ha sido desarrollado con seguridad a través de su desarrollo:

4.2.9.1 Ejecución Predecible. La certeza de que cuando se ejecute funcione como se había previsto. Reducir o eliminar la posibilidad de que entradas maliciosas alteren la ejecución o las salidas[24].

4.2.9.2 Fiabilidad. La meta es que no haya vulnerabilidades que se puedan explotar[25].

[19] CEBALLOS, Julián y MARULANDA, Cesar. Op. cit,. p. 17
[20] Ibíd., p 17.
[21] Ibíd., p. 17.
[22] Ibíd., p. 17.
[23] Ibíd., p. 17.
[24] Ibíd., p. 17.
[25] Ibíd., p. 17.

4.2.9.3 Conformidad. Actividades planeadas, sistemáticas y multidisciplinarias que aseguren que los componentes y productos de software están conforme a los requisitos, procedimientos y estándares aplicables para su uso específico[26].

Algunas propiedades fundamentales que son vistas tanto como atributos de seguridad, como propiedades del software son:

4.2.9.4 Confidencialidad. Debe asegurar que cualquiera de sus características incluyendo sus relaciones con ambiente de ejecución y sus activos y/o contenidos no sean accesibles para no autorizados[27].

4.2.9.5 Integridad. El software y sus activos deben ser resistentes a subversión. Subversión es llevar a cabo modificaciones no autorizadas al código fuente, activos, configuración o comportamientos, o cualquier modificación por entidades no autorizadas. Tales modificaciones incluyen sobre-escritura, corrupción, sabotaje, destrucción, adición de lógica no planeada —inclusión maliciosa— o el borrado. La integridad se debe preservar tanto en el desarrollo como durante su ejecución[28].

4.2.9.6 Disponibilidad. El software debe ser funcional y accesible por usuarios autorizados cada vez que sea necesario. Al mismo tiempo, esta funcionalidad y estos privilegios deben ser inaccesibles por usuarios no autorizados[29].

Dos propiedades adicionales, comúnmente asociadas con usuarios humanos, se requieren en entidades de software que actúan como usuarios, por ejemplo, agentes proxy y servicios Web.

[26] Ibíd., p. 17.
[27] Ibíd., p. 17.
[28] Ibíd., p. 17.
[29] Ibíd., p. 17.

4.2.9.7 Responsabilidad. Todas las acciones relevantes de seguridad del software o de los usuarios se deben almacenar y rastrear, con atribución de responsabilidad. Este rastreo debe ser posible en ambos casos, es decir, mientras y después de que la acción registrada ocurra. Según la política de seguridad para el sistema se debería indicar cuáles acciones se consideran como seguridad relevante, lo que podría hacer parte de los requisitos de auditoría[30].

4.2.9.8 No repudio. Esta propiedad le permite al software y a los usuarios refutar o denegar responsabilidades de acciones que ha ejecutado. Esto asegura que la responsabilidad no puede ser derribada o evadida. Otras propiedades influyentes en el software seguro son la exactitud, la capacidad de pronóstico, la fiabilidad y la protección, las cuales están también influenciadas por el tamaño, la complejidad y la trazabilidad del software[31].

4.2.9.9 Análisis de riesgos. Debido a las constantes amenazas, toda organización es vulnerable Riesgos debido a la existencia de vulnerabilidades[32].

4.2.10 Amenazas a los que se encuentran expuestos las aplicaciones web y móviles[33].

El software está sujeto a dos categorías generales de amenazas:

4.2.10.1 Amenazas durante el desarrollo. Un ingeniero de software puede sabotear el programa en cualquier punto de su desarrollo[34].

[30] Ibíd., p. 17.
[31] Ibíd., p. 17.
[32] Ibíd., p. 17.
[33] Ibíd., p. 17.
[34] Ibíd., p. 17,

4.2.10.2 Amenazas durante la operación. Un agresor intenta sabotear el sistema[35].

El software en la red está comprometido por el aprovechamiento de sus debilidades, las cuales se derivan de las siguientes fuentes:

- Complejidad o cambios incluidos en el modelo de procesamiento[36].

- Suposiciones incorrectas del ingeniero, incluyendo las relacionadas con la capacidad, las salidas, el comportamiento de los estados del ambiente de ejecución el software o con entradas esperadas de entidades externas[37].

- Implementación defectuosa en el diseño o en los requisitos de interfaces del software con otras entidades externas o en ambiente de ejecución de los componentes del software[38].

- Interacción no planeada entre componentes de software, incluyendo los de otros proveedores[39].

4.2.11 Ataques a un dispositivo móvil. Según RAMÍREZ[40]**,** los ataques a un dispositivo móvil están asociados a diferentes riesgos y vulnerabilidades que se derivan de la interacción con dicho dispositivo.

[35] Ibíd., p. 17.
[36] Ibíd., p. 18.
[37] Ibíd., p. 18.
[38] Ibíd., p. 18.
[39] Ibíd., p. 18.
[40] RAMÍREZ, Mauricio. Seguridad en aplicaciones móviles. 2013. [En línea][Citado el 22 de mayo, 2016] Disponible en internet http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/index.html

4.2.11.1. Puntos sensibles de un dispositivo móvil[41]. Los siguientes son algunos de los puntos de ataque a los que se encuentran expuestos los dispositivos móviles:

- Las credenciales y los servicios externos del dispositivo como el correo electrónico, las cuentas de bancos, etc.

- Los datos personales de los usuarios.

- Los datos de los dispositivos como los números de cuenta, números de las tarjetas y las fechas de expiración.

- Acceso al dispositivo para revisar la sim card, revisión de las conexiones telefónicas y de internet, uso del dispositivo para enviar virus, malware y procesamiento de actividades, robo de datos secretos y datos sensibles del dispositivo.

- Almacenamiento de datos, robo, revisión y modificación de claves, información de las bases de datos, archivos de configuración, archivos de las aplicaciones, las caches de los sistemas.

- Archivos binarios, realización de ingeniería inversa para entender el binario, búsqueda de las vulnerabilidades que pueden ser explotadas, incrustar credenciales y generación automática de claves.

- Plataformas móviles de enganche de las plataformas, instalación de malware, aplicaciones móviles de ejecuciones automáticas no autorizadas, las decisiones de la arquitectura de aplicaciones basadas en la plataforma.

[41] Ibíd.

- El almacenamiento de datos, los archivos binarios y la plataforma no son independientes y se encuentran relacionados entre sí.

4.2.11.2 Tipos de ataques. Los siguientes son algunos de los tipos de ataques que se pueden hacer a un dispositivo móvil[42]:

- Malwares: un software malicioso que a menudo se hace pasar como un juego, parche, o una aplicación que realiza una funcionalidad específica; pero que en realidad inicia una variedad de ataques y se extiende a otros dispositivos. Un malware puede incluir virus, troyanos, y malwares[43].

- Vulnerabilidades del navegador: son problemas relacionados con el uso del navegador del dispositivo. Un simple clic en un hipervínculo o acceder a un sitio web malicioso, puede traer como consecuencia la instalación de un malware o la ejecución de procesos dañinos para el dispositivo[44].

- Interceptación de datos: un atacante puede escuchar a escondidas las comunicaciones que entran y salen desde el dispositivo móvil. Técnicas como man-in-the-middle y Wifi Sniffing son algunas de las utilizadas por un atacante para realizar el espionaje electrónico[45].

- keyLogger: es un tipo de malware que lleva un registro de las pulsaciones realizadas en el teclado del dispositivo, que un atacante puede revisar posteriormente para tratar de identificar datos confidenciales[46].

[42] PUJA S., SHELKE C.J. A Survey on different Attacks on Mobile Devices and its Security. 2014. [En línea][Citado el 18 de mayo, 2016] Disponible en internet: http://www.ijaiem.org/volume3issue2/IJAIEM-2014-02-28-080.pdf

[43] Ibíd., p. 249.

[44] Ibíd., p. 249.

[45] Ibíd., p. 248.

[46] Ibíd., p. 249.

- Seguimiento no autorizado de la ubicación: lo cual permite a un atacante conocer las ubicaciones que frecuenta el usuario del dispositivo[47].

- Exploits de red: Esta modalidad de ataque se aprovecha de vulnerabilidades de los sistemas de conexión del dispositivo para identificar y secuestrar credenciales de usuarios y propagar malwares dentro de una red[48].

- Phishing: se trata de una estafa en la que se utiliza el correo electrónico, un popup, o un simple botón; para inducir a los usuarios a realizar una acción en la que debe proporcionar algún tipo de información confidencial[49].

- Spamming: se trata de publicidad de productos, servicios, o sitios web que llega al correo electrónico del usuario, sin que éste lo haya solicitado[50].

- Spoofing: sitios web falsificados con los que se puede capturar información del usuario o instalar algún tipo de malware en el dispositivo móvil[51].

- Pérdida o robo del dispositivo: algunas de las mayores ventajas de un dispositivo móvil son su pequeño tamaño, portabilidad, y la posibilidad de uso en cualquier lugar; pero esto también implica un mayor riesgo de que el usuario pierda su dispositivo o ser víctima de los amigos de lo ajeno. Si el dispositivo cae en manos de un atacante, fácilmente puede obtener la información que en éste se almacena[52].

[47] Ibíd., p. 249.
[48] Ibíd., p. 249.
[49] Ibíd., p. 249.
[50] Ibíd., p. 249.
[51] Ibíd., p. 249.
[52] Ibíd., p. 250.

- Exploit del día cero (ZETA): un ataque que se produce antes que una vulnerabilidad sea corregida por los desarrolladores de una aplicación o los fabricantes del dispositivo[53].

4.2.11.3 Anatomía de un ataque móvil

Figura 1. Anatomía de un Ataque móvil.

Fuente: https://www.nowsecure.com/resources/secure-mobile-development/primer/mobile-security/

Un ecosistema móvil tiene tres puntos que los atacantes buscan explotar:

- ***Ataques al dispositivo móvil**: l*os ataques centrados en el dispositivo móvil tienen que ver con problemas del hardware y software del dispositivo, como son:

  - Navegador

[53] Ibíd., p. 249.

- Correo electrónico.
- El teléfono.
- Mensajes de texto.
- Aplicaciones.
- El mismo sistema operativo.
- Bluetooth.
- Banda base de radiofrecuencia

Los ataques más comunes son los siguientes:

- Phishing: que, como se dijo anteriormente, tiene que ver con engañar al usuario para que proporcione información confidencial.

- Framing: que consiste en cargar un sitio web legítimo dentro de un iframe, que a su vez es cargado dentro de una página web maliciosa. Los atacantes suelen poner enlaces o botones con una apariencia similar al sitio web legítimo, que una vez presionado por el usuario, se ejecutará algún tipo de acción maliciosa[54].

- ClickJacking: es un tipo de ataque relacionado con el framing, en el que un atacante usa una serie de capas transparentes u opacas para ubicar botones o enlaces de sitios web maliciosos que son cargados dentro de un iframe. Los enlaces o botones son presionados por los usuarios creyendo que hacen parte del sitio web legítimo[55].

[54] RYDSTEDT, Gustav. Framing Attacks on Smart Phones and Dumb Routers: Tap-jacking and Geo-localization Attacks. [En línea][Revisado el 19 de mayo, 2016] Disponible en internet: http://seclab.stanford.edu/websec/framebusting/tapjacking.pdf

[55] OWASP. Clickjacking. 2015. [En línea][Citado el 28 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Clickjacking

- Drive-by-downloading: se trata de un tipo de malware que se descarga desde un sitio web, sin que el usuario se dé cuenta.

- Man in the mobile (MitMo): es un tipo de ataque que puede espiar los mensajes de texto, servicios OTP, y llamadas de voz para luego enviarlos al atacante[56].

- Ataques en la banda base (GSM, 3G): se ha demostrado que es posible generar un error de desbordamiento en el código que gestiona la pila de protocolos de comunicaciones y explotarlo para introducir código malicioso en el dispositivo móvil. El algoritmo A5/1 (algoritmo de cifrado para GSM) y el algoritmo A5/3 KASUMI (algoritmo de cifrado para 3G), son objeto de estudio de investigadores, y en ambos casos se ha demostrado que tienen problemas de seguridad[57].

- SmiShing: o SMS phishing, es un tipo de ataque que envía mensajes de textos a dispositivos móviles con el objetivo de inducir a los usuarios a proporcionar información confidencial[58].

- jailbreak: es un término relacionado con dispositivos de Apple, consiste en eliminar algunas restricciones impuestas por los fabricantes de los dispositivos[59]. El jailbreak permite acceder al kernel del sistema operativo y modificarlo, lo que puede dar lugar a riesgos más importantes, como una escalada de privilegios o la instalación de algún tipo de malware.

---

[56] SOLIDPADD. Man-in-the-Phone Attacks (Man-in-the-Mobile/MitMo Attacks). [En línea][Revisado el 28 de mayo, 2016] Disponible en internet: http://www.solidpass.com/threats/man-in-the-phone-mitp-attacks.html

[57] PICO, José. Amenazas de seguridad en comunicaciones de datos móviles. 2011. [En línea][Revisado el 29 de mayo, 2016] Disponible en internet: http://es.slideshare.net/eventoscreativos/amenazas-de-seguridad-en-comunicaciones-de-datos-mviles).

[58] RSA. Phishing, Vishing and Smishing: Old Threats Present new Risks. [En línea][Citado el 29 de mayo, 2016] Disponible en internet https://www.rsa.com/content/dam/rsa/PDF/h11933-wp-phishing-vishing-smishing.pdf

[59] ROUSE, Margaret. Jailbreaking. 2013. [En línea][Citado el 27 de mayo, 2016] Disponible en internet: http://whatis.techtarget.com/definition/jailbreaking

- Rooteo del dispositivo: relacionado con dispositivos móviles con el sistema operativo Android, en el que se obtiene acceso a dichos dispositivos como un usuario root, lo que permite tener un control total sobre ellos.

En este punto resulta importante resaltar las vulnerabilidades propias de los sistemas operativos y el tipo de aplicaciones móviles:

- ***Vulnerabilidades a nivel de sistema operativo***. Las vulnerabilidades en los sistemas operativos móviles son muy comunes y suelen ser el blanco de los atacantes que buscan causar un alto impacto en el sistema.

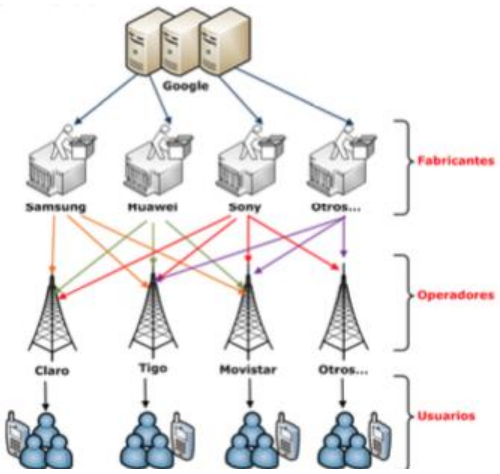
Si se habla del dispositivo Android hay que decir que Google actualiza periódicamente el sistema operativo con mejoras en funcionalidad, parches y correcciones de seguridad; pero muchas veces estas actualizaciones no coinciden con las modificaciones realizadas en el firmware del fabricante del dispositivo móvil, lo que puede generar algunos riesgos de seguridad[60].

Adicionalmente, los fabricantes de dispositivos u operadores de telefonía móvil modifican el sistema operativo para incluir información o funciones propias y estos cambios pueden generar huecos de seguridad debido a que el sistema operativo se modifica sin saber lo que sucederá en el dispositivo. En la siguiente figura se muestra el modelo de actualización del sistema operativo Android, donde intervienen Google, fabricantes de dispositivos, y los operadores de telefonía celular en Colombia[61]:

[60] COLORADO, Pedro y TORRES, Inírida. Análisis de seguridad de aplicaciones móviles nativas para el sistema operativo Android versión Jelly Bean 4.1.2 en dispositivos móviles Smartphone. 2015. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: repository.unad.edu.co/bitstream/10596/3412/1/40186727.pdf

[61] Ibíd., p. 44.

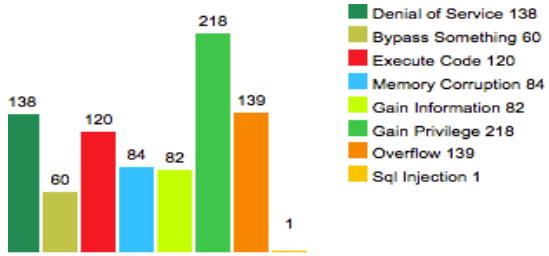
Figura 2. Modelo de actualización Android.



Fuente: repository.unad.edu.co/bitstream/10596/3412/1/40186727.pdf

De acuerdo a las estadísticas realizadas por CVE DETAILS[62], las vulnerabilidades más significativas del sistema operativo Android en el año 2016 son: Denegación de servicio, Salto de controles, (bypass something), ejecución de código sin autorización, corrupción de memoria, obtención de información, obtención de privilegios, overflow, y Sql injection; lo que se evidencia en la siguiente figura:

Figura 3. Vulnerabilidades más significativas en Android, 2016.

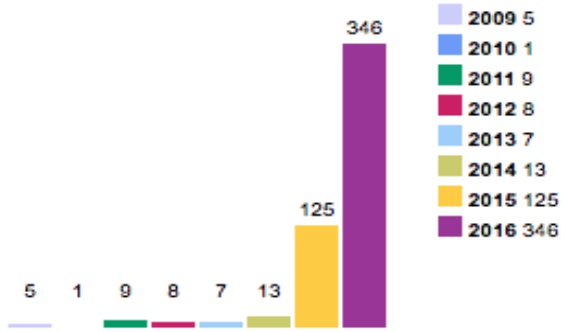


Fuente: http://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224

[62] CVE DETAILS. The Ultimate Security Vulnerability Datasource. 2016. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: http://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224

Llama la atención como el número de vulnerabilidades ha venido incrementando año tras año, siendo el 2016 el año en el que se han encontrado una gran cantidad, casi que triplicando las vulnerabilidades del 2015:

Figura 4. Estadísticas vulnerabilidades 2009 - 2016



Fuente: http://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224

- **Riesgos de seguridad de acuerdo el tipo de aplicación móvil***:* cada tipo de aplicación móvil presenta un conjunto de riesgos ligeramente distinto porque cada uno tiene un diseño y capacidades diferentes:

- **Riesgos de Seguridad en Aplicaciones móviles nativas**[63]*:* las aplicaciones móviles pueden acceder a la libreta de direcciones, obtener datos de ubicación, enviar mensajes de texto, hacer llamadas, y acceder a las redes internas.

En el caso de una aplicación móvil nativa se tienen dos tipos de riesgos. Los riesgos para el dispositivo y los riesgos para la misma aplicación. Un riesgo para la aplicación es todo aquel que pueda poner en peligro la información o la aplicación misma; un riesgo para el dispositivo es todo aquel que puede ocurrir fuera de la aplicación, como agotar la carga de la batería, enviar mensajes de texto, hacer llamadas, transferencias de minutos, de datos, etc.

[63] COLORADO, Pedro y TORRES, Inírida. Op. cit., p 45.

- **Riesgos de seguridad de aplicaciones web**[64]*:* las aplicaciones web generalmente tienen dos componentes principales: el servidor y el cliente. Las vulnerabilidades del servidor pueden estar presentes en la parte de la aplicación que se ejecuta en el servidor. Las vulnerabilidades del cliente pueden ser explotadas potencialmente dentro de la página web cuando esta es representada y ejecutada en el navegador web del dispositivo.

En el lado del servidor, este puede aceptar datos de clientes que no son de confianza y procesar dichos datos para devolver una respuesta al cliente. Estos datos no verificados pueden emplearse para acceder a una base de datos, un sistema de archivos u otras fuentes de información vital para la seguridad. Si el servidor no limpia adecuadamente los datos no verificados, estos podrían provocar el deterioro de la base de datos, exponer archivos confidenciales o abrir la puerta a daños de otra clase.

En el lado del cliente, ejecutar una página web enviada desde el servidor generalmente implica cargar la página y ejecutar JavaScript. El cliente ejecuta todo el código en el contexto de un origen específico, por lo que, si se ejecutan de algún modo datos no verificados de un atacante, el atacante goza de plena potestad para acceder a la página y modificarla. Esto significa que los atacantes pueden capturar pulsaciones de teclas, robar los datos introducidos, alterar la página o ejecutar un ataque de phishing convincente.

- **Riesgos de seguridad de aplicaciones híbridas**[65]***.*** Dado que una aplicación híbrida es una mezcla de una aplicación nativa y una aplicación web, reúne los riesgos de ambos tipos de aplicación, lo que las hace más vulnerables.

Las aplicaciones híbridas basadas en HTML5 son más susceptibles a cualquier tipo de ataque que las aplicaciones nativas, lo que puede resultar en la captura de información personal y el posterior envío a través de un software malicioso a los contactos de las víctimas en un mensaje de texto[66].

---

[64] Ibíd., p. 45.

[65] Ibíd., p. 45.

[66] MOVILION. Aplicaciones móviles híbridas: bomba de tiempo en materia de seguridad. 2014.[En línea][Citado el 16 de mayo, 2016] Disponible en internet] Disponible en internet: http://www.movilion.com/aplicaciones-moviles-hibridas-seguridad/

A diferencia de las aplicaciones nativas que sólo muestran el posible código malicioso, la aplicación basada en *HTML5*, dependiendo del *JavaScript API*, ejecuta directamente el código[67].

- **Ataques a la red del dispositivo.** Los ataques a la red tienen como objetivo las conexiones de red a los que se puede conectar el dispositivo móvil, generalmente las conexiones WIFI. Algunos de los ataques más importantes son los siguientes:

  - Escaneo de paquetes
  - Man in the Middle (MITM): en el que un atacante se interpone en la comunicación entre dos partes, lo que le permite leer la información intercambiada e incluso enviar mensajes en nombre de alguna de las dos partes.
  - SSLStrip
  - Secuestro de sesión
  - DNS Spoofing
  - Falsificación de certificados SSL
  - Denegación de servicios

- **Ataques orientados a los almacenes de datos.** Los ataques que tienen como objetivo los almacenes de datos se enfocan principalmente en el servidor web y las bases de datos remotas y locales, en los que se aprovechan vulnerabilidades del lado del servidor y la validación débil de las entradas. Los ataques más comunes son los siguientes:

  - XSS
  - CSRF

[67] Ibíd., p 45.

- Ataques de fuerza bruta
- Inyección SQL
- Ejecución de comandos del sistema operativo
- Escalamiento de privilegios
- Volcado de datos

4.2.12 OWASP. Comunidad abierta dedicada a habilitar a las organizaciones para desarrollar, comprar y mantener aplicaciones confiables. Todas las herramientas, documentos, foros y capítulos de OWASP son gratuitos y abiertos a cualquiera interesado en mejorar la seguridad de aplicaciones. Abogan por resolver la seguridad de aplicaciones como un problema de gente, procesos y tecnología porque las soluciones más efectivas incluyen mejoras en todas estas áreas[68].

Uno de los proyectos más importantes de OWASP, es el OWASP Mobile Security Project, que tiene como propósito la clasificación de los riesgos presentes en aplicaciones móviles. Para cada riesgo se define la facilidad de explotación, el predominio y nivel de detección, y los impactos que puede traer su explotación a nivel técnico y del negocio.

[68] OWASP. Sobre OWASP. 2014. [En línea][Citado el 30 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Sobre_OWASP

Figura 5. OWASP Mobile Top 10 Risks

Fuente: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks

4.2.12.1 M1 Weak Server Side Controls[69]. (Debilidad en los controles del lado del servidor de la aplicación). Como su nombre lo indica, corresponde a problemas de seguridad que no se encuentran en la aplicación móvil, sino en el servidor, en el que generalmente se encuentran sistemas con los que interactúa la aplicación, como pueden ser APIS, sistemas de Backend, Bases de datos, etc.

[69] OWASP. Mobile Top 10 2014-M1. 2014. [En línea][Revisado el 16 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M1

Tabla 1. M1 Weak Server Side.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Fácil	Común	Promedio	Grave	Específico de la aplicación y negocio

Fuente: El autor.

Agentes de amenazas: cualquier fuente de información poco fiable para el sistema de backend del que depende la aplicación, por ejemplo:

- Usuarios
- Malwares
- Otra aplicación dentro del dispositivo móvil.

Para que esta vulnerabilidad pueda ser aprovechada por un atacante, la organización debe exponer una llamada a un API o servicio web, el cual presenta algunas de las vulnerabilidades expuestas en el OWASP Top 10 del lado del servidor.

Los impactos técnicos y del negocio dependerán en gran medida del tipo de vulnerabilidad encontrado en el lado del servidor.

Son varios los factores que pueden dar a lugar a una proliferación del lado del servidor, estos factores incluyen:

- Afán por salir al mercado.
- La falta de conocimiento sobre la seguridad en nuevos lenguajes.
- Uso de frameworks de desarrollo que no dan prioridad a la seguridad.
- En muchos casos el desarrollo del lado del servidor es hecho por un tercero.
- Presupuestos de seguridad inferior para aplicaciones móviles.
- Suposición de que el sistema operativo móvil asume la plena responsabilidad de la seguridad.
- Debilidad debido al desarrollo de plataforma cruzada y compilación.

4.2.12.2 M2 Insecure Data Storage[70]. (Almacenamiento de datos inseguro): el almacenamiento de datos inseguro puede resultar en la pérdida de datos para un usuario - por ejemplo - que pierde su teléfono; o para varios usuarios si por ejemplo - una aplicación está asegurada de forma inadecuada, dejando a todos los usuarios en situación de riesgo.

Tabla 2. M2 Insecure Data Storage.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Fácil	Común	Promedio	Grave	Específico de la aplicación y negocio

Fuente: El autor

[70] OWASP. Mobile Top 10 2014-M2. 2014. [En línea][Revisado el 16 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M2

Agentes de amenazas:

- Un atacante que tiene acceso a un celular perdido o robado.
- Un malware.
- Una aplicación que ha sido modificada y reempaquetada para realizar alguna acción maliciosa.

Cuando un atacante tiene acceso físico a un dispositivo móvil puede utilizar herramientas de libre uso, acceder al sistema de directorio y/o almacenamiento del dispositivo para obtener información personal del usuario. Con un simple rooting o jailbreaking del dispositivo, el atacante puede romper cualquier estrategia de encriptación en el dispositivo. Un atacante también puede desarrollar un malware para hacer este trabajo automáticamente.

Desde el punto de vista técnico, una vulnerabilidad frente a este riesgo puede resultar en la pérdida de información de uno o varios usuarios, los datos más importantes que puede perder son los siguientes:

- Nombres de usuarios.
- Contraseñas
- Tokens de autenticación
- Cookies
- Datos de localización
- UDID/IMEI: identificador del dispositivo, datos de conexión de red.
- Información personal: fecha de nacimiento, dirección, información social, hasta datos de una tarjeta de crédito.
- Datos de la aplicación: registros de logs, información de depuración, mensajes de aplicación en caché, historial de transacciones.

Desde el punto del negocio, este riesgo puede resultar en alguna de las siguientes situaciones:

- Robo de identidad
- Fraude
- Daño de reputación
- Violación de políticas externas
- Pérdida de material

Es importante conocer los datos que una aplicación almacena y la forma como esto se hace en un dispositivo móvil. Algunos almacenamientos de datos que pueden significar riesgo son:

- Bases de datos SQLite
- Archivos de Log
- Los archivos plist (IOS)
- Archivos Manifest XML (Android)
- Almacenes de datos binarios
- Cookies
- Tarjeta SD

4.2.12.3 M3 Insufficient Transport Layer Protection[71]. (Protección insuficiente en la capa de transporte): En una aplicación que requiere el intercambio de información con un sistema de Backend externo debe transmitir y recibir información haciendo uso de la red móvil y de internet. Un atacante podría hacer uso de herramientas que le permitan obtener datos sensibles mientras estos viajan por la red.

Tabla 3. M3 Insufficient Transport Layer Protection.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Difícil	Común	Fácil	Moderado	Específico de la aplicación y negocio

Fuente: El autor.

Agentes de amenaza:

- Un atacante con acceso a la red local con usuario que tiene el dispositivo conectado su dispositivo móvil al wifi.
- Dispositivos de transmisión o de red: routers, antenas de telefonía celular, un proxy, etc.
- Un malware en el dispositivo móvil.

Llegar a escuchar el tráfico en una red puede ser un trabajo difícil, pero una vez logrado por un atacante, éste puede llegar a hacer mucho daño.

En muchas ocasiones los desarrolladores de aplicaciones móviles no protegen adecuadamente el tráfico de red. Muchos utilizan SSL/TLS para realizar un proceso

[71] OWASP. Mobile Top 10 2014-M3. 2014. [En línea][Revisado el 16 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M3

de autenticación, pero no en otros procesos; y este hecho puede llevar a que el atacante puede llevar a la exposición de datos en otros procesos, incluyendo los datos de sesión que fueron obtenidos adecuadamente durante una autenticación segura.

Como impactos técnicos se tiene la posibilidad que una vulnerabilidad frente a este riesgo puede exponer los datos de un usuario y conducir al robo de su cuenta. Si el usuario tiene un perfil de administrador las consecuencias pueden ser devastadoras para el sistema en general. Este defecto también puede llevar a ataques de phishing y MITM para usuarios con menos permisos en el sistema, pero con información sensible, como puede ser su tarjeta de crédito.

Para organizaciones en la que se manejan datos sensibles, este riesgo como mínimo redunda en la violación de la privacidad de sus usuarios, lo que puede resultar en:

- Robo de identidad
- Fraude dentro de la organización y/o daño de su reputación.

Para averiguar si una aplicación tiene suficiente protección de la capa de transporte, se puede revisar el tráfico de las aplicaciones a través de un proxy y dar respuesta a interrogantes como:

- ¿Son todas las conexiones, incluyendo a servicios de terceros, debidamente encriptadas?
- ¿Se cuenta con certificados SSL actualizados, firmados, y aceptados por entidades reconocidas?

4.2.12.4 M4 Unintended Data Leakage[72]. (Fuga de datos involuntaria): debido a que las aplicaciones móviles tienen que interactuar con un sistema operativo, Frameworks, entorno de compilación, hardware, que no puede ser controlado por los desarrolladores, es posible que se presenten problemas de seguridad relacionados con estos ítems y que pueden comprometer a la aplicación y la información que esta maneja.

Tabla 4. M4 Unintended Data Leakage.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Fácil	Común	Fácil	Grave	Específico de la aplicación y/o negocio

Fuente: El autor.

Agentes de amenazas:

- Malware
- Aplicaciones modificadas para incluir código malicioso, reempaquetadas y distribuidas nuevamente.
- Atacante con acceso físico al dispositivo.

Un atacante puede aprovechar un acceso físico al dispositivo y utilizar herramientas forenses de libre acceso para hacer ingeniería inversa de la aplicación, modificarla y re-empaquetarla.

[72] OWASP. Mobile Top 10 2014-M4. 2014. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M4

El atacante puede usar también un Malware para que éste se conecte a un API o servicio de backend que es permisivo y tiene acciones documentadas para efectuar acciones sin consentimiento del usuario.

Una fuga de datos involuntaria puede ocurrir cuando los desarrolladores de aplicaciones móviles guardan inadvertidamente información sensible en ubicaciones del dispositivo que pueden ser fácilmente accesibles para otras aplicaciones o usuarios. Las rutinas programadas por el desarrollador pueden tener efectos secundarios que da lugar a que se almacene información en lugares desprotegidos y perfectamente accesibles por las demás aplicaciones dentro del dispositivo, incluyendo malwares. Generalmente este tipo defectos son consecuencias del trabajo de desarrolladores con poca experiencia y desconocimiento de la forma en que trabaja el sistema operativo subyacente.

Los desarrolladores deben hacer un esfuerzo por conocer los procesos internos de los cuales hace uso la aplicación, como pueden ser:

- La forma en que el Sistema Operativo y frameworks utilizados guardan los datos de caché, las imágenes, las teclas presionadas, registro y buffers.

- La cantidad de información mostrada y almacenada sobre anuncios, analítica, redes sociales, imágenes, etc.

Las vulnerabilidades frente a este riesgo pueden tener como consecuencia técnica la extracción de datos sensibles de la aplicación a través de malware, modificación de la misma aplicación, y/o la utilización de herramientas forenses.

Para organizaciones en la que se manejan datos sensibles, este riesgo como mínimo redunda en: la violación de la privacidad de sus usuarios, lo que puede resultar en:

- Violación de privacidad
- Violaciones PCI (Payment card Industry).
- Fraude dentro de la organización y/o daño de su reputación.

4.2.12.5 M5 Poor Authorization and Authentication[73]. (Autenticación y autorización pobres): las aplicaciones y los sistemas con que se conectan deben ser protegidos adecuadamente con las mejores prácticas de autorización y autenticación. Esto asegura que los dispositivos, usuarios, y sistemas están autorizados para transferir datos dentro del flujo de trabajo de la aplicación y si se llegan a conectar dispositivos no autorizados, los usuarios y los scripts son identificados y bloqueados.

Tabla 5. M5 Poor Authorization and Authentication.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Fácil	Común	Fácil	Grave	Específico de la aplicación y/o negocio

Fuente: El autor.

Agentes de amenazas:

- Malware
- Botnets
- Ataques por fuerza bruta

[73] OWASP. Mobile Top 10 2014-M5. 2014. [En línea][Citado el 18 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M5

Un atacante puede utilizar una de las herramientas mencionadas en el listado anterior, y si detecta un pobre o nulo esquema de autenticación y autorización, empieza a interactuar directamente con el sistema de backend de la aplicación, de manera anónima y sin necesidad de interactuar con la aplicación móvil.

Pero los problemas pueden no estar solo en el sistema de backend del que depende la aplicación móvil. Los esquemas de autenticación y autorización de una aplicación móvil son diferentes a los sistemas de autenticación web tradicionales debido a la disponibilidad. Mientras que en un sistema de autenticación web los usuarios se encuentran todo en tiempo en línea y se autentican en tiempo real con el servidor, en una aplicación móvil se puede dar el caso en el que los usuarios no estén en línea, por lo que puede resultar apropiado establecer requisitos de tiempo que requieren autenticación fuera de línea. No solo se deben realizar validaciones del lado del servidor, sino también en la aplicación móvil, sobre todo en procesos críticos que solo deben ser llevados a cabo por usuarios con un perfil alto.

Los impactos técnicos frente a este riesgo suelen ser graves, y se resumen a continuación:

- El sistema no es capaz de identificar al usuario que solicita o realiza una acción y por ende no llevar un registro de auditoría.

- Debido a lo anterior, resulta complicado identificar el origen de un ataque, su naturaleza, y tomar medidas para prevenir futuros ataques.

- Afectación del comportamiento del sistema, sobre todo en funciones específicas que dependen de la identidad y permisos del usuario.

Desde el punto de vista del negocio, las consecuencias pueden ser las siguientes:

- Daño a la reputación

- Fraude y Robo de información.

4.2.12.6 M6 Broken Cryptography[74]. (Criptografía rota): La criptografía rota es uno de los problemas más comunes en las aplicaciones móviles. Se presenta cuando se utilizan algoritmos de cifrado débiles o que son defectuosos en su funcionamiento, lo que permite a un atacante descifrar la información fácilmente.

Tabla 6. M6 M6 Broken Cryptography.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Fácil	Común	Fácil	Grave	Específico de la aplicación y/o negocio

Fuente: El autor.

Agentes de amenazas:

- Un atacante con acceso físico al dispositivo
- Un dispositivo móvil controlado por un malware.

Si un atacante tiene acceso físico al dispositivo y obtiene la información cifrada, la dificultad para devolver los datos cifrados a información legible dependerá de los algoritmos de cifrado utilizados. Si los desarrolladores de la aplicación móvil utilizaron algoritmos de encriptación débiles o defectuosos, seguramente podrá acceder a la información encriptada.

[74] OWASP. Mobile Top 10 2014-M6. 2014. [En línea][Citado el 19 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M6

Para un negocio o empresa en el que se maneja información a través de su aplicación móvil, la criptografía rota dará lugar a lo siguiente:

- Violación de privacidad
- Robo de información
- Robo de código
- Robo de propiedad intelectual
- Daño a la reputación.

Los siguientes son algunos de las situaciones que pueden originar problemas de criptografía rota:

- Confianza en procesos de encriptación integrados: los desarrolladores de aplicaciones móviles tienden a confiar en los procesos de encriptación del sistema operativo subyacente (IOS, Android, etc…).

- Gestión defectuosa de llaves: En muchos casos se suelen utilizar los algoritmos más sofisticados y seguros, pero se desprotegen las llaves o firmas utilizadas para encriptar los datos o se combinan estos algoritmos con protocolos de encriptación propios.

- Creación y uso de Protocolos de encriptación propios: el peor error en el que puede caer un desarrollador es crear su propio protocolo e implementarlo en sus propias aplicaciones.
- Uso de algoritmos de encriptación en desuso: para protocolos como MD5 y SHA1 se ha demostrado que tienen deficiencias, por lo que se debe evitar su uso.

4.2.12.7 M7 Client Side Injection[75]. (Inyección del lado del cliente).

Tabla 7. M7 Client Side Injection.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Fácil	Común	Fácil	Moderado	Específico de la aplicación y/o negocio

Fuente: El autor.

Agentes de ataques:

- Cualquier persona que pueda enviar datos no confiables a la aplicación.
- Usuarios externos.
- Usuarios internos.
- La misma aplicación.
- Otras aplicaciones instaladas en el mismo dispositivo.

No resulta complicado para un atacante cargar un texto que pueda ser interpretado por los frameworks y el sistema operativo subyacente a la aplicación móvil. En el mejor de los casos el código malicioso se ejecutará con los permisos del usuario que sin saberlo ejecuta la aplicación, pero también puede que se ejecute con mayores privilegios, lo que puede traer consecuencias más graves.

[75] OWASP. Mobile Top 10 2014-M7. 2014. https://www.owasp.org/index.php/Mobile_Top_10_2014-M7

Ataques de inyección, tales como inyección de SQL en los dispositivos cliente, pueden ser graves si en una misma aplicación o dispositivo compartido se maneja más de una cuenta de usuario. Otros puntos de inyección están destinados a aplicaciones de desbordamiento de los componentes y procesos, pero tienen menos probabilidades de conseguir un resultado de alto impacto debido a las protecciones de código proporcionadas por la arquitectura subyacente.

Por lo general, el código malicioso roba información confidencial (contraseñas, cookies de sesión, información de identificación personal; etc.). Lo que puede traer como consecuencias para la organización:

- Fraude
- Violaciones de privacidad.

Dado que existen diferentes fuentes de datos para una aplicación móvil, es importante hacer una lista de ellos clasificados por lo que tratan de lograr. En general, los ataques de inyección en los dispositivos móviles tienen como objetivo:

*Los datos en el dispositivo**:***

- Inyección de SQL: SQLite (el almacenamiento de datos por defecto en muchos dispositivos móviles) puede estar sujeto a la inyección al igual que en las aplicaciones web. Si un atacante logra hacer una inyección de este tipo, puede significar un alto riesgo para una aplicación que es utilizada por varios usuarios dentro de un mismo dispositivo, especialmente si se maneja información confidencial, como datos de pagos.

- Inclusión de archivos locales: se trata de algo similar al caso anterior, pero en vez de tratar de buscar en una base de datos, se haría en archivos que pertenecen a un usuario específico.

*La sesión de usuario**:***

JavaScript inyección (XSS, Etc.): El navegador móvil está sujeto a la inyección de JavaScript también. Por lo general, el navegador móvil tiene acceso a la cookie de aplicaciones móviles, lo que puede conducir al robo de sesión.

*Código binario de sí mismo**:***

Un malware u otra aplicación maliciosa puede realizar un ataque binario contra la capa de presentación (HTML, JavaScript, CSS hojas de estilo en cascada) o el binario real del ejecutable de la aplicación móvil. Estas inyecciones de código se ejecutan ya sea por el marco de la aplicación móvil o el binario en sí en tiempo de ejecución.

4.2.12.8 M8 Security Decisions Via Untrusted Inputs[76]**.** (Decisiones de seguridad basadas en entradas no confiables):

Tabla 8. M8 Security Decisions Via Untrusted Inputs.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Fácil	Común	Fácil	Grave	Específico de la aplicación y/o negocio

Fuente: El autor.

[76] OWASP. Mobile Top 10 2014-M8. 2014. [En línea][Citado el 17 de mayo, 2016] Disponible en internet https://www.owasp.org/index.php/Mobile_Top_10_2014-M8

Agentes de amenazas:

- Malware
- Usuarios
- Aplicaciones vulnerables

No resulta difícil explotar una pobre o nula validación de entradas dentro de una aplicación. Un atacante puede interceptar llamados a procesos o funciones específicos en los que se evalúa algún tipo de bandera (como por ejemplo una variable oculta) para determinar si se puede o no llevar a cabo dicha función o proceso. Una vez el atacante identifica la “bandera” y sus posibles valores, puede empezar a hacer peticiones a la funcionalidad desde sus malwares u otras aplicaciones dentro del dispositivo; la funcionalidad hará su trabajo, pues no se tiene en cuenta el origen de la petición.

Es totalmente normal y válido que una aplicación móvil acepte peticiones de todo tipo de fuentes, incluyendo la Comunicación entre Procesos (IPC). Pero en los casos en los que se necesite una interacción IPC, se debe manejar una lista blanca de aplicaciones de confianza, además de en lo posible requerirse la interacción del usuario, y la validación rigurosa de las entradas.

Los impactos técnicos, producto de la explotación una vulnerabilidad relacionada con este riesgo, se pueden resumir en una *escalada de privilegios*, lo que permite a un atacante llevar acciones que solo debería hacer un usuario al que se la hayan concedido los permisos legítimamente; esto a su vez puede derivar en *pérdida de confidencialidad e integridad* de los datos manejados por la aplicación.

Para una organización, la explotación de este riesgo puede implicar *pérdida de la reputación*, también producto de la pérdida de la confidencialidad e integridad de la información.

4.2.12.9 M9 Improper Session Handling[77]. (Manejo inapropiado de sesiones): el manejo incorrecto de sesiones es muy similar a la autorización y autenticación débil establecidos en riesgo M5.

Tabla 9. M9 Improper Session Handling.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Fácil	Común	Fácil	Grave	Específico de la aplicación y/o negocio

Fuente: El autor.

En términos generales, un atacante puede interceptar el tráfico de red de una aplicación móvil y tratar de capturar las cookies que fueron obtenidas en un proceso de autenticación correcto. En muchas ocasiones se utiliza un token para realizar la autenticación frente a un servicio backend, en el caso en el que la autenticación resulte exitosa, el servidor emite una cookie de sesión, que es añadida en cada petición hecha por la aplicación al backend; el problema radica en que en ocasiones se expone ésta cookie o el mismo token en peticiones posteriores a la de autenticación.

Acciones como la validación de la sesión en cada acción a realizarse tanto en el cliente como en el servidor, establecer tiempo de expiración, o la creación de tokens seguros pueden ayudar a prevenir problemas de seguridad relacionados con este riesgo.

Un adversario que tiene acceso a un token de sesión es capaz de hacerse pasar por el usuario y hacer peticiones al servidor en su nombre. Por lo tanto, el impacto técnico depende del usuario que está siendo suplantado y la acción o servicio que se solicita. En el peor de los casos, el atacante se hace pasar por un usuario

[77] OWASP. Mobile Top 10 2014-M9. 2014. [En línea][Revisado el 20 de mayo, 2016] Disponible en internet https://www.owasp.org/index.php/Mobile_Top_10_2014-M9

administrador y hacer una solicitud de un proceso administrativo. En casos menos dramáticos, un usuario con un perfil más bajo pierde el control de sus cuentas.

Un ataque por este riesgo puede significar para una empresa o negocio:

- Fraude
- El robo de información
- Interrupción del negocio.

Algunas de las situaciones en las que se hace un inadecuado manejo de sesiones son las siguientes:

- No se invalidan los tokens en el lado del servidor
- Falta de protección adecuada por tiempo de espera
- Si no se rotan adecuadamente las cookies.
- Creación de tokens inseguros.

4.2.12.10 M10 Lack of Binary Protections[78]. (Falta de protección de los binarios): Hace referencia a lo vulnerable que resulta una aplicación cuando ésta se encuentra alojada en ambientes de los cuales no se tiene control.

[78] OWASP. Mobile Top 10 2014-M10. 2014. [En línea][Citado el 21 de mayo, 2016] Disponible en internet https://www.owasp.org/index.php/Mobile_Top_10_2014-M10

Tabla 10. M10 Lack of Binary Protections.

Explotabilidad	Predominio	Detectabilidad	Impactos técnicos	Impactos al negocio
Medio	Común	Difícil	Grave	Específico de la aplicación y/o negocio

Fuente: El autor.

Agente de amenazas:

- Un atacante que tiene acceso al paquete de la aplicación (por ejemplo un APK) y se dispone a realizar ingeniería inversa de la misma.

- Para esto un atacante necesita de una herramienta automatizada que le permita obtener el código de la aplicación y modificarlo incluyendo algún tipo de código malicioso que lleve a cabo alguna tarea oculta.

- A pesar de todos los esfuerzos por proteger los archivos binarios de una aplicación, siempre va a ser posible para un atacante obtener dicho código. Todas las medidas que se tomen siempre van a redundar en una mayor dificultad para el atacante, pero no en un blindaje totalmente efectivo.

Resulta difícil saber si una aplicación ha sido objeto de un proceso de ingeniería inversa. Normalmente esto se sabe cuándo se observa que una versión de la aplicación se encuentra en las tiendas de aplicaciones.

Desde el punto del negocio, la falta de protección de los archivos binarios de una aplicación puede redundar en los siguientes problemas:

- Robo de datos confidenciales
- Accesos no autorizados y el fraude
- Pérdida de ingresos causado por la circulación de copias piratas de la aplicación.

4.2.13 Otros modelos de penetración. A parte del OWASP Mobile Security Project, no existen muchos modelos y metodologías enfocados en la seguridad para las aplicaciones móviles. A continuación, revisaremos algunos de los modelos, metodologías, estándares y certificaciones que se enfocan o se pueden utilizar para desarrollar y analizar aplicaciones móviles:

4.2.13.1 OSSTMM. (Open Source Security Testing Methology Manual), es una metodología de pruebas de seguridad gratuita y abierta del instituto ISECOM. La premisa principal del manual de esta metodología es que "Los hechos no provienen de grandes saltos de descubrimiento, sino más bien de los pequeños pasos y cuidadosos de verificación". El manual de esta metodología se encuentra en constante y continua revisión y mejoramiento, es revisado por pares de pruebas de seguridad. En general la OSSTMM trata la seguridad operacional para conocer y medir cual es el nivel del funcionamiento de la seguridad[79].

4.2.13.2 ISSAF. (Information System Security Assessment Framework) El Marco de Evaluación de Seguridad de Sistemas de Información es una metodología estructurada de análisis de seguridad en varios dominios y detalles específicos de test o pruebas para cada uno de estos. Su objetivo es proporcionar procedimientos muy detallados para el testing de sistemas de información que reflejan situaciones reales. ISSAF proponen cinco fases: I Planeación, II Evaluación, III Tratamiento, IV Acreditación y VI Mantenimiento[80].

[79] RAMÍREZ, Mauricio. La Seguridad En Aplicaciones Móviles: Estrategias En El Mundo Actual. [En línea][Revisado el 26 de mayo, 2016] Disponible en internet http://datateca.unad.edu.co/contenidos/233016/Articulos/La_Seguridad_en_Aplicaciones_Moviles_Estrategias_en_el_Mundo_Actual_Gabriel_Ramirez.pdf

[80] Ibíd.

4.2.13.3 CHECK IT Health Check. Es un esquema creado para garantizar que las redes sensibles de los gobiernos y la infraestructura crítica nacional fueran probadas y garantizadas para un alto nivel de seguridad alto y constante. La metodología tiene como objetivo identificar las vulnerabilidades de las tecnologías de la información y las redes que puedan comprometer la seguridad, confidencialidad, disponibilidad de la información contenida en los sistemas informáticos[81].

4.2.13.4 ISACA. Se fundó en 1967 y se ha convertido en una organización global para el gobierno de la información, control, seguridad y auditoría de los profesionales de la auditoría de sistemas. Sus normas de auditoría y de control de los sistemas de información son seguidas por los profesionales de todo el mundo y sus temas de investigación en el área. CISA Certified Information Systems Auditor es la certificación principal de ISACA.

Desde 1978, el examen CISA ha medido la excelencia en el área de auditoría, control y seguridad, y se ha convertido en una certificación mundialmente reconocida y adoptada en todo el mundo como símbolo de logro. Actualmente han desarrollado estrategias y actividades en el área de la seguridad móvil para que sean implementados en el gobierno de tecnología de las organizaciones y algunos principios de seguridad que deben ser implementados[82].

4.3. REFERENTES LEGALES

La Constitución Política de Colombia de 1991 en el artículo 61 capítulo 1 establece “el Estado protegerá la propiedad intelectual por el tiempo y las formalidades que establezca la Ley”[83].

[81] Ibíd.

[82] Ibíd.

[83] Secretaría General de la Alcaldía Mayor de Bogotá D.C. Circular Conjunta 1 de 2006 Procuraduría General de la Nación 2006. 2006. http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=21006

4.3.1. Software. En la reglamentación jurídica del comercio electrónico se encontró la decisión 351 de 1993, artículo 3º, capítulo 1. Define al software como:

"Expresión de un conjunto de instrucciones mediante palabras, códigos, planes o en cualquier otra forma que al ser incorporadas en un dispositivo de lectura automatizada es capaz de hacer que el ordenador ejecute determinada tarea u obtenga determinado resultado."[84]

Este comprenderá además la documentación técnica de su uso.

Con respecto a las normas relativas a la existencia, alcance y ejercicio de los derechos de propiedad intelectual aprobado por la ley 170 de 1994 artículo 10 "programa de ordenador y compilaciones de datos establece que los programas de ordenador, sean programas fuentes o programas de objetos, serán protegidos como obras literarias según el convenio de Berna de 1971."

Es indudable que el software es el elemento más peculiar y caracterizado de lo que se denomina contratación informática. En tal sentido existen como contratos básicos los siguientes:

En esta clase de contratos el titular de los derechos de explotación, es decir quien tiene los derechos patrimoniales sobre el software, autoriza a determinada persona la distribución del mismo a cambio de una contraprestación y de unos términos establecidos.

[84] Leyes y reglamentos / Comunidad Andina de Naciones. Decisión 351 de 1993 Régimen común sobre Derecho de Autor y derechos conexos. 1993. http://www.cerlalc.org/derechoenlinea/dar/leyes_reglamentos/Colombia/Dec_andina_351.htm

4.3.1.1 Contrato de licencia de uso de software. A diferencia del caso anterior en estos contratos es el titular de los derechos de software quien autoriza a usar o utilizar dicho software con una contraprestación, sin que por ello transfiera el dominio sobre el mismo.

4.3.1.2 Contrato de Escrow. Este contrato consiste en el acuerdo entre el programador del software y el usuario en cuanto a pactar la entrega del código de fuente a un tercero (una especie de fedatario público o notario) que se constituye en depositario del mismo y se obliga a entregarlo al usuario cuando se cumplan las condiciones que se hallan previsto en el contrato[85].

4.3.1.3 Contrato de licencia de distribución de software.

Artículo 269A[86]. Acceso abusivo a un sistema informático. El que, sin autorización o por fuera de lo acordado, acceda en todo o en parte a un sistema informático protegido o no con una medida de seguridad, o se mantenga dentro del mismo en contra de la voluntad de quien tenga el legítimo derecho a excluirlo, incurrirá en pena de prisión de cuarenta y ocho (48) a noventa y seis (96) meses y en multa de 100 a 1000 salarios mínimos legales mensuales vigentes.

4.3.1.4 Artículos relativos a propiedad intelectual y propiedad industrial

Artículo 270[87]. 1. Será castigado con la pena de prisión de seis meses a dos años y multa de 12 a 24 meses quien, con ánimo de lucro y en perjuicio de tercero, reproduzca, plagie, distribuya o comunique públicamente, en todo o en parte, una obra literaria, artística o científica, o su transformación, interpretación o ejecución artística fijada en cualquier tipo de soporte o comunicada a través de cualquier

[85] GARCÍA, J. Lo que debes saber sobre el contrato de escrow. 2014. http://mprende.co/legal/lo-que-debes-saber-sobre-el-contrato-de-escrow

[86] MINTIC. Ley No 1273. 2009. [En línea][Citado el 12 de junio, 2016] Disponible en internet: http://www.mintic.gov.co/portal/604/articles-3705_documento.pdf.

[87] LEYES Y REGLAMENTOS / ESPAÑA. Capítulo XI De los delitos relativos a la propiedad intelectual e industrial, al mercado y a los consumidores. 1995. [En línea][Citado el 12 de junio, 2016] Disponible en internet: http://www.cerlalc.org/derechoenlinea/dar/leyes_reglamentos/Espana/Ley_organica.htm

medio, sin la autorización de los titulares de los correspondientes derechos de propiedad intelectual o de sus cesionarios.

2. Será castigado con la pena de prisión de seis meses a dos años y multa de 12 a 24 meses quien intencionadamente exporte o almacene ejemplares de las obras, producciones o ejecuciones a que se refiere el apartado anterior sin la referida autorización.

Igualmente incurrirán en la misma pena los que importen intencionadamente estos productos sin dicha autorización, tanto si éstos tienen un origen lícito como ilícito en su país de procedencia; no obstante, la importación de los referidos productos de un Estado perteneciente a la Unión Europea no será punible cuando aquellos se hayan adquirido directamente del titular de los derechos en dicho Estado, o con su consentimiento.

3. Será castigado también con la misma pena quien fabrique, importe, ponga en circulación o tenga cualquier medio específicamente destinado a facilitar la supresión no autorizada o la neutralización de cualquier dispositivo técnico que se haya utilizado para proteger programas de computador o cualquiera de las otras obras, interpretaciones o ejecuciones en los términos previstos en el apartado 1 de este artículo.

Artículo 271[88]. Se impondrá la pena de prisión de uno a cuatro años, multa de 12 a 24 meses e inhabilitación especial para el ejercicio de la profesión relacionada con el delito cometido, por un período de dos a cinco años, cuando concurra alguna de las siguientes circunstancias:

a. Que el beneficio obtenido posea especial trascendencia económica.

b. Que los hechos revistan especial gravedad, atendiendo el valor de los objetos producidos ilícitamente o a la especial importancia de los perjuicios ocasionados.

[88] Ibíd.

c. Que el culpable perteneciere a una organización o asociación, incluso de carácter transitorio, que tuviese como finalidad la realización de actividades infractoras de derechos de propiedad intelectual.

d. Que se utilice a menores de 18 años para cometer estos delitos.

Artículo 272[89]. 1. La extensión de la responsabilidad civil derivada de los delitos tipificados en los dos artículos anteriores se regirá por las disposiciones de la Ley de Propiedad Intelectual relativas al cese de la actividad ilícita y a la indemnización de daños y perjuicios.

2. En el supuesto de sentencia condenatoria, el Juez o Tribunal podrá decretar la publicación de ésta, a costa del infractor, en un periódico oficial.

Decreto número 1360 de 1989[90]. El cual se reglamenta la inscripción del soporte lógico (software) en el Registro Nacional del Derecho de Autor.

Artículo 1º. De conformidad con lo previsto en la Ley 23 de 1982 sobre Derechos de Autor, el soporte lógico (software) se considera como una creación propia del dominio literario[91].

Artículo 2º. El soporte lógico (software) comprende uno o varios de los siguientes elementos: el programa de computador, la descripción de programa y el material auxiliar[92].

[89] Ibíd.
[90] SECRETARÍA GENERAL DE LA ALCALDÍA MAYOR DE BOGOTÁ D.C. Decreto 1360 de 1989 Nivel Nacional. 1989. [En línea][Citado el 12 de julio, 2016] Disponible en internet: http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=10575
[91] Ibíd.
[92] Ibíd.

Artículo 3º. Para los efectos del artículo anterior se entiende por[93]:

a) "Programa de computador": la expresión de un conjunto organizado de instrucciones, en lenguaje natural o codificado, independientemente del medio en que se encuentre almacenado, cuyo fin es el de hacer que una máquina capaz de procesar información, indique, realice u obtenga una función, una tarea o un resultado específico[94].

b) "Descripción de un Programa": una presentación completa de procedimientos en forma idónea, lo suficientemente detallada para determinar un conjunto de instrucciones que constituya el programa de computador correspondiente[95].

c) "Material auxiliar": todo material, distinto de un programa de computador o de una descripción de programa, creado para facilitar su comprensión o aplicación, como por ejemplo descripción de problemas e instrucciones para el usuario[96].

Artículo 4º. El soporte lógico (software), será considerado como obra inédita, salvo manifestación en contrario hecha por el titular de los derechos de autor[97].

Artículo 5º. Para la inscripción del soporte lógico (software) en el Registro Nacional del Derecho de Autor, deberá diligenciarse una solicitud por escrito que contenga la siguiente información[98]:

1. Nombre, identificación y domicilio del solicitante, debiendo manifestar si habla a nombre propio o como representante de otro en cuyo caso deberá acompañar la prueba de su representación[99].

[93] Ibíd.
[94] Ibíd.
[95] Ibíd.
[96] Ibíd.
[97] Ibíd.
[98] Ibíd.
[99] Ibíd.

2. Nombre e identificación del autor o autores[100].

3. Nombre del productor[101].

4. Título de la obra, año de creación, país de origen, breve descripción de sus funciones, y en general, cualquier otra característica que permita diferenciarla de otra obra de su misma naturaleza[102].

5. Declaración acerca de si se trata de obra original o si por el contrario, es obra derivada[103].

6. Declaración acerca de si la obra es individual, en colaboración, colectiva, anónima, seudónima o póstuma[104].

Artículo 6º. A la solicitud de que trata el artículo anterior, deberá acompañarse por lo menos uno de los siguientes elementos: el programa de computador, la descripción del programa y/o el material auxiliar[105].

Artículo 7º. La protección que otorga el derecho de autor al soporte lógico (Software) no excluye otras formas de protección por el derecho común[106].

Artículo 8º. Este Decreto rige a partir de la fecha de su publicación[107].

[100] Ibíd.
[101] Ibíd.
[102] Ibíd.
[103] Ibíd.
[104] Ibíd.
[105] Ibíd.
[106] Ibíd.
[107] Ibíd.

Ley Estatutaria 1581 de 2012[108]. De esta norma se destacan el artículo 4, en el que se establecen los principios para el tratamiento de datos personales, especialmente el inciso g, principios de seguridad:

Artículo 4°. *Principios para el Tratamiento de datos personales[109].* En el desarrollo, interpretación y aplicación de la presente ley, se aplicarán, de manera armónica e integral, los siguientes principios[110]:

a) Principio de legalidad en materia de Tratamiento de datos: El Tratamiento a que se refiere la presente ley es una actividad reglada que debe sujetarse a lo establecido en ella y en las demás disposiciones que la desarrollen.[111]

b) Principio de finalidad: El Tratamiento debe obedecer a una finalidad legítima de acuerdo con la Constitución y la Ley, la cual debe ser informada al Titular.[112]

c) Principio de libertad: El Tratamiento sólo puede ejercerse con el consentimiento, previo, expreso e informado del Titular. Los datos personales no podrán ser obtenidos o divulgados sin previa autorización, o en ausencia de mandato legal o judicial que releve el consentimiento.[113]

d) Principio de veracidad o calidad: La información sujeta a Tratamiento debe ser veraz, completa, exacta, actualizada, comprobable y comprensible. Se prohíbe el Tratamiento de datos parciales, incompletos, fraccionados o que induzcan a error.[114]

---

[108] ALCALDÍA MAYOR DE BOGOTÁ. LEY ESTATUTARIA 1581 DE 2012. 2012. [En línea][Citado el 20 de octubre, 2017]. Disponible en internet: http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=49981

[109] Ibíd.

[110] Ibíd.

[111] Ibíd.

[112] Ibíd.

[113] Ibíd.

[114] Ibíd.

e) Principio de transparencia: En el Tratamiento debe garantizarse el derecho del Titular a obtener del Responsable del Tratamiento o del Encargado del Tratamiento, en cualquier momento y sin restricciones, información acerca de la existencia de datos que le conciernan.[115]

f) Principio de acceso y circulación restringida: El Tratamiento se sujeta a los límites que se derivan de la naturaleza de los datos personales, de las disposiciones de la presente ley y la Constitución. En este sentido, el Tratamiento sólo podrá hacerse por personas autorizadas por el Titular y/o por las personas previstas en la presente ley.[116]

Los datos personales, salvo la información pública, no podrán estar disponibles en Internet u otros medios de divulgación o comunicación masiva, salvo que el acceso sea técnicamente controlable para brindar un conocimiento restringido sólo a los Titulares o terceros autorizados conforme a la presente ley;[117]

g) Principio de seguridad: La información sujeta a Tratamiento por el Responsable del Tratamiento o Encargado del Tratamiento a que se refiere la presente ley, se deberá manejar con las medidas técnicas, humanas y administrativas que sean necesarias para otorgar seguridad a los registros evitando su adulteración, pérdida, consulta, uso o acceso no autorizado o fraudulento.[118]

h) Principio de confidencialidad: Todas las personas que intervengan en el Tratamiento de datos personales que no tengan la naturaleza de públicos están obligadas a garantizar la reserva de la información, inclusive después de finalizada su relación con alguna de las labores que comprende el Tratamiento, pudiendo sólo realizar suministro o comunicación de datos personales cuando ello corresponda al desarrollo de las actividades autorizadas en la presente ley y en los términos de la misma.

[115] Ibíd.
[116] Ibíd.
[117] Ibíd.
[118] Ibíd.

Y el artículo 18, en el cual se establecen los deberes de los responsables del tratamiento de la información de los consumidores. Especialmente el inciso b, en el que se exige mantener la información bajo las medidas de seguridad necesarias:

Artículo 18. *Deberes de los Encargados del Tratamiento*[119]***.*** Los Encargados del Tratamiento deberán cumplir los siguientes deberes, sin perjuicio de las demás disposiciones previstas en la presente ley y en otras que rijan su actividad:

a) Garantizar al Titular, en todo tiempo, el pleno y efectivo ejercicio del derecho de hábeas data.[120]

b) Conservar la información bajo las condiciones de seguridad necesarias para impedir su adulteración, pérdida, consulta, uso o acceso no autorizado o fraudulento;[121]

c) Realizar oportunamente la actualización, rectificación o supresión de los datos en los términos de la presente ley.[122]

d) Actualizar la información reportada por los Responsables del Tratamiento dentro de los cinco (5) días hábiles contados a partir de su recibo.[123]

e) Tramitar las consultas y los reclamos formulados por los Titulares en los términos señalados en la presente ley.[124]

[119] Ibíd.
[120] Ibíd.
[121] Ibíd.
[122] Ibíd.
[123] Ibíd.
[124] Ibíd.

f) Adoptar un manual interno de políticas y procedimientos para garantizar el adecuado cumplimiento de la presente ley y, en especial, para la atención de consultas y reclamos por parte de los Titulares.[125]

g) Registrar en la base de datos la leyenda "reclamo en trámite" en la forma en que se regula en la presente ley.[126]

h) Insertar en la base de datos la leyenda "información en discusión judicial" una vez notificado por parte de la autoridad competente sobre procesos judiciales relacionados con la calidad del dato personal.[127]

i) Abstenerse de circular información que esté siendo controvertida por el Titular y cuyo bloqueo haya sido ordenado por la Superintendencia de Industria y Comercio.[128]

j) Permitir el acceso a la información únicamente a las personas que pueden tener acceso a ella.[129]

k) Informar a la Superintendencia de Industria y Comercio cuando se presenten violaciones a los códigos de seguridad y existan riesgos en la administración de la información de los Titulares.[130]

l) Cumplir las instrucciones y requerimientos que imparta la Superintendencia de Industria y Comercio.[131]

[125] Ibíd.
[126] Ibíd.
[127] Ibíd.
[128] Ibíd.
[129] Ibíd.
[130] Ibíd.
[131] Ibíd.

4.3.2 Google. En las condiciones del servicio de Google Inc., con última modificación el 02 de Julio de 2016, establece que al utilizar los servicios se aceptan las políticas y condiciones establecidas e informan que sus servicios usan cierto contenido que no pertenece a la empresa, el cual es responsabilidad del proveedor que lo pone a disposición. Google puede revisar si el contenido publicado es ilegal o infringe las políticas o la ley, eliminándolo o rechazándolo[132].

A su vez Google establece un Acuerdo de distribución para desarrolladores[133] de Google Play en lo cual se puede resaltar los siguientes aspectos:

El Desarrollador acepta utilizar Google Play Store solo para los fines permitidos por el Acuerdo y cualquier ley, normativa, norma o práctica de aceptación general aplicable existente en las jurisdicciones correspondientes[134].

El uso de Google Play Store por parte del desarrollador es para distribuir Productos, proteger los derechos legales y la privacidad de los usuarios en cuanto a nombres de usuario, contraseñas u otra información personal o de inicio de sesión, por lo cual debe informar a los usuarios que su información estará disponible para su Producto, proporcionándole un aviso de privacidad legalmente válido, así como la protección correspondiente[135].

El Producto del Desarrollador solo podrá utilizar la información de usuarios para los fines específicos que previamente se le ha concedido permiso. Cuando el producto almacena información personal o confidencial proporcionada por los usuarios, debe hacerlo de un modo suficientemente seguro y únicamente durante el tiempo necesario. No obstante, si el usuario ha suscrito un acuerdo independiente con el Desarrollador que permite que el Desarrollador o su Producto almacenen o utilicen información personal o confidencial directamente relacionada con su Producto, el uso de dicha información se regirá por las condiciones de ese acuerdo independiente[136].

---

[132] Ibíd.

[133] GOOGLE. Acuerdo de Distribución para Desarrolladores de Google.2016. [En línea][Citado el 12 de junio, 2016] Disponible en internet: Playhttps://play.google.com/intl/all_es/about/developer-distribution-agreement.html

[134] Ibíd.

[135] Ibíd.

[136] Ibíd.

5. DISEÑO METODOLÓGICO PRELIMINAR

5.1. TIPO DE INVESTIGACIÓN

La presente investigación es Descriptiva, porque uno de los objetivos del estudio es describir las vulnerabilidades que pueden estar presentes en cualquier tipo de aplicación móvil, teniendo en cuenta los riesgos establecidos por el OWASP; y de tipo experimental porque se realizarán pruebas de penetración a la aplicación móvil Prism Android para tratar de identificar algunas de las vulnerabilidades descritas previamente.

5.2. DISEÑO

Para obtener la información necesaria se harán consultas en la web sobre las principales amenazas, vulnerabilidades, y los trabajos realizados por organizaciones como la OWASP. Documentos como el OWASP Mobile Top 10 Risks y la guía de desarrollo y pruebas OWASP, marcarán el camino de este trabajo.

Se aplicarán técnicas de investigación como:

Observación directa, para obtener información de primera mano sobre la exposición de la aplicación móvil a los riesgos establecidos en el Top 10 de riesgos móviles del proyecto OWASP.

Documentos (medios impresos, grabaciones de audio y video, documentación en medios electrónicos) sobre la arquitectura del sistema operativo Android, su esquema de seguridad, y el proyecto OWASP Security Mobile.

Investigación de las herramientas utilizadas para analizar aplicaciones móviles Android. Se revisan las técnicas de ingeniería inversa, descompiladores, desensambladores, funcionamiento del SDK.

Resultados obtenidos de las pruebas de hacking ético (análisis dinámico, análisis estático) realizadas a la aplicación instalada en un dispositivo móvil y/o emulador de tipo Smartphone con sistema operativo Android.

5.2.1. Población y muestra. La población es delimitada a la aplicación móvil Prism para el sistema operativo Android desde la versión 4.0 ICE CREAM, y el sistema de backend asociado.

5.2.2. Análisis de resultados. Una vez obtenida la información de la aplicación, identificados los riesgos, y hacer la selección de las herramientas apropiadas, se procede con las siguientes actividades:

- Configuración del entorno e instalación de herramientas de trabajo para la realización de las pruebas a la aplicación Prism para Android.

- Realización de pruebas de hacking ético a la aplicación Prism para Android, instalada en emulador y una Tablet con sistema operativo Android versión 5.0 Lollipop, siguiendo el Plan de pruebas diseñado y usando las herramientas escogidas para el desarrollo de las pruebas de hacking ético.

- Construcción del informe del proceso de evaluación de seguridad de la aplicación Prism.

- Construcción del documento final del proyecto de investigación.

5.3. RECURSOS DISPONIBLES

5.3.1. Recursos humanos. Para el desarrollo del presente proyecto se dispone del trabajo de su autor, un Ingeniero de Sistemas y estudiante de la Especialización Seguridad Informática en la Universidad Nacional Abierta y a Distancia – UNAD. De acuerdo al artículo 18 del reglamento general estudiantil de la UNAD, también se dispone un director de Trabajo de Grado, quien podrá estar o no vinculado a la Institución, y su función será asesorar, orientar y hacer seguimiento al desarrollo del trabajo de grado, también, participar en la sustentación y, además, será el responsable institucional de la calidad del proceso y del resultado del trabajo.

5.3.2. Recursos económicos. Se estima que para la implementación del trabajo investigativo se debe disponer de los siguientes recursos económicos:

Tabla 11. Recursos económicos.

Actividades/herramientas	**Costo**
Software	
Microsoft Office	$250.000
Equipos	
Equipo de computo	$2.200.000
Impresora Laser	$450.000
Tablet Android 5.0 Lollipop	$900.000
Servicios	
Internet x 6 meses	$480.000
Energía Eléctrica x 6 meses	$300.000

Tabla 12. (Continuación)

Actividades/herramientas	Costo
Materiales	
Insumos para equipo de cómputo e impresión	$250.000
Imprevistos	
Imprevistos	$100.000
Total	**$4.930.000**

Fuente: El autor.

5.4. CRONOGRAMA DE ACTIVIDADES.

Tabla 13. Cronograma de actividades.

CRONOGRAMA DE ACTIVIDADES																									
ACTIVIDAD	**SEMANA**																								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Estudio de Sistema Operativo Android y su arquitectura																								
Estudio del listado de riesgos establecidos en el OWASP Mobile Security Project																								
Identificación de herramientas de software a utilizar																								
Configuración de ambientes de pruebas																								
Análisis estático y dinámico de la aplicación.																								
Análisis de los resultados.																								
Construcción del informe con recomendaciones de seguridad.																								
Presentación de documento final de investigación																								

Fuente: El autor

6. DESARROLLO

6.1 PLAN DE PRUEBAS

Para el desarrollo del análisis de seguridad informática sobre la aplicación PRISM y el sistema de Backend relacionado, se debe diseñar, organizar e implementar una guía que contribuya a mantener el enfoque y objetivo del proceso, por esta razón se ha creado un plan de pruebas basado en la Metodología del Proyecto de Seguridad Móvil OWASP Mobile Security Project, enfocado en aspectos relacionados con los riesgos establecidos en dicho proyecto, y que permita la identificación de vulnerabilidades dentro de la aplicación y/o el sistema de Backend con el que ésta interactúa[137].

Según el OWASP móvil, un plan de pruebas de penetración de una aplicación móvil, independiente del tipo de aplicación y la plataforma(s) a la que esté destinada, debe tener por lo menos las siguientes tres fases[138]:

- Recopilación de información de la aplicación.
- Análisis estático
- Análisis dinámico

En la fase de recopilación de información se reconocen las funcionalidades de la aplicación, al igual que su alcance. Algunos de los interrogantes a dar respuesta en esta fase son los siguientes[139]:

[137] OWASP. OWASP Mobile Security Project - Security Testing Guide. 2013. [En línea][Revisado el 16 de septiembre, 2016] Disponible en internet https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Security_Testing_Guide

[138] Ibíd.

[139] Ibíd.

- ¿La aplicación realiza transacciones electrónicas?

- ¿Dentro de la aplicación se compran bienes o servicios?

- ¿La aplicación interactúa con alguno de los siguientes componentes de hardware?:

    - NFC
    - GPS
    - MICRÓFONO
    - CÁMARA
    - USB
    - BLUETOOTH
    - SENSORES

- La aplicación interactúa con otras aplicaciones, servicios, o datos como:

    - Telefonía (SMS, teléfono).
    - Agenda de contactos
    - Recepción de datos de aplicaciones y otros servicios en el dispositivo.
    - Almacenamiento en la nube.
    - Google Wallet
    - iCloud
    - Redes sociales
    - Dropbox

- Evernote
- Email

- ¿La aplicación requiere una o más cuentas de usuario para las pruebas de auditoría?, ¿qué tipos de usuarios o roles existen?

- ¿La aplicación utiliza algunas de las siguientes interfaces de red?

  - 3G/4G
  - WiFi
  - Bluetooth
  - NFC (Near Field Communication)
  - VPN (Virtual Private Network).

- ¿Todas las funcionalidades de la aplicación se ejecutan utilizando conexiones 3G/4G o es el WiFi requerido para acciones como la sincronización de datos?

- ¿Qué protocolos de red se utilizan? ¿Se utilizan protocolos de red seguros donde se necesitan?, en caso de ser utilizados, ¿se pueden cambiar a puertos inseguros?

- ¿La aplicación hace uso de librerías o componentes de terceros?

- ¿La aplicación valida si el dispositivo en el que se está ejecutando está rooteado?

- ¿Es posible determinar algún tipo de información del servidor?

    - Proveedor de hosting
    - Entorno de desarrollo

- ¿La aplicación hace uso de algún mecanismo o API de autenticación (Google Apps, Facebook, iTunes, OAuth, etc)?

El análisis estático se realiza sobre el código fuente de la aplicación, el cual preferiblemente debe ser proporcionado por su equipo de desarrollo, o en su defecto, puede ser obtenido mediante un proceso de decompilación del archivo APK. Para este caso en específico, la empresa BarcodeApps ha proporcionado el código fuente de la aplicación, pero de todas formas se realiza el proceso de decompilación y se trata de obtener el código fuente, para así determinar el nivel de ofuscación y el grado de protección que la aplicación establece frente a procesos de Ingeniería inversa.

En términos generales, durante el análisis estático se llevan a cabo las siguientes actividades[140]:

- Revisión de los permisos solicitados por la aplicación y los recursos a los cuales puede acceder. Esta información en su mayoría es obtenida del archivo AndroidManifest.xml.

- Tratar de identificar problemas en los archivos de configuración de la aplicación. Como ejemplo de esto se puede mencionar el valor de la propiedad “debug”, permisos de lectura y escritura, Content Providers, Intent Filters, etc.

[140] Ibíd.

- Identificar las librerías que la aplicación utiliza, tanto las nativas de Android como las desarrolladas por terceros, luego hacer una revisión en la web para determinar si tienen vulnerabilidades conocidas, si están desactualizadas, o llevan a cabo tareas que requieren altos privilegios, por ejemplo, localización y agenda de contactos.

- Identificar si toda la ejecución de la aplicación se realiza dentro de su entorno de ejecución (SandBox). Básicamente se debe verificar si se cargan librerías en tiempo de ejecución o están fuera del entorno de ejecución de la aplicación.

- Determinar los tipos de componentes utilizados para crear los objetos visuales dentro de la aplicación. En este paso es importante verificar si existen secciones dentro de la App en la que se utilicen componentes del tipo WebView, o si solo se utilizan componente de interfaz gráfica para mostrar todos los contenidos.

- Determinar si todos los permisos declarados en el AndroidManifest.xml son realmente necesarios. Debe haber por lo menos una tarea que justifique la solicitud de cada uno de los permisos.

- Identificar credenciales, API keys, o tokens quemados (hard coded) dentro del código de la aplicación.

- Identificar los puntos de entrada de datos a la aplicación y la validación de los mismos:

    - Llamados a Web Services.
    - Entradas de datos de otras aplicaciones o servicios en el dispositivo.
    - Lectura de información del sistema de archivos.
    - Mensajes de texto entrantes.

- Localizar el código encargado del proceso de autenticación y su interfaz gráfica. Basado en lo encontrado determinar:

    - Posibles métodos para romperlo: fuerza bruta, manipulación de parámetros, inyecciones SQL, etc.
    - Se utilizan otro tipo de información aparte del par usuario/contraseña: IMEI, UDID, localización, certificados, tokens, etc.
    - Componentes utilizados para mostrar el formulario de autenticación.
    - Limitación en el número de intentos fallidos.

- ¿Recibe la aplicación notificaciones del tipo push?, ¿verifica el origen de las notificaciones?

- Determinar si existen posibles valores o preferencias que pueden ser establecidos en base a entradas no validadas correctamente y que puede resultar en problemas relacionados con el escalamiento de privilegios:

    - Base de datos
    - Archivos planos
    - Respuestas HTTP.

- Determinar si existen secciones de la aplicación a los cuales se podría acceder directamente, sin la debida autorización o flujo de proceso correcto.

- Determinar si existen mecanismos adoptados por la aplicación para detectar manipulación del código de la aplicación y tomar las medidas necesarias:

    - ¿Se envía algún tipo de alerta al equipo de desarrollo?

- ¿La aplicación falla o no se cierra?
- ¿La aplicación elimina los datos?

- Determinar los mecanismos para proteger la sesión del usuario.

  - ¿La sesión expira del lado del cliente y en el servidor?
  - ¿Se utiliza algún tipo de información sensible aun después de expirada la sesión?
  - ¿Se limpia información de la memoria después de haber cerrado sesión?

- Identificar, si es que existen, los algoritmos de encriptación utilizados y sus vulnerabilidades conocidas.

- Identificar si la aplicación utiliza áreas de almacenamiento fuera de su SandBox. Por ejemplo, SD Card, carpetas temporales, almacenamiento en la nube, etc.

- Verificar si la aplicación escribe información sensible en el sistema de archivos o cualquier otro lugar:

  - Nombre de usuario y contraseñas
  - Api Key
  - Tokens de autenticación
  - Datos de usuarios o clientes.
  - Archivos de firmas

- Determinar si la aplicación utiliza certificados apropiadamente.

- En caso de utilizar certificados, verificar si se validan las siguientes situaciones:

    - Expiración
    - ¿El certificado fue emitido por una entidad de certificación válida?

- Determinar si la aplicación registra información sensible en el log durante la operación normal de la misma.

- Determinar si la aplicación registra información sensible en el log al momento de presentarse algún tipo de excepción.

Por otra parte, en el análisis dinámico se pone en marcha la aplicación en un dispositivo o en el emulador, ejecutando las diferentes funcionalidades para comprobar lo encontrado y/o identificar otros aspectos no identificados durante el análisis estático. En términos generales el análisis dinámico implica evaluar la comunicación local entre interprocesos de la aplicación, el análisis forense del sistema de archivos local, y evaluar las dependencias de servicios remotos[141].

6.1.1 Selección de herramientas. Para la realización de las pruebas de penetración a la aplicación PRISM y el sistema de Backend asociado, se seleccionaron las herramientas relacionadas a continuación, teniendo en cuenta las recomendaciones del OWASP[142] y otras muy buenas herramientas conocidas durante el periodo académico de la especialización:

[141] Ibíd.

[142] OWASP. Mobile Security Project - M-Tools. 2016. [En línea][Revisado el 19 de septiembre, 2016] Disponible en internet https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=M-Tools

Tabla 14. Herramientas seleccionadas.

Herramienta	Descripción	Licencia
ADB(Android Debug Bridge)	Es una herramienta de línea de comandos que permite la comunicación con la instancia de un emulador o dispositivo conectado[143].	Free
Androbugs	Es un Framework de análisis de vulnerabilidades en Android, que permite encontrar potenciales vulnerabilidades en las aplicaciones Android.[144]	Free
AndroGuard	Una herramienta desarrollada en Python que permite realizar un análisis estático de una aplicación, realizar búsqueda de malware, y evaluar el nivel de ofuscación del código de una App[145].	Free
Android Studio	Es el IDE de desarrollo Oficial para Android[146]	Free
Apktool	Permite realizar ingeniería inversa a partir de archivos APK[147].	Free
Burp Suite	Es una plataforma integrada para realizar pruebas de penetración de aplicaciones web y móviles. Entre las funcionalidades más importantes está el actuar como proxy para la interceptación de datos transmitidos por la red[148].	Free

[143] Android Developers. Android Debug Bridge. [En línea][Citado el 11 de septiembre, 2016]. Disponible en internet:https://developer.android.com/studio/command-line/adb.html
[144] Androbugs Framework. 2016. [En línea][Revisado el 12 de septiembre, 2016]. Disponible en internet:https://github.com/AndroBugs/AndroBugs_Framework
[145] DESNOS, Antony. AndroGuard. [En línea][Citado el 20 de septiembre, 2016] Disponible en internet: https://github.com/androguard/androguard
[146] ANDROID DEVELOPERS. Android Studio. [En línea][Citado el 12 de septiembre, 2016]. Disponible en internet: https://developer.android.com/studio/index.html?hl=es-419
[147] APKTOOL. [En línea][Citado el 16 de mayo, 2016]. Disponible en internet: https://ibotpeaches.github.io/Apktool
[148] PORTSWIGGER. Burp Suite. [En línea][Revisado el 22 de septiembre, 2016] Disponible en internet: https://portswigger.net/burp/

Tabla 15. (Continuación)

Herramienta	Descripción	Licencia
Dex2Jar	Es una herramienta que convierte archivos del tipo .DEX a código .Jar (Java)[149]	Free
Drozer	Es un Framework para el análisis de seguridad en Android desarrollado por MWR Labs. Drozer permite la manipulación de una Aplicación Android y su interacción con otras aplicaciones a través de IPC's (Inter-Process Communication).[150]	Free
Eclipse	Es un IDE muy completo y adaptable, que permite configurar el ambiente de desarrollo y vincular plugins como módulos independientes que brindan un enfoque robusto para desarrollos JEE[151].	Free
FindBugs	Es una herramienta de análisis estático que permite hacer una búsqueda de posibles errores dentro de una aplicación java. La herramienta es un JAR que puede ser usada independientemente o como plugin del entorno de desarrollo ECLIPSE[152].	Free
JD-GUI	Es una utilidad gráfica que muestra el código Java de archivos ".class". Con JD-GUI es posible examinar el código y acceder a métodos y atributos de la clase[153].	Free

[149] Sourceforge. Dex2jar. 2016. [En línea][Revisado el 23 de septiembre, 2016] Disponible en internet: https://sourceforge.net/projects/dex2jar/

[150] INFOSEC INSTITUTE. INTRODUCTION TO DROZER. 2014. [En línea][Revisado el 23 de septiembre, 2016] Disponible en internet: http://resources.infosecinstitute.com/android-hacking-security-part-13-introduction-drozer/

[151] COLORADO, Pedro y TORRES, Inírida. Análisis de seguridad de aplicaciones móviles nativas para el sistema operativo Android versión Jelly Bean 4.1.2 en dispositivos móviles Smartphone. 2015. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: repository.unad.edu.co/bitstream/10596/3412/1/40186727.pdf

[152] UNIVERSITY OF MARYLAND. FindBugs in Java Programs. 2015. [En línea][Revisado el 28 de septiembre, 2016] Disponible en internet: http://findbugs.sourceforge.net/

[153] TAYLOR, A. JD-GUI. 2015. [En línea][Revisado el 16 de septiembre, 2016] Disponible en internet: https://github.com/java-decompiler/jd-gui

Tabla 16. (Continuación)

Herramienta	Descripción	Licencia
JDWP[154]	(Java Debug Wire Protocol), es una utilidad de ORACLE que permite realizar depuración de aplicaciones JAVA, incluyendo aplicaciones Android.	Free
LLDB	Esta herramienta permite hacer un debug remoto de aplicaciones móviles implementando una arquitectura cliente - servidor. El cliente y el servidor se comunican usando el protocolo GDB- remote, usualmente sobre el protocolo TCP/IP[155].	Free
MobSF	Mobile Security Framework es una herramienta que permite realizar análisis estáticos y dinámicos de aplicaciones Android y iOS[156].	Free
Nmap	Es una utilidad que permite descubrir y auditar redes. Nmap permite levantar información de redes, servicios o servidores, para lo cual envía paquetes y analiza las respuestas a dichos paquetes[157].	Free
QARK	Es una herramienta para análisis estáticos de aplicaciones Android.	Free
Sqliteman	Es una aplicación que gestiona bases de datos SQLite desde una interfaz muy sencilla.	Free

[154] ORACLE. JDWP. 2016. [En línea][Citado el 17 de septiembre, 2016] Disponible en internet: https://docs.oracle.com/javase/8/docs/technotes/guides/troubleshoot/introclientissues005.html
[155] THE LLDB DEBUGGER. [En línea][Revisado el 16 de septiembre, 2016] Disponible en internet: http://lldb.llvm.org/remote.html
[156] GITHUB. Mobile Security Framework. 2016. [En línea][Citado el 17 de noviembre, 2016] Disponible en internet: https://github.com/ajinabraham/Mobile-Security-Framework-MobSF
[157] NMAP ORG. Nmap (Network mapper). [En línea][Revisado el 17 de septiembre, 2016] Disponible en internet: https://nmap.org/

Tabla 17. (Continuación)

Herramienta	Descripción	Licencia
SQLite Manager	Extensión de Mozilla Firefox para la gestión de bases de datos SQLite[158].	Free
W3af	Es un Framework desarrollado en python que permite hacer ataques y auditoría a aplicaciones web. W3af permite hacer un escaneo dinámico de una aplicación web, enfocado en un grupo de objetivos específicos[159].	Free

Fuente: El autor.

6.1.2. Lista de chequeo para análisis de seguridad de la aplicación y sistema de Backend. A continuación se establece la lista de chequeo, la cual se ha realizado teniendo en cuenta las recomendaciones del OWASP[160] [161] [162]. Se detallan las actividades a realizar para identificar posibles problemas en cada uno de los riesgos, el tipo de análisis, y las herramientas a utilizar, las cuales fueron descritas en la subsección anterior.

[158] COLORADO, Pedro y TORRES, Inírida. Análisis de seguridad de aplicaciones móviles nativas para el sistema operativo Android versión Jelly Bean 4.1.2 en dispositivos móviles Smartphone. 2015. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: repository.unad.edu.co/bitstream/10596/3412/1/40186727.pdf

[159] Wa3f. 2013. [En línea][Revisado el 2 de octubre, 2016] Disponible en internet: http://w3af.org/

[160] OWASP. Android Testing Cheat Sheet. 2016. [En línea][Revisado el 9 de octubre, 2016] Disponible en internet: https://www.owasp.org/index.php/Android_Testing_Cheat_Sheethttps://www.owasp.org/index.php/Android_Testing_Cheat_Sheet

[161] OWASP. Mobile Checklist Final 2016. 2016. [En línea][Citado el 11 de octubre, 2016] Disponible en internet: https://drive.google.com/file/d/0BxOPagp1jPHWYmg3Y3BfLVhMcmc/viewhttps://drive.google.com/file/d/0BxOPagp1jPHWYmg3Y3BfLVhMcmc/view

[162] CARTER, Jonathan y SINGH, Milán. OWASP Mobile Application Security Guide. . [En línea][Citado el 9 de octubre, 2016] Disponible en internet: https://drive.google.com/file/d/0BxOPagp1jPHWczhwYjRQNzZIekU/view

Tabla 18. Lista de chequeo del lado del cliente.

Lista de chequeo del lado del cliente.				
M10. Falta de protección de los binarios				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-01	La aplicación es vulnerable a ataques de Ingeniería Inversa	Decompilar la aplicación y establecer el nivel de ofuscación de código de la misma. Analizar los archivos AndroidManifest.xml y build.gradle. Realizar ataques binarios a la aplicación e intentar realizar manipulación y modificación del código.	Dex2Jar, APKTool, Android Studio	Estático
MSTG-02	La aplicación es depurable	Verificar el valor de la propiedad o atributo android:debuggable en el archivo AndroidManifest.xml. Revisión del código de la aplicación en busca de establecimiento de la propiedad debuggable y analizar el APK de la aplicación para detectar el tipo de “release”.	Android Studio, Drozer	Estático

Tabla 19. (Continuación)

M10. Falta de protección de los binarios				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-03	Falta de controles de Checksum y detección de modificación de código	Verificar si en el código de la aplicación se detecta la modificación de código y se toman acciones al respecto.	Android Studio, Eclipse, ADB	Estático Dinámico
MSTG-04	La aplicación se puede ejecutar en un dispositivo rooteado	Buscar fragmentos de código en los que se verifique si el dispositivo está rooteado. Se verificará la existencia de test-keys, certificados OTA, y APK's que solo se ejecutan en dispositivos rooteados.	Android Studio.	Estático.

Tabla 20. (Continuación)

M2. Almacenamiento de datos Inseguro				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-05	Almacenamiento de datos sensibles en archivos de Logs	Verificar los archivos de logs de la aplicación, logs de eventos, logs de fallos, etc. Luego tratar de encontrar algún tipo de información sensible como credenciales, direcciones IP, información de sesión, etc.	ADB, Android Studio.	Estático - Dinámico
MSTG-06	Credenciales quemadas en el código (Hard coded)	Revisar el código de la aplicación, para tratar de identificar algún tipo de información sensible quemada en el código.	Android Studio.	Estático
MSTG-07	Credenciales son almacenadas en la base de datos sin ser encriptadas	Verificar si la aplicación interactúa con base de datos locales. Revisar tablas en las que se almacenan credenciales de usuarios sin ser encriptados.	JD-GUI, Sqliteman	Dinámico

Tabla 21. (Continuación)

M2. Almacenamiento de datos Inseguro				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-08	Acceso a Información sensible después de realizar un volcado de memoria	Realizar un volcado de memoria del dispositivo y verificar si aún se puede acceder a algún tipo de información sensible que debería ser protegido por la aplicación.	ADB	Dinámico
MSTG-09	Almacenamiento de datos sensibles fuera del Sandbox de la aplicación.	Revisar el código de la aplicación para tratar de identificar fragmentos en los que se interactúe con la tarjeta SD. Verificar el sistema de archivos, específicamente la tarjeta SD.	ADB, Android Monitor.	Estático, Dinámico
MSTG-10	Almacenamiento de datos sensibles en caché	Verificar si la aplicación realiza almacenamiento de datos sensibles en la caché.	Android Monitor.	Dinámico

Tabla 22. (Continuación)

M3. Protección insuficiente en la capa de transporte				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-11	Almacenamiento de datos sensibles en preferencias	Verificar si la aplicación realiza almacenamiento de datos sensibles en las preferencias. Verificar valores de propiedades MODE_WORLD_READABLE de las preferencias.	ADB, Android Studio.	Estático, Dinámico
MSTG-12	Información sensible es enviada como texto legible a través de la red	Analizar el tráfico de red de la aplicación y determinar si se transmite información sensible por el protocolo HTTP en vez de HTTPS.	Burp Suite	Dinámico
MSTG-13	Datos sensibles son enviados como un parámetro del tipo querystring	Verificar si existen mecanismos de validación de los parámetros que se transmiten en peticiones a través de la red.	Android Studio, Burp Suite	Estático, Dinámico
MSTG-14	Protocolos de red inseguros	Identificar los protocolos de red utilizados por la aplicación	Burp Suite	Estático - Dinámico

Tabla 23. (Continuación)

M4. Fuga de datos involuntaria				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-15	Exposición de datos en el log	Tratar de identificar fragmentos de código en los que se almacenen datos sensibles en el log. Verificar el log del dispositivo para corroborar el almacenamiento de datos sensibles.	Android Studio, Android Monitor.	Estático - Dinámico
MSTG-16	Portapapeles (Clipboard) no es deshabilitado	Verificar si es posible el copiado y pegado de datos sensibles en campos de textos.	ADB	Dinámico

Tabla 24. (Continuación)

M4. Fuga de datos involuntaria				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-17	"Application Backgrounding"[163] [164]	Identificación de Screenshots tomados mientras la aplicación entra en reposo. Verificar posibles capturas de imágenes con datos sensibles.	ADB	Dinámico
MSTG-18	"Url Caching" (peticiones y respuestas)	Revisar el log del dispositivo para tratar de identificar cookies almacenadas en éste. Identificar componentes del tipo WebView y verificar si se limpia su caché.	ADB	Dinámico

[163] CHELL, D. ERASMUS, T, & COLLEY, S. The Mobile Application Hacker's Handbook. [En línea][Citado el 16 de octubre, 2016] Disponible en internet: https://books.google.com.co/books?id=lgNhBgAAQBAJ&pg=PA139&lpg=PA139&dq=application+background+screenshot&source=bl&ots=lLmx0bbgpB&sig=LDlPnrNx_VAbi-mESTBAufFMXRk&hl=es-419&sa=X&ved=0ahUKEwi7ifG26-fPAhVHYiYKHVx1DAE4FBDoAQhBMAU#v=onepage&q=application%20background%20screenshot&f=false

[164] VAN, Rutger. iOS App Security - Backgrounding screenshot. 2013. [En línea][Revisado el 28 de noviembre, 2016] Disponible en internet: https://technology.amis.nl/2013/05/03/ios-app-security-backgrounding-screenshot/

Tabla 25. (Continuación)

M5. Autenticación y autorización pobres				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-19	Permisos de aplicación inseguros.	Verificar los permisos solicitados por la aplicación y si todos. ¿En caso de permisos considerados “Dangerous”, se realiza algún tipo de verificación y/o requieren de la interacción del usuario?	Android Studio, AndroGuard, ADB.	Estático, Dinámico
MSTG-20	Bloqueo de cuenta no implementado	Realizar 10 intentos de autenticación de vendedores y clientes con credenciales incorrectas.	ADB	Dinámico
MSTG-21	La autenticación se puede evadir.	Llevar a cabo ataques binarios contra la aplicación para evadir autenticación offline. Identificar controles de acceso faltantes en funcionalidades específicas. Intentar llevar a cabo funcionalidades del Backend, removiendo todos los tokens de sesión en las peticiones realizadas por la aplicación.	Burp Suite, ADB, JDWP.	Dinámico

Tabla 26. (Continuación)

M5. Autenticación y autorización pobres				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-22	Escalamiento de privilegios	Llevar a cabo ataques binarios contra la aplicación e intentar ejecutar funcionalidades que sólo deberían ser ejecutados por usuarios con altos privilegios	Burp Suite, ADB, JDWP.	Dinámico
MSTG-23	No implementación o implementación no apropiada de una sección para cambiar contraseñas.	En caso de estar implementado, verificar si un usuario puede cambiar la contraseña de otro. Verificar si la funcionalidad requiere de una confirmación por parte del usuario. Verificar si la funcionalidad de cambiar contraseña es susceptible de falsificación.	Android Studio, ADB	Estático, Dinámico

Tabla 27. (Continuación)

M5. Autenticación y autorización pobres				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-24	Políticas de contraseñas débiles	Verificar que tipos de caracteres son permitidos en las contraseñas. Verificar la distinción entre mayúsculas y minúsculas. En caso de tener implementado la funcionalidad de cambiar contraseña, ¿el usuario puede volver a ingresar una contraseña antigua?	ADB	Dinámico
MSTG-25	Opción de autocompletar no está deshabilitada	Verificar los campos en formularios de logins para determinar si tienen la opción de autocompletar activado.	Android Studio, JD-GUI, ADB.	Estático, Dinámico
MSTG-26	Uso de IMEI/UDID falsos para la autenticación	Verificar si se utiliza IMEI/UDID para procesos de autenticación y autorización. Intentar realizar procesos de autenticación y autorización con IMEI/UDID falsos	Burp Suite	Dinámico

Tabla 28. (Continuación)

M6. Criptografía rota				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-27	Opción “Recordar credenciales” activa	Identificar si existen credenciales o sesiones de usuario guardadas en el dispositivo	ADB.	Dinámico
MSTG-28	Aplicación hace uso de Algoritmos de encriptación débiles	Identificar en el código algoritmos de encriptación desaconsejados (RC4, MD4, MD5, SHA1). Identificar Keys, tokens, o credenciales quemadas en el código de la aplicación o que puedan ser interceptados mediante un ataque binario. Identificar si existen protocolos o métodos de encriptación no estándares o creados por los desarrolladores de la aplicación.	Android Studio, ADB.	Estático

Tabla 29. (Continuación)

M6. Criptografía rota				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-29	Envío de información en claro a través de túnel SSL	Verificar si los certificados son expedidos por una entidad de certificación (CA) válida. Verificar si los certificados no han expirado. Verificar que la información de lado del servidor y la establecida en el certificado coinciden.	ADB, Burp Suite	Dinámico
MSTG-30	Almacenamiento de claves de encriptación localmente	Verificar si dentro de la misma aplicación se guardan claves o firmas de certificados.	ADB	Estático Dinámico
M7. Inyecciones en lado del cliente				
MSTG-31	La aplicación es vulnerable a XSS	Identificar malas configuraciones en los WebViews que hagan la aplicación vulnerable a ataques de JavaScript Injections (XSS)	Android Studio, JDWP, ADB	Estático - Dinámico

Tabla 30. (Continuación)

M7. Inyecciones en lado del cliente				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-32	Validaciones del lado del cliente pueden ser evadidas.	Identificar posibles riesgos de Inyecciones SQL e inclusión local de archivos a través de Content Providers. Verificar si el JavaScript y la ejecución de plugins están deshabilitado para los WebViews. Verificar si la aplicación permite la inclusión local de archivos a través de WebViews, utilizando directivas como setAllowFileAccess.	Android Studio, ADB, Drozer.	Estático Dinámico
MSTG-33	Inyecciones SQL.	Identificar entradas a la aplicación en las que se puedan insertar cadenas de texto que permitan realizar inyecciones SQL.	Android Studio, ADB.	Estático - Dinámico

Tabla 31. (Continuación)

M8. Decisiones de seguridad basadas en entradas no confiables				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-34	Fuga de datos en Content Providers	Verificar en el Archivo AndroidManifest la declaración de Content Providers Verificar el valor de atributo android:exported Realizar peticiones del tipo Content Query a los componentes.	Android Studio, Drozer	Estático Dinámico
MSTG-35	Inclusión local de archivos (LFI).	Identificar Content Providers con atributo “exported”. Intentar acceder a archivos del sistema a través de Content Providers vulnerables. Identificar WebViews dentro de la aplicación e intentar cargar archivos del sistema dentro de éstas.	Android Studio, JDWP, ADB, Drozer	Estático Dinámico

Tabla 32. (Continuación)

M8. Decisiones de seguridad basadas en entradas no confiables				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-36	Modificación de URL	Identificar los llamados a URL´s desde la aplicación y determinar si se realiza algún tipo de validación de la URL o los parámetros de la misma antes de realizar estos llamados.	Android Studio, JDWP, ADB.	Estático, Dinámico
MSTG-37	Abuso de componentes a través de IPC Intents[165]	Identificar si existen Android Exported Components en la aplicación. Verificar si se validan las aplicaciones con las que interactúa la aplicación a través de IPC. Verificar si se envía información sensible a través de IPC.	Android Studio, Drozer, ADB	Estático

[165] ANDROID DEVELOPER. Security Tips. [En línea][Citado el 18 de octubre, 2016]. Disponible en internet: https://developer.android.com/training/articles/security-tips.html

Tabla 33. (Continuación)

M8. Decisiones de seguridad basadas en entradas no confiables				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-38	Uso indebido de URL Schemes[166] [167]	Identificar el llamado a URL´s dentro de la aplicación revisando el código de la aplicación y obtener su estructura.	Drozer, ADB.	Estático
M9. Manejo Inapropiado de sesiones				
MSTG-39	Fijación de sesión	Interceptar respuestas del servidor que contengan “Set-cookie”, cambiar su valor, verificar si la aplicación continúa su ejecución y obtiene respuestas exitosas del servidor utilizando el valor alterado.	Burp Suite	Dinámico

[166] ZANDBERGUEN, F. URL Schemes for IOS and Android. 2014. [En línea][Revisado el 21 de octubre, 2016] Disponible en internet: http://fokkezb.nl/2013/08/26/url-schemes-for-ios-and-android-1/

[167] THE POWER OF WINGS. Abusing the Intent URL Scheme Redux. 2015. [En línea][Citado el 22 de octubre, 2016] Disponible en internet: http://rotlogix.com/2015/05/09/the-power-of-wings-abusing-the-intent-url-scheme/

Tabla 34. (Continuación)

M9. Manejo Inapropiado de sesiones				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-40	La aplicación no implementa una opción para cerrar sesión	Verificar si existe un botón para finalizar sesión. Verificar si se invalida la sesión después de cierto tiempo. Verificar si se realiza una adecuada validación de sesión, no solo en la aplicación, sino también del lado del servidor.	ADB, Drozer, Burp suite	Dinámico
MSTG-41	Uso de Cookies persistentes	Verificar si se resetean las cookies al realizar proceso de autenticación, cambios de estado y/o usuario, etc.	ADB, Burp Suite	Dinámico
MSTG-42	Tokens inseguros	Verificar si la aplicación utiliza tokens y si estos son basados en algoritmos estándares, suficientemente largos, complejos, y aleatorios.	Android Studio, Burp Suite	Estático - Dinámico

Fuente: El autor.

Tabla 35. Lista de chequeo del lado del servidor.

Lista de chequeo del lado del servidor.				
M1. Debilidad en los controles del lado del servidor de la aplicación				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-43	Contraseñas legibles en respuestas desde Web Services	Interceptar respuestas desde el servidor Verificar si se envían datos sensibles en las respuestas del servidor, y si estos se transmiten en texto claro (tanto en cabeceras como en el cuerpo)	Burp Suite	Dinámico
MSTG-44	Referencia directa a recursos internos sin autenticación	Identificar los servicios o recursos remotos a los cuales accede la aplicación. Realizar peticiones a estos servicios y recursos en el servidor de forma anónima	Burp Suite	Dinámico
MSTG-45	Aplicación no tiene o implementa indebidamente un sistema de manejo de sesiones	Verificar si la aplicación, aparte de lado del cliente, invalida las sesiones de usuarios en el lado del servidor.	Burp Suite.	Dinámico

Tabla 36. (Continuación)

M1. Debilidad en los controles del lado del servidor de la aplicación				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-46	Falta de protección adecuada por “Timeout”	Verificar si es posible realizar peticiones al servidor después de haber tenido un tiempo de inactividad considerable.	Burp Suite	Dinámico
MSTG-47	Validación incorrecta de entradas en el lado del servidor	Identificar los parámetros enviados a los Web Services Manipular el valor de los parámetros y verificar si estos son validados del lado del servidor.	Burp Suite.	Dinámico
MSTG-48	Se muestra información interna sensible en páginas de error	Identificar los parámetros enviados a los Web Services Manipular el valor de los parámetros y verificar si estos son validados del lado del servidor. Analizar las respuestas de error por parte del servidor, para tratar de identificar información sensible.	Burp Suite	Dinámico

Tabla 37. (Continuación)

M1. Debilidad en los controles del lado del servidor de la aplicación				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-49	Subida de archivos maliciosos en el servidor	Identificar Web Services en el servidor en el que reciba algún tipo de archivo adjunto. Intentar subir al servidor archivos que podrían considerarse maliciosos.	ADB, Android Monitor, Burp Suite, Browser	Dinámico
MSTG-50	XSS (Cross Site Scripting)	Identificar entradas en la aplicación, en las que se pueda ingresar información que posteriormente pueda ser enviada al servidor. Ingresar cadenas con código malicioso que permita llevar a cabo algún tipo de ataque por XSS. Verificar si existe algún tipo de filtro en la aplicación. Intentar enviar los datos maliciosos al servidor.	ADB, Burp Suite.	Dinámico

Tabla 38. (Continuación)

M1. Debilidad en los controles del lado del servidor de la aplicación				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-51	CORS (Cross Origin Resource Sharing)	Realizar escaneo dinámico del servidor e Intentar cargar recursos web de dominios no alojados en el servidor.	NMap, W3af, Browser	Dinámico
MSTG-52	Es posible realizar peticiones HTTP diferentes a GET y POST	Realizar escaneo dinámico del servidor y tratar de identificar verbos HTTP diferentes GET y POST.	NMap, W3af	Dinámico
MSTG-53	Falsificación de petición en sitios cruzados (Cross Site Request Forgery (CSRF)/SSRF)	Realizar escaneo dinámico del servidor y tratar de identificar vulnerabilidades del tipo CSRF.	Nmap, W3af	Dinámico
MSTG-54	Respuestas HTTPS almacenadas en caché	Verificar las respuestas del servidor e identificar las transferencias hechas por el puerto HTTPS. Verificar los valores de las cabeceras Cache-control y Pragma.	Burp Suite	Dinámico

Tabla 39. (Continuación)

M1. Debilidad en los controles del lado del servidor de la aplicación				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-55	Propiedad o atributo “Path” no establecido en Cookies	Realizar escaneo dinámico del servidor con herramientas automáticas. Analizar las respuestas del servidor. Verificar si existen archivos PHP con la función phpinfo y el resultado de su ejecución.	Nmap, W3af, Burp Suite.	Dinámico
MSTG-56	Cookie Sin HttpOnly habilitado	Realizar escaneo dinámico del servidor con herramientas automáticas. Analizar las respuestas del servidor. Verificar si existen archivos PHP con la función phpinfo y el resultado de su ejecución.	Nmap, W3af, Burp Suite.	Dinámico

Tabla 40. (Continuación)

M1. Debilidad en los controles del lado del servidor de la aplicación				
Test ID	**Nombre de Prueba**	**Descripción**	**Herramientas**	**Tipo de Análisis**
MSTG-57	Propiedad o atributo “Secure” no establecido en Cookies	Realizar escaneo dinámico del servidor con herramientas automáticas. Analizar las respuestas del servidor. Verificar si existen archivos PHP con la función phpinfo y el resultado de su ejecución.	Nmap, w3af, Burp Suite.	Dinámico
MSTG-58	Es posible hacer “Fingerprinting” del Servidor y/o el Sistema Operativo	Realizar escaneo dinámico del servidor con herramientas automáticas. Analizar las respuestas del servidor. Verificar si existen archivos PHP con la función phpinfo y el resultado de su ejecución.	Nmap, w3af, Burp Suite.	Dinámico

Fuente: El autor.

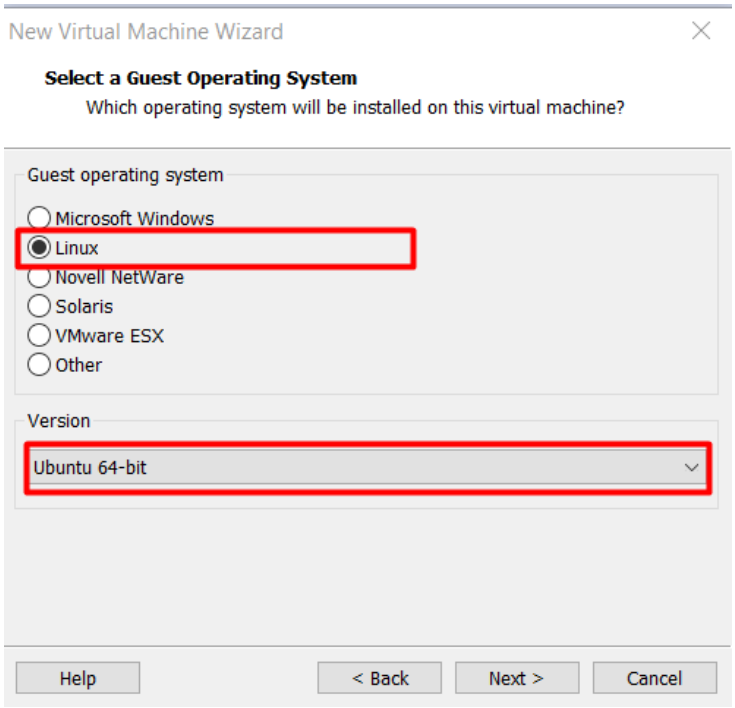
6.2 CONFIGURACIÓN DE AMBIENTE DE PRUEBAS

La mayoría de las herramientas seleccionadas para la ejecución de las pruebas son integradas en Santoku. Santoku es una distribución de Linux basada en Ubuntu, que contiene herramientas preinstaladas para la realización de labores de ingeniería inversa, pruebas de penetración para sitios web y aplicaciones móviles, principalmente aplicaciones Android, lo cual facilita en gran medida el trabajo de Pentesting de la aplicación PRISM.

Santoku fue descargado de la dirección electrónica: https://santoku-linux.com/. El sistema operativo fue instalado en una máquina virtual creada con VMWare Work Station 10.0.0. A continuación algunas evidencias del proceso:

Con la ayuda del asistente se procedió a crear una máquina de tipo Linux, Ubuntu de 64 bits:

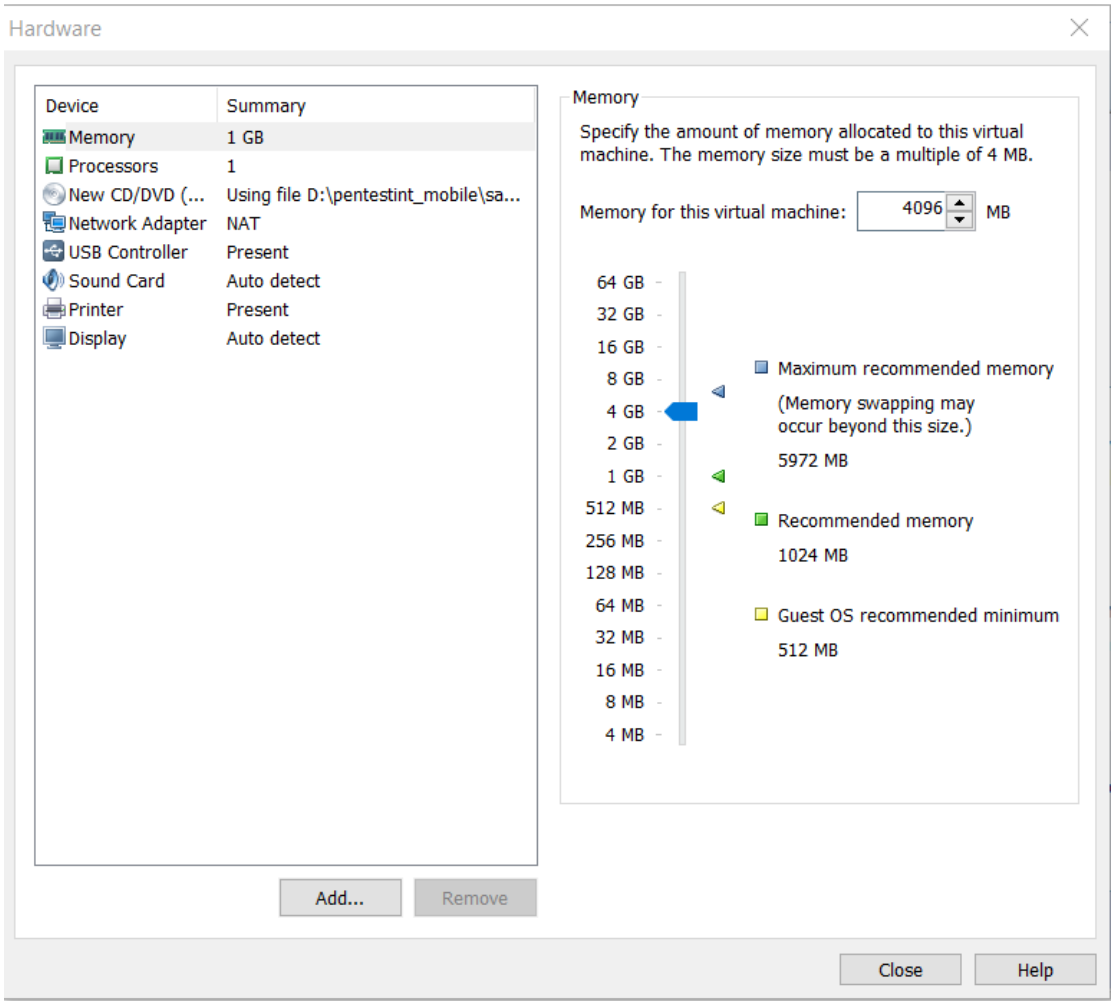
Figura 6. Configuración máquina virtual.



Fuente: el autor.

Otra parte importante fue establecer el tamaño de la memoria que según la documentación del sitio oficial se recomienda como mínimo 4GB:

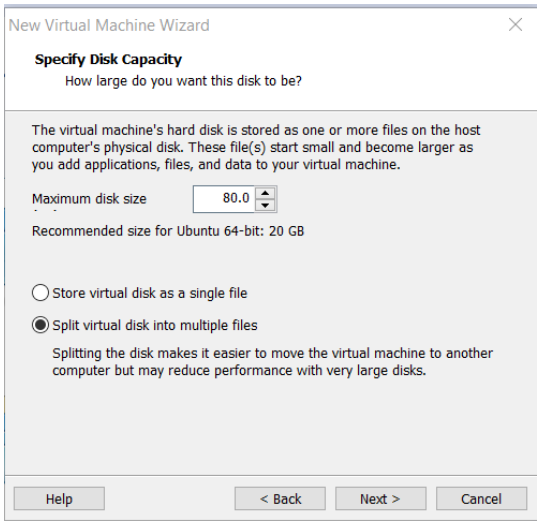
Figura 7. Tamaño de memoria.



Fuente: el autor.

El tamaño del disco mínimo recomendado es de 80GB, por lo que se procedió a indicar dicho tamaño:

Figura 8. Tamaño de memoria asignado.



Fuente: el autor.

Luego de haber configurado la máquina virtual se procedió a arrancarla con la imagen de disco descargada, el cual permite arrancar el sistema como un Live. A continuación, se muestran algunos de los pasos importantes de la instalación:

Lo primero fue arrancar el instalador ubicado en el escritorio:

Figura 9. Instalador Santoku.

Fuente: el autor.

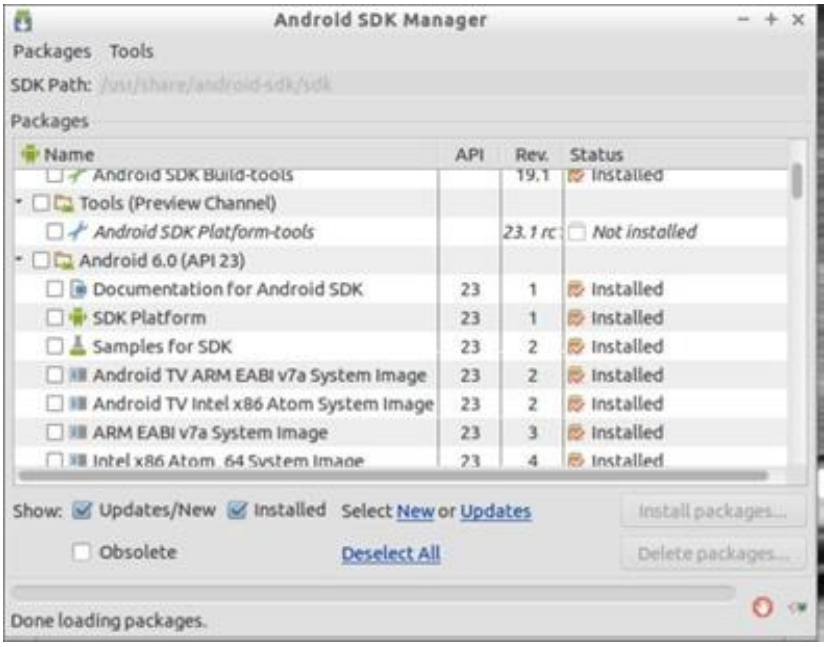
En el transcurso de la instalación se debió establecer el nombre de la máquina y datos de acceso del usuario, además de datos de ubicación y configuración del teclado, como debe hacerse en cualquier distribución de Linux:

Figura 10. Datos de usuario Santoku.

Fuente: el autor.

Luego de haber instalado el sistema se realizó una exploración de las herramientas que este ofrece, encontrando que efectivamente varias de las herramientas propuestas inicialmente vienen preinstaladas, como son drozer, burpsuite, Apktool, dex2jar, JD-GUI, Android SDK Manager, entro otros. A continuación, se muestra la ventana de actualización del SDK hasta el API 23:

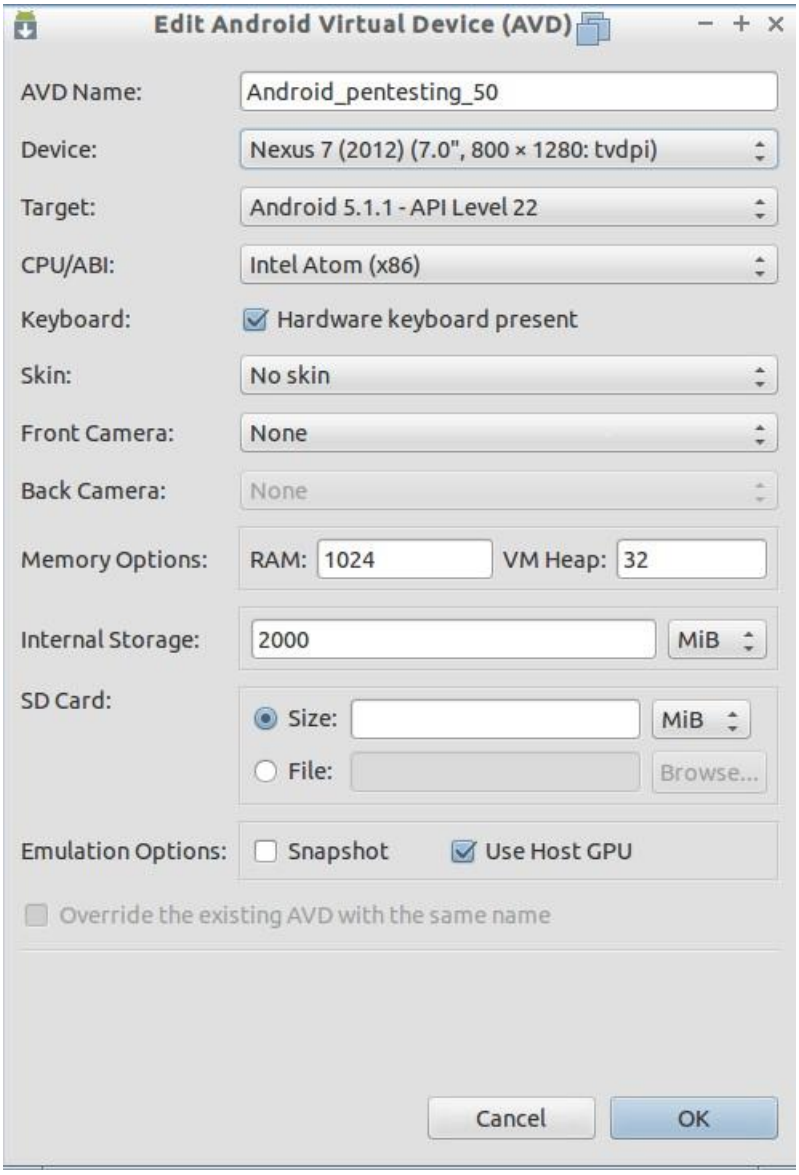
Figura 11. Android SDK Manager.



Fuente: el autor.

El siguiente paso fue la creación de un dispositivo virtual para la realización de las pruebas, a continuación, la configuración utilizada:

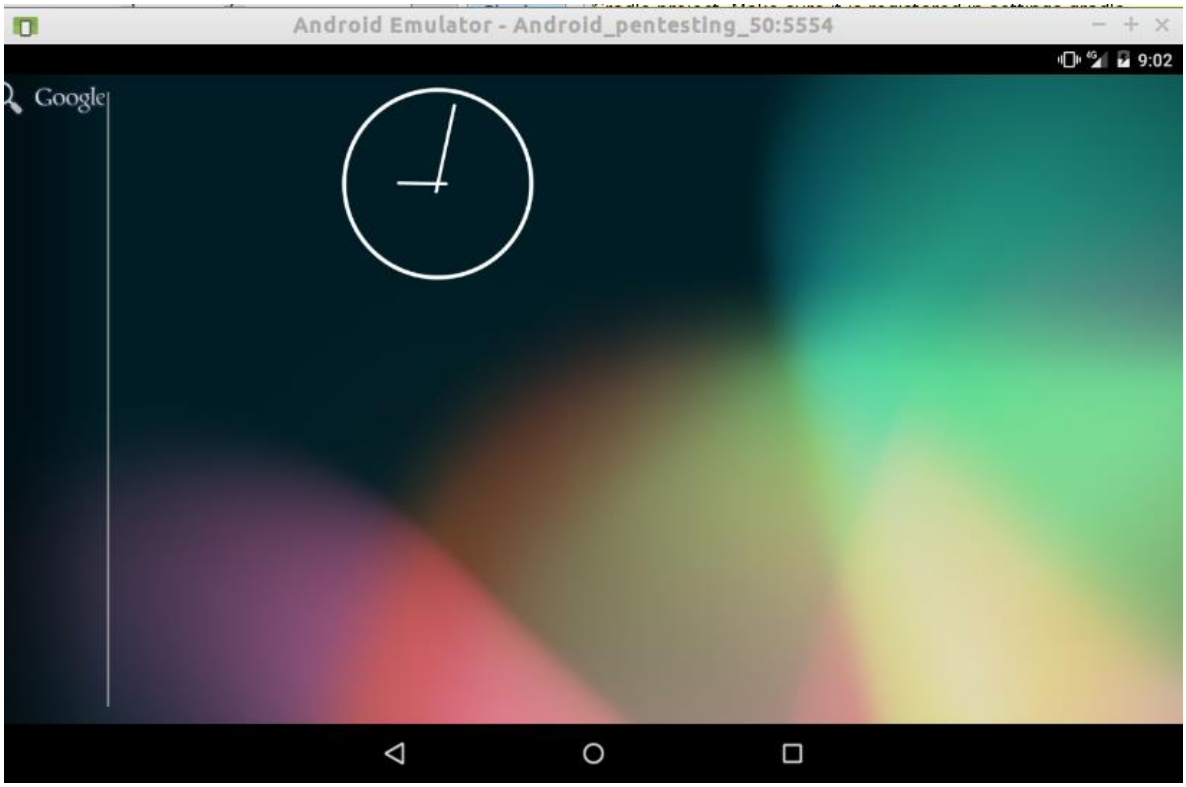
Figura 12. Creación dispositivo virtual Android.



Fuente: el autor.

El cual se puede ver funcionando en la siguiente imagen:

Figura 13. Emulador en ejecución.



Fuente: el autor.

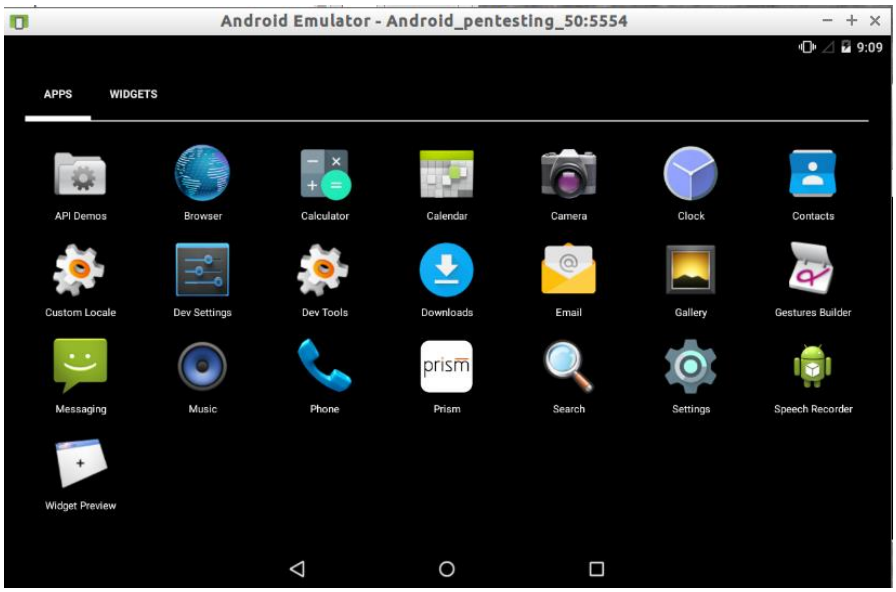
Con el emulador funcionando se procedió a instalar la aplicación, utilizando el APK proporcionado por la empresa:

Figura 14. Instalación de la aplicación en el emulador.

Fuente: El autor.

Con lo que aparece la aplicación en el listado de aplicaciones del emulador:

Figura 15. Verificación instalación de la aplicación.



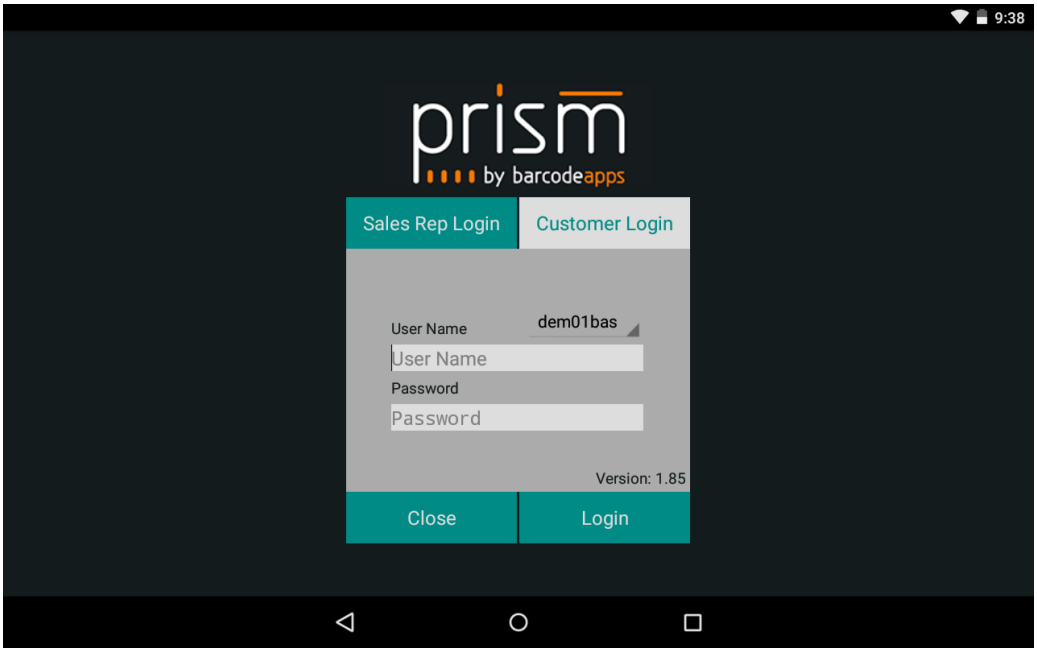
Fuente: El autor.

6.3 RECOPILACIÓN DE INFORMACIÓN SOBRE LA APLICACIÓN

A continuación, se describen las principales funcionalidades y características de la aplicación:

Sistema de autenticación:

Figura 16. Pantalla login Prism Android.



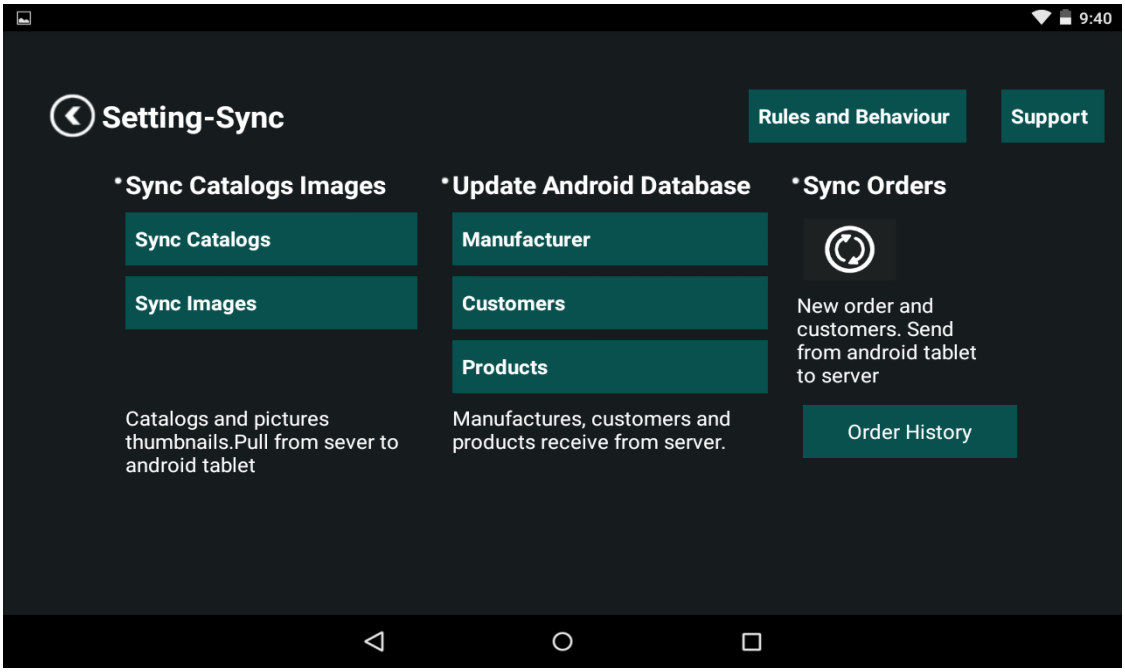
Fuente: El autor.

La aplicación trabaja principalmente con dos tipos de usuarios: Sales Rep (vendedores) y Customers (clientes), quienes pueden acceder a funcionalidades específicas y visualizar información que solo a ellos les interesa.

En el caso en el que la aplicación no tenga conexión a internet el proceso de autenticación se realiza en base al sistema de almacenamiento local, que en este caso se trata de una base de datos SQLite.

*Sincronización de Datos: m*ientras un vendedor de una compañía específica está cubriendo una ruta de venta, o se encuentra en una feria comercial, necesita tener la información de clientes, productos, y catálogos en el dispositivo. Por lo que periódicamente debe realizar una sincronización de estos datos, como se muestra en la siguiente imagen:

Figura 17. Sincronización de datos.

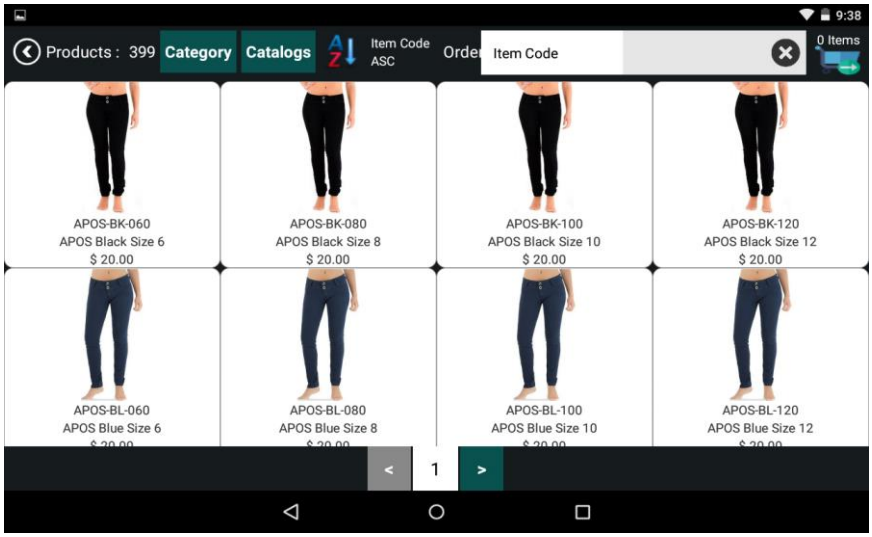


Fuente: El autor.

Para realizar todo este proceso de sincronización, la aplicación debe comunicarse con una serie de Web Services que se encuentran alojados en servidores que pueden ser propiedad de la compañía BarcodeApps y en algunos casos de las empresas a quienes BarcodeApps les proporciona la herramienta (self hosted).

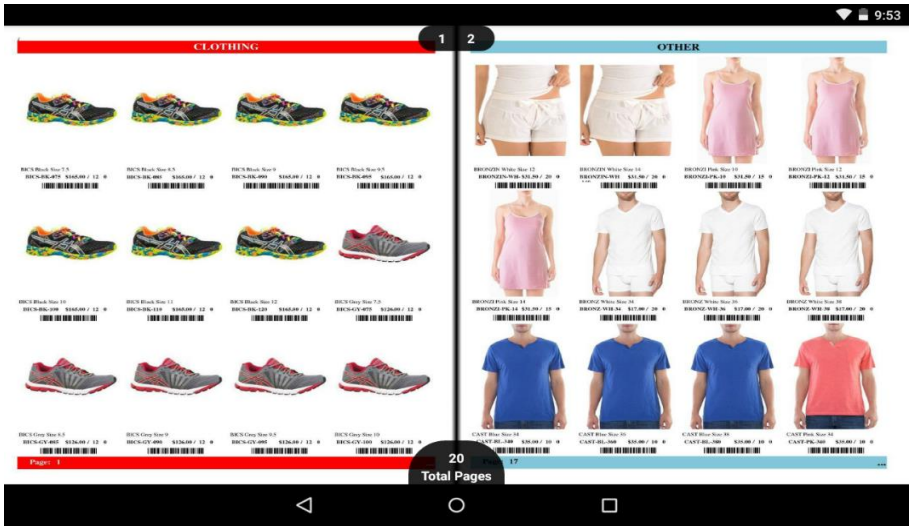
Luego de realizar todo el proceso de sincronización de datos, un vendedor puede exhibir los catálogos y productos a clientes existentes y/o potenciales, como se muestra en las siguientes figuras:

Figura 18. Listado de categorías y productos.



Fuente: El autor.

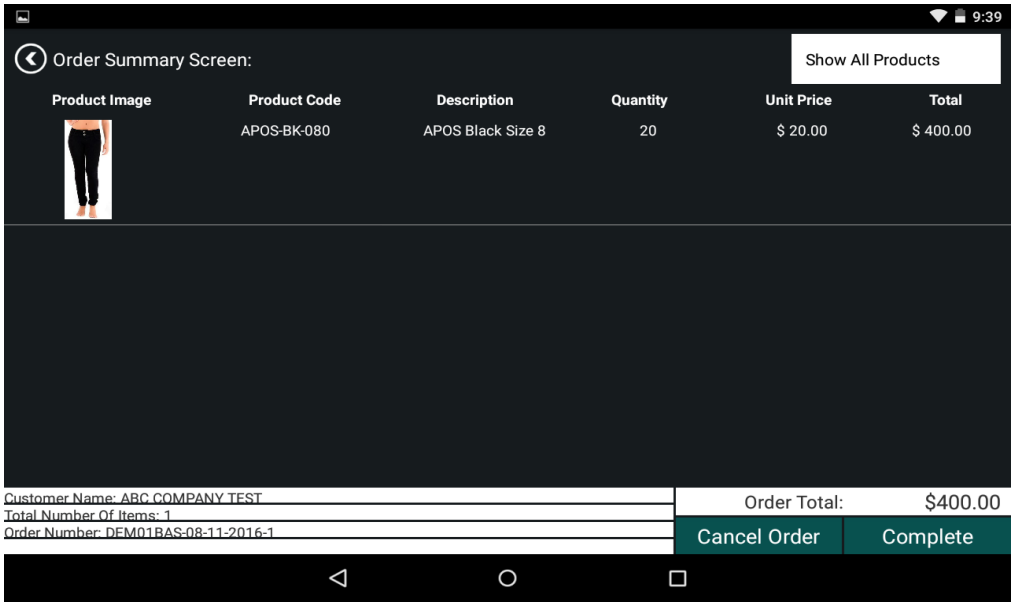
Figura 19. Presentación de productos como catálogo.



Fuente: El autor.

El vendedor puede ir agregando los productos y ver el resumen del pedido en cualquier momento si su cliente así se lo pide:

Figura 20. Resumen de un pedido.



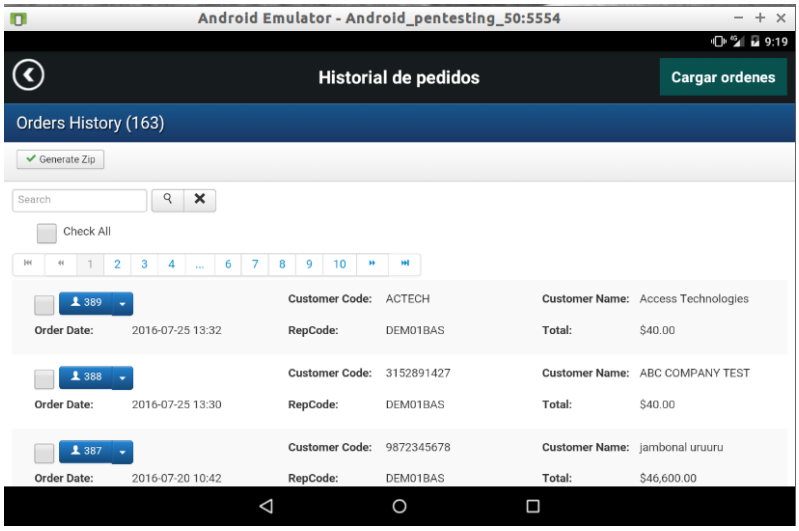
Fuente: el autor.

Al seleccionar la opción completar, la orden se almacena localmente y en el caso de contar con conexión a internet puede ser enviada inmediatamente al servidor; si no está conectado a internet, la orden puede ser sincronizada posteriormente.

Si la orden capturada es para un cliente existente, el vendedor puede seleccionarlo desde un listado al que él como vendedor tiene acceso. Si por el contrario es para un cliente nuevo, la aplicación le permite capturar los datos de la persona para luego crear el registro del cliente localmente y enviar también sus datos a un Web Service que además de registrar la orden, creará también el registro del cliente en la base de datos del servidor.

Un usuario vendedor también puede acceder a su historial de pedidos, para lo cual existe una sección en la que se le muestra el listado de pedidos en el servidor. Esta sección consta de un WebView, en el que se carga una sección de un portal web, en la que puede seleccionar una o más órdenes, generar un archivo ZIP, que luego puede ser descargado desde la aplicación.

Figura 21. Historial de pedidos cargados en WebView.



Fuente: El autor.

La aplicación tiene una funcionalidad que permite solicitar soporte técnico a la empresa BarcodeApps, enviándoles un correo electrónico con información del estado de la aplicación y sus datos.

Figura 22. Correo de Soporte Técnico.

Fuente: El autor.

Un cliente de una compañía puede instalar también la aplicación, ver el listado de productos, y crear pedidos a su nombre, pero obviamente no puede ver información de otros clientes y mucho menos realizar pedidos a nombre de otra persona.

A partir de la revisión hecha se da respuesta a los interrogantes planteados anteriormente:

1. ¿La aplicación realiza transacciones electrónicas?

No, por el momento no realiza transacciones electrónicas, pero es una de las funcionalidades a incluir.

2. ¿Dentro de la aplicación se compran bienes o servicios?

SI. La aplicación permite capturar pedidos de clientes.

3. ¿La aplicación interactúa con alguno de los siguientes componentes de hardware?:

☐NFC

☒GPS

☐MICRÓFONO

☐CÁMARA

☐USB

☒BLUETOOTH

☐SENSORES

4. La aplicación interactúa con otras aplicaciones, servicios o datos como:

☐Telefonía (SMS, teléfono).

☐Agenda de contactos

☐Recepción de datos de aplicaciones y otros servicios en el dispositivo.

☒Almacenamiento en la nube.

☐Google Wallet

☐iCloud

☐Redes sociales

☐Dropbox

☐Evernote

☒Email

5. ¿La aplicación requiere una o más cuentas de usuario para las pruebas de auditoría?, ¿qué tipos de usuarios o roles existen?

Si. La aplicación trabaja principalmente con dos tipos de usuarios: Sales Rep (vendedores) y Customers (clientes), quienes pueden acceder a funcionalidades específicas y visualizar información que solo a ellos les interesa.

6. ¿La aplicación utiliza algunas de las siguientes interfaces de red?

☒3G/4G

☒WiFi

☐Bluetooth

☐NFC (Near Field Communication)

☐VPN (Virtual Private Network).

7. ¿Todas las funcionalidades de la aplicación se ejecutan utilizando conexiones 3G/4G, o es el WiFi requerido para acciones como la sincronización de datos?

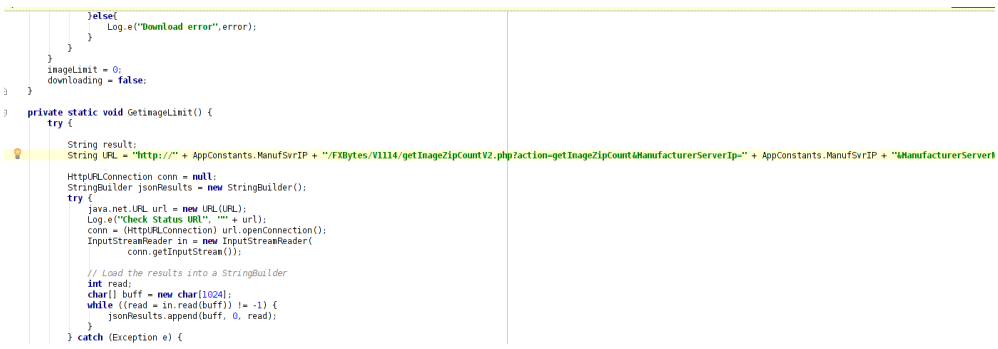
Existen funcionalidades que pueden ser realizadas sin estar conectado a internet mediante 3G/4G o WiFi.

Por su parte las funcionalidades de sincronización de productos, clientes, y/o pedidos necesitan obviamente de una conexión a internet. Sin embargo, no se valida si hace a través de WiFi o 3G/4G.

8. ¿Qué protocolos de red se utilizan? ¿Se utilizan protocolos de red seguros donde se necesitan?, en caso de ser utilizados, ¿se pueden cambiar a puertos inseguros?

La aplicación hace uso de los protocolos HTTP y HTTPS:

Figura 23. Uso de protocolo HTTP.



Fuente: El Autor.

Figura 24. Uso de protocolo HTTPS.

```
package com.bas.prism;

import ...

@SuppressLint({ "JavascriptInterface", "SetJavaScriptEnabled" })
public class PrintDialogActivity extends Activity {
    private static final String PRINT_DIALOG_URL = "https://www.google.com/cloudprint/dialog.html";
    private static final String JS_INTERFACE = "AndroidPrintDialog";
    private static final String CONTENT_TRANSFER_ENCODING = "base64";

    private static final String ZXING_URL = "http://zxing.appspot.com";
    private static final int ZXING_SCAN_REQUEST = 65743;

    /**
     * Post message that is sent by Print Dialog web page when the printing
     * dialog needs to be closed.
     */
    private static final String CLOSE_POST_MESSAGE_NAME = "cp-dialog-on-close";

    /**
     * Web view element to show the printing dialog in.
     */
    private WebView dialogWebView;

    /**
     * Intent that started the action.
     */
    Intent cloudPrintIntent;
```

Fuente: El Autor.

9. ¿La aplicación hace uso de librerías o componentes de terceros?

Figura 25. Librerías utilizadas por la aplicación.

```
        exclude 'META-INF/LICENSE'
        exclude 'META-INF/license.txt'
        exclude 'META-INF/LGPL2.1'
        exclude 'META-INF/NOTICE.txt'
        exclude 'META-INF/NOTICE'
        exclude 'META-INF/notice.txt'
    }
}

dependencies {
    compile 'com.android.support:support-v4:21.0.3'
    compile files('libs/ScanAPIAndroid.jar')
    compile files('libs/ScanAPIFactoryAndroid.jar')
    compile files('libs/apache-mime4j-core-0.7.2.jar')
    compile files('libs/droidText.0.2.jar')
    compile files('libs/httpclient-4.3.1.jar')
    compile files('libs/httpcore-4.3.jar')
    compile files('libs/httpmime-4.3.1.jar')
    compile files('libs/picasso-2.4.0.jar')
    compile files('libs/redlasersdk.jar')
}
```

Fuente: El Autor.

Si, la aplicación hace uso de las siguientes librerías:

Android Support: en la biblioteca de compatibilidad de Android se ofrecen varias funciones que no vienen integradas al Framework. Estas bibliotecas ofrecen versiones de funciones nuevas que son compatibles con versiones anteriores, proporcionan elementos de UI útiles que no se incluyen en el Framework y ofrecen diferentes utilidades a las cuales las apps pueden recurrir[168].

Apache Mime: Es una librería de Apache que provee un analizador (MimeStreamParser), para los flujos de mensajes de correo electrónico en formato rfc822 y MIME. El analizador utiliza un mecanismo de devolución de llamada para informar de los sucesos de análisis, como el inicio del Header y Body de una petición[169].

Dentro del proyecto se utiliza para enviar archivos ZIP que contiene archivos XML´s a un Web Service que procesa e inserta pedidos en el servidor remoto.

DroidText: se trata de una librería que permite generar archivos PDF desde Java y Android, de acuerdo a lo consultado, el sitio oficial está fuera línea desde 2013, lo que implica que ya no hay actualizaciones y soporte para esta librería.

Apache HttpCore: es un conjunto de componentes de transporte HTTP de bajo nivel que se pueden utilizar para crear servicios HTTP personalizados tanto del lado del cliente como de servidor con una huella mínima. HttpCore admite dos modelos de E / S: modelo de E / S de bloqueo basado en el E / S de Java clásico y modelo de E / S sin bloqueo basado en Java NIO[170].

[168] ANDROID DEVELOPERS. Biblioteca de compatibilidad. [En línea][Citado el 13 de mayo, 2016]. Disponible en internet: https://developer.android.com/topic/libraries/support-library/index.html

[169] The Apache Software Foundation. Mime4j. 2016. https://james.apache.org/mime4j/index.html

[170] The Apache Software Foundation. Apache HttpComponents. 2016.https://hc.apache.org/

Apache HttpClient: es una implementación de HTTP compatible con HTTP / 1.1 basada en HttpCore. También proporciona componentes reutilizables para la autenticación del cliente, la administración de estado HTTP y la administración de la conexión HTTP[171].

RedLaser SDK: El SDK de RedLaser es una librería que permite implementar un escáner de código de barras en una aplicación Android o iOS[172].

Durante la revisión del código del proyecto no se encontró implementación de esta librería, lo que puede implicar un riesgo de seguridad.

ScanAPIAndroid y ScanAPIFactoryAndroid: Son librerías que proporcionan un API para trabajar con lectores de códigos barras conectados a través de Bluetooth a un computador o dispositivo móvil[173].

Picasso: Es una librería que permite la carga de imágenes dentro de una aplicación Android con una menor afectación en el tiempo de carga de la misma. Algunas de las características de esta librería son las siguientes[174]:

- Administrar el reciclado de los ImageView y cancelar la descarga en un adapter.

- Transformación compleja de imágenes con un uso de memoria mínimo.

[171] Ibíd
[172] eBay Inc. Barcode Scanning SDK for iOS and Android Documentation. 2012. http://redlaser.com/developers/documentation/
[173] Socket Mobile. Scan Api Reference. 2016. http://www.socketmobile.com/docs/default-source/developer-documentation/scanapi.pdf?sfvrsn=2
[174] Square Inc. Picasso - A powerful image downloading and caching library for Android. 2013. http://square.github.io/picasso/

- Caching automático de memoria y disco.

10. ¿La aplicación valida si el dispositivo en el que se está ejecutando está rooteado?

No. Luego de hacer una revisión del código fuente no se encontraron métodos de validación de dispositivo rooteado. Se buscaron comandos como “xbin”, “su”, “sbin”, “system”.

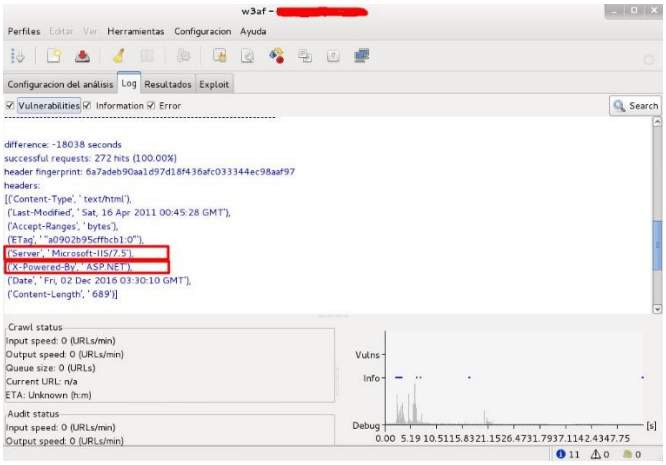
Tampoco se encontraron validaciones de presencia de aplicaciones que solo se ejecutan en dispositivos rooteados, por ejemplo la aplicación “Superuser.apk”. En la prueba MSTG-04 se revisará en mayor detalle la validación de dispositivo rooteado.

11. ¿Es posible determinar algún tipo de información del servidor?

El servidor es propiedad de la empresa, la cual gestiona su DNS a través de un tercero.

Se pudo establecer la versión del servidor web y el entorno de desarrollo

Figura 26. Información del servidor.



Fuente: El autor.

12. La aplicación hace uso de algún mecanismo o API de autenticación (Google Apps, Facebook, iTunes, OAuth, etc).

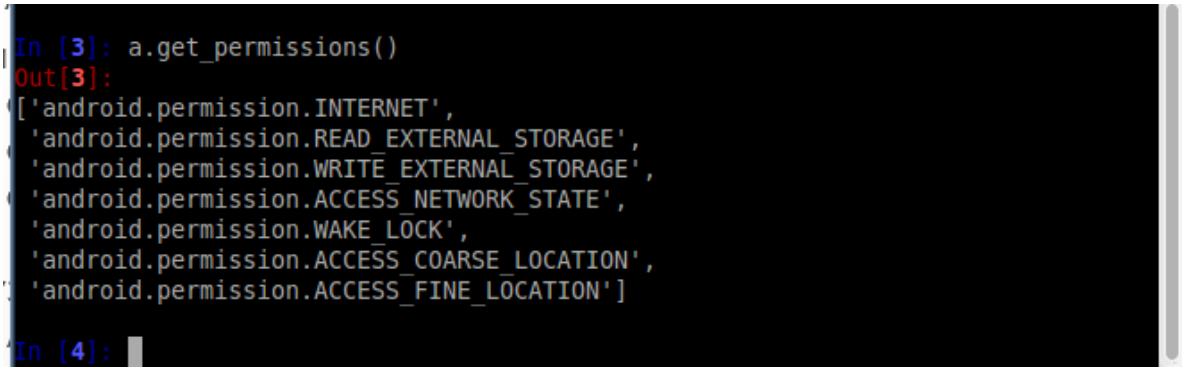
No. La aplicación no hace uso de ningún mecanismo de autenticación externa. Cuando un usuario accede por primera vez a la aplicación e ingresa sus credenciales, estas son validadas en el servidor, si el proceso de autenticación es exitoso se realiza un proceso de sincronización de productos, clientes, e imágenes; almacenándose en una base de datos local la información del usuario y la de los productos y clientes sincronizados.

La próxima vez que el usuario ingrese a la aplicación, las credenciales se validan contra el sistema de almacenamiento local.

Identificación de permisos:

Al obtener los permisos solicitados por la aplicación:

Figura 27. Identificación de permisos.



Fuente: El autor.

Se obtiene el siguiente listado:

Tabla 41. Permisos solicitados por la aplicación.

Permiso	Nivel de protección	Descripción
INTERNET	Normal	Permite a la aplicación abrir sockets de red[175]. Básicamente es el que permite que una aplicación pueda conectarse a internet.
ACCESS_NETWORK_STATE	Normal	Permite a la aplicación conocer información del estado de las redes.
WAKE_LOCK	Normal	Permite que la aplicación controle que el procesador no entre en reposo y que la pantalla del dispositivo se atenúe[176].

[175] Android Developers. Manifest Permissions. 2016. https://developer.android.com/reference/android/Manifest.permission.html
[176] Ibíd.

Tabla 42. (Continuación)

Permiso	Nivel de protección	Descripción
READ_EXTERNAL_STORAGE	Peligroso	Permite a una aplicación leer archivos del almacenamiento externo[177].
WRITE_EXTERNAL_STORAGE	Peligroso	Que permite a la aplicación escribir y leer (implícitamente el permiso READ_EXTERNAL_STORAGE es true) archivos del almacenamiento externo[178].
ACCESS_COARSE_LOCATION	Peligroso	Permite a una aplicación acceder a la ubicación del teléfono basado en la red de datos o WiFi.
ACCESS_FINE_LOCATION	Peligroso	Permite a una aplicación acceder a la ubicación precisa basada en GPS.
READ_PHONE_STATE	Peligroso	Permite el acceso de sólo lectura al estado del teléfono, incluido el número de teléfono del dispositivo, la información actual de la red celular, el estado de las llamadas en curso y una lista de todas las cuentas de teléfono registradas en el dispositivo[179].

Fuente: El autor.

[177] Ibíd.
[178] Ibíd.
[179] Ibid

6.4 EJECUCIÓN DE LAS PRUEBAS

Tomando el APK proporcionado por la empresa, se procede a realizar las pruebas, de acuerdo lo planificado y establecido en la lista de chequeo. …En el Anexo A… se muestra en detalle el procedimiento realizado en cada de unas las pruebas establecidas, acompañadas de las recomendaciones pertinentes.

6.5 ANÁLISIS DE RESULTADOS

Finalizado el proceso de ejecución del plan de pruebas sobre la aplicación PRISM para Android, los resultados obtenidos se analizan para establecer el grado de seguridad de ésta, teniendo en cuenta cada uno de los riesgos establecidos en el proyecto de seguridad móvil OWASP:

6.5.1 M1 Weak Server Side Controls. Al realizar el análisis de seguridad del servidor, se encontraron varios problemas de seguridad:

En primera instancia la exposición de información de la configuración del servidor, a través de varios archivos PHP que contienen la función phpinfo(), lo que permitiría a personas malintencionadas encaminar un posible ataque.

Inadecuada configuración de las propiedades HttpOnly, Path, y Secure, lo que expone la información de las cookies y sesiones creadas en el servidor.

Método HTTP TRACE activado, lo que podría permitirle a un atacante obtener información sensible y ser utilizado para otro tipo de ataques como XSS.

Manejo inadecuado de sesiones, causado por la falta de una verificación efectiva de la identidad de los usuarios que realizan las peticiones a los Web Services consumidos por la aplicación móvil.

Se evidenció que en los Web Services consumidos por la aplicación no se hace una adecuada validación de los parámetros enviados en la URL, tampoco del tipo de archivos enviados dentro archivos comprimidos y de la información que estos contienen. Este problema deriva en ataques muchos más severos, como inyecciones SQL, subida de archivos maliciosos al servidor, y XSS almacenado.

Transferencia de información sensible tanto en peticiones como respuestas: los Web Services que la aplicación consume pueden ser accedidos por HTTP, y en muchos de ellos se recibe y envía información sobre parámetros de conexiones a bases de datos en el servidor y del usuario autenticado en la aplicación.

Por lo anterior, se puede decir que es necesario mejorar los controles existentes en el servidor, con el objetivo de mantener la confidencialidad e integridad de la información que en éste se aloja.

6.5.2 M2. Insecure Data Storage. Luego de hacer una revisión del código de la aplicación, las bases de datos que utiliza, los archivos de registro, las memorias internas y caché del dispositivo, se halló lo siguiente:

La aplicación almacena información sensible de usuarios en las bases de datos, archivos de logs, y preferencias. Se evidenció como en los procesos de autenticación se registra en el log las consultas realizadas a la base de datos y las peticiones a los Web Services, para lo cual se utilizan nombres de usuarios, contraseñas, e incluso el identificador del dispositivo.

Las credenciales de usuario e información de conexión a base de datos son almacenadas en la base de datos SQLite del dispositivo móvil sin aplicarles ningún tipo de encriptación.

Después que los usuarios se autentican dentro de la aplicación, ésta recuerda sus credenciales, para lo cual las almacena sin ningún tipo de encriptación en el archivo de preferencias de la aplicación.

Lo anterior indica que la aplicación se expone a problemas que podrían ir desde la pérdida de datos para un usuario específico, hasta la ejecución de fraudes y con ello el daño de la reputación de la empresa.

Como hecho positivo se destaca que no se encontraron credenciales quemadas en el código de la aplicación, y tampoco se encontró información sensible en la memoria del dispositivo luego hacer volcados de memoria.

6.5.3 M3 Insufficient Transport Layer Protection. Luego de capturar y analizar el tráfico de red generado durante las conexiones establecidas entre la aplicación y los diferentes Web Services con los interactúa, se pudo constatar que la comunicación entre las partes se realiza utilizando el protocolo HTTP, lo que implica un alto grado de vulnerabilidad, ya que en las mayorías de las peticiones se identificó información de los nombres de usuarios, contraseñas, información de conexión a la base de datos en el servidor, y el identificador del dispositivo en el que se ejecuta la aplicación móvil.

Además de lo anterior, muchas de las peticiones a los Web Services se realizan utilizando peticiones del tipo GET, lo que implica el envío de información sensible como parámetros en la URL.

6.5.4 M4 Unintended Data Leakage. Durante el análisis se detectó que la a aplicación almacena datos sensibles en el log del dispositivo y el portapapeles no es deshabilitado para los campos en los que se requiere ingresar información confidencial, más específicamente en campos de nombres de usuarios y contraseñas. Lo anterior indica que existe el riesgo latente que un atacante pueda obtener información sensible, al revisar el log o instalar en el dispositivo móvil una aplicación que capture constantemente la información puesta en el portapapeles del dispositivo, para luego almacenarla o enviarla a una ubicación que pueda analizar posteriormente**.**

Por otra parte, se identificó que la aplicación solicita para su ejecución permisos con nivel de protección “dangerous” (peligrosos), algunos son necesarios, y otros no tanto. Los permisos ACCESS_COARSE_LOCATION y ACCESS_FINE_LOCATION se requieren para capturar la ubicación desde la que se realiza una orden o pedido; el permiso WRITE_EXTERNAL_STORAGE, es

necesario para, entre otras cosas, copiar la base de datos de la aplicación a la memoria externa del dispositivo y posteriormente adjuntarla a un correo electrónico que se envía al personal de soporte técnico de la empresa, lo cual es un problema de seguridad, ya que el archivo correspondiente a la base de datos no es eliminado inmediatamente después de enviar el correo de solicitud de soporte, y además contiene información sensible sin encriptar.

El mayor problema encontrado es que la aplicación captura innecesariamente el identificador del dispositivo para controlar las instalaciones de la aplicación por parte de un mismo usuario en diferentes dispositivos, y para ello requiere del permiso READ_PHONE_STATE. Una aplicación con este permiso, además del identificador físico, puede acceder al número de teléfono del dispositivo y a los números con los que éste ha establecido una llamada telefónica, lo que acompañado del hecho que la aplicación es depurable, puede significar un problema de seguridad.

6.5.5 M5 Poor Authorization And Authentication. Aprovechando el hecho de que la aplicación se puede depurar, fue posible arrancar actividades a las que en teoría solo se puede acceder después de un proceso de autenticación exitoso. Esto podría permitirle a personas malintencionadas evadir la validación de sus credenciales, e incluso llevar a cabo funcionalidades que no tiene permitido. Como ejemplo de esto, se resalta la prueba en la que se ingresó como un cliente y desde la consola se arrancó la actividad que permite listar otros clientes de la compañía, para luego seleccionar uno de ellos, y completar la orden a nombre del cliente seleccionado.

En la ventana de autenticación se encontró que la aplicación no limita el número de intentos de autenticación fallidos, tiene las opciones de autocompletar y recordar credenciales activas, implementa unas políticas de contraseñas débiles, y utiliza innecesariamente el identificador del dispositivo en procesos de autenticación y para controlar el número de instalaciones que un mismo usuario puede hacer de la aplicación.

Otro aspecto importante es que una vez un usuario se encuentra autenticado de la aplicación, no se realiza una adecuada limpieza de las variables que almacenan información de su sesión.

Por todo lo anterior se puede decir que la aplicación no implementa un sistema de autenticación y autorización efectivo, al no poder identificar con certeza al usuario que lleva a cabo una función dentro de la aplicación o una petición al servidor. Esto además de poner en riesgo la integridad de la información, dificulta llevar un registro de log o auditoría de la actividad del usuario dentro de la aplicación.

6.5.6 M6 Broken Cryptography. Como se evidencia en la prueba MSG-29, no se encontraron peticiones realizadas a través de túneles SSL. No porque no se realice comunicación con un sistema remoto, sino porque toda la comunicación entre la aplicación y el servidor se realiza por el protocolo HTTP, lo que implica un problema mayor.

Como consecuencia de lo anterior, en la revisión del código no se encontró uso de algoritmos de encriptación desaconsejados o creados por los mismos desarrolladores de la aplicación, y tampoco tokens, claves, o firmas almacenados localmente o quemados en el código de la misma.

6.5.7 M7 Client Side Injection. Durante el análisis se hizo una búsqueda de Content Providers, sin encontrar el uso de los mismos dentro de la aplicación. Esto indica que es poco probable que la aplicación pueda sufrir ataques de inyecciones SQL o fuga de datos a través de Content Providers**.**

Por su parte, para los WebViews utilizados dentro de la aplicación se encontró que existen algunas configuraciones que podrían resultar en problemas de seguridad bajo algunas circunstancias. Se demostró que es posible ejecutar código JavaScript y acceder al sistema de archivos desde los Webviews, lo que podría significar riesgos de seguridad en el caso en que personas malintencionadas puedan manipular la información que se visualiza dentro del navegador web cargado dentro de un WebView.

El hallazgo más importante fue el derivado de la no validación de los datos ingresados en los campos de texto. Mediante una inyección SQL, ingresando la cadena **x' or 'x'=x',** fue posible "burlar" el sistema de autenticación de la aplicación.

6.5.8 M8 Security Decisions Via Untrusted Inputs. Como se mencionó anteriormente, la aplicación no hace uso de Content Providers, por lo que es poco vulnerable frente a ataques en los que se aproveche del envío de información sensible a los Content Providers.

Pero, por otro lado, en uno de los componentes gráficos del tipo WebView se carga una URL que inicialmente proviene de uno de los Web Services y se almacena en la base de datos local del dispositivo, sin ningún tipo de encriptación y validación. Un atacante podría modificar la URL a cargar en el WebView si logra interceptar y alterar las respuestas del servidor, tener acceso a la base de datos local, o alterar el comportamiento de la aplicación aprovechando que ésta es depurable. El atacante podría indicar una URL que contenga algún tipo de código JavaScript malicioso, o la ruta de uno de los archivos del sistema para llevar a cabo una Inclusión Local de archivos (LFI).

6.5.9 M9 Improper Session Handling. Durante las pruebas relacionadas con este riesgo, se evidenció que la aplicación presenta varios problemas relacionados con el manejo de sesiones:

En primer lugar, la funcionalidad que permite a un usuario salir de la aplicación no realiza una adecuada eliminación o limpieza de los datos del usuario, los cuales son almacenados en un conjunto de variables estáticas públicas. Esto se traduce en problemas de seguridad que podrían ser aprovechados por un atacante para llevar a cabo ataques de evasión de la autenticación y escalamiento de privilegios.

Del lado del servidor, se encontró que es posible acceder directamente a los Web Services, sin tener que enviar credenciales o algún tipo de token que permita identificar el origen de las peticiones y el usuario que las realiza.

6.5.10 M10 Lack Of Binary Protection. A partir de la revisión del código de la aplicación, se encontró que ésta ofrece un bajo nivel de protección de sus binarios, ya que:

La aplicación no tiene mecanismos adecuados de ofuscación y protección de su código. Fue posible decompilar el APK proporcionado por la empresa, obteniéndose gran parte del código en actividades del tipo Activity, con nombres de métodos y atributos (variables) son su nombre original, lo que facilita a un atacante comprender la lógica del negocio. Este problema se debe a que la propiedad minifyEnabled en el archivo build.gradle está con un valor false[180].

No existen controles de Checksum y detección de posibles modificaciones al código de la aplicación.

El APK proporcionado es una versión de la aplicación que es depurable. La propiedad Android:debuggable no es explícitamente establecida a true en el archivo AndroidManifest.xml o en el código de la aplicación, por lo que se podría decir que no es depurable, pues el valor por defecto a false[181]. Sin embargo, el APK no fue firmado apropiadamente, además de ser una versión de depuración, lo que implica que el comportamiento de la aplicación se puede analizar y alterar utilizando herramientas especializadas para este propósito, como JDWP.

No se encontró ningún fragmento de código para verificar si el dispositivo en el que se está ejecutando la aplicación se encuentra rooteado e interrumpir su ejecución en caso de ser así. Básicamente se trató de encontrar fragmentos de código en los que se valide la presencia de certificados OTA, comandos del tipo SU, y algunos APK que sólo se ejecutan en dispositivos rooteados[182].

Como queda evidenciado, la aplicación presenta por lo menos algún tipo de vulnerabilidad en 9 de los 10 riesgos analizados. En el riesgo M6, donde se hizo una búsqueda de posibles problemas relacionados con la implementación de mecanismos de encriptación, no se encontró ningún problema, no porque la aplicación no implemente algoritmos de encriptación desaconsejados, sino porque no implementa ninguno durante el almacenamiento de datos sensibles en el

[180] ANDROID DEVELOPERS. Shrink Your Code and Resources. [En línea][Citado el 14 de diciembre, 2016]. Disponible en internet: https://developer.android.com/studio/build/shrink-code.html#shrink-code

[181] Android Developers. Application Element. [En línea][Revisado el 12 de diciembre, 2016]. Disponible en internet: https://developer.android.com/guide/topics/manifest/application-element.html

[182] OWASP. Mobile Top 10 2014-M10. 2014. [En línea][Revisado el 12 de diciembre, 2016] Disponible en internet https://www.owasp.org/index.php/Mobile_Top_10_2014-M10

dispositivo y en los procesos de comunicación entre la aplicación y el sistema de Backend con el que se comunica, lo cual es un problema mayor.

La siguiente tabla muestra los resultados para cada una de las pruebas ejecutadas, marcando si la aplicación móvil es vulnerable:

Tabla 43. Resultados de las pruebas ejecutadas en la aplicación.

Lista de chequeo del lado del cliente.		
M10. Falta de protección de los binarios		
Test ID	**Nombre de Prueba**	**Vulnerable**
MSTG-01	La aplicación es vulnerable a ataques de Ingeniería Inversa	Si
MSTG-02	La aplicación es depurable	Si
MSTG-03	Falta de controles de Checksum y detección de modificación de código	Si
MSTG-04	La aplicación se puede ejecutar en un dispositivo rooteado	Si
M2. Almacenamiento de datos Inseguro		
MSTG-05	Almacenamiento de datos sensibles en archivos de Logs	Si
MSTG-06	Credenciales quemadas en el código (Hard coded)	No

Tabla 44. (Continuación)

M2. Almacenamiento de datos inseguro		
Test ID	**Nombre de Prueba**	**Vulnerable**
MSTG-07	Credenciales son almacenadas en la base de datos sin ser encriptadas	Si
MSTG-08	Acceso a Información sensible después de realizar un volcado de memoria	No
MSTG-09	Almacenamiento de datos sensibles fuera del Sandbox de la aplicación.	Si
MSTG-10	Almacenamiento de datos sensibles en caché	No
MSTG-11	Almacenamiento de datos sensibles en preferencias	Si
M3. Protección insuficiente en la capa de transporte		
MSTG-12	Información sensible es enviada como texto legible a través de la red	Si
MSTG-13	Datos sensibles son enviados como un parámetro del tipo querystring	Si
MSTG-14	Protocolos de red inseguros	Si

Tabla 45. (Continuación)

M4. Fuga de datos involuntaria		
Test ID	**Nombre de Prueba**	**Vulnerable**
MSTG-15	Exposición de datos en el log	Si
MSTG-16	Portapapeles (Clipboard) no es deshabilitado	Si
MSTG-17	“Application Backgrounding”	No
MSTG-18	“Url Caching” (peticiones y respuestas)	No
MSTG-19	Permisos de aplicación inseguros.	Si
M5. Autenticación y autorización pobres		
MSTG-20	Bloqueo de cuenta no implementado	Si
MSTG-21	La autenticación se puede evadir.	Si
MSTG-22	Escalamiento de privilegios	Si
MSTG-23	No implementación o implementación no apropiada de una sección para cambiar contraseñas.	Si
MSTG-24	Políticas de contraseñas débiles	Si
MSTG-25	Opción de autocompletar no está deshabilitada	Si
MSTG-26	Uso de IMEI/UDID falsos para la autenticación	Si

Tabla 46. (Continuación)

M5. Autenticación y autorización pobres		
Test ID	**Nombre de Prueba**	**Vulnerable**
MSTG-27	Opción “Recordar credenciales” activa	Si.
M6. Criptografía rota		
MSTG-28	Aplicación hace uso de Algoritmos de encriptación débiles	No
MSTG-29	Envío de información en claro a través de túnel SSL	No
MSTG-30	Almacenamiento de claves de encriptación localmente	No
M7. Inyecciones en lado del cliente		
MSTG-31	La aplicación es vulnerable a XSS	Si
MSTG-32	Validaciones del lado del cliente pueden ser evadidas.	Si
MSTG-33	Inyecciones SQL.	Si
MSTG-34	Fuga de datos en Content Providers	No
MSTG-35	Inclusión local de archivos (LFI).	Si

Tabla 47. (Continuación)

M8. Decisiones de seguridad basadas en entradas no confiables		
Test ID	**Nombre de Prueba**	**Vulnerable**
MSTG-36	Modificación de URL	Si
MSTG-37	Abuso de componentes a través de IPC Intents	No
MSTG-38	Uso indebido de URL Schemes	No
M9. Manejo Inapropiado de sesiones		
MSTG-39	Fijación de sesión	No
MSTG-40	La aplicación no implementa una opción para cerrar sesión	Si
MSTG-41	Uso de Cookies persistentes	Si
MSTG-42	Tokens inseguros	No

Fuente: El autor.

En la siguiente tabla se muestran los resultados de las pruebas ejecutadas en el servidor, marcando si éste es vulnerable o no:

Tabla 48. Resultados de las pruebas ejecutadas en el servidor.

Lista de chequeo del lado del servidor.		
M1. Debilidad en los controles del lado del servidor de la aplicación		
Test ID	**Nombre de Prueba**	**Vulnerable**
MSTG-43	Contraseñas legibles en respuestas desde Web Services	Si
MSTG-44	Referencia directa a recursos internos sin autenticación	Si
MSTG-45	Aplicación no tiene o implementa indebidamente un sistema de manejo de sesiones	Si
MSTG-46	Falta de protección adecuada por “Timeout”	Si
MSTG-47	Validación incorrecta de entradas en el lado del servidor	Si
MSTG-48	Se muestra información interna sensible en páginas de error	Si
MSTG-49	Subida de archivos maliciosos en el servidor	Si
MSTG-50	XSS (Cross Site Scripting)	Si
MSTG-51	CORS (Cross Origin Resource Sharing)	No
MSTG-52	Es posible realizar peticiones HTTP diferentes a GET y POST	Si

Tabla 49. (Continuación)

M1. Debilidad en los controles del lado del servidor de la aplicación		
Test ID	**Nombre de Prueba**	**Vulnerable**
MSTG-53	Falsificación de petición en sitios cruzados (Cross Site Request Forgery (CSRF)/SSRF)	Si
MSTG-54	Respuestas HTTPS almacenadas en caché	Si
MSTG-55	Propiedad o atributo “Path” no establecido en Cookies	Si
MSTG-56	Cookie Sin HttpOnly habilitado	Si
MSTG-57	Propiedad o atributo “Secure” no establecido en Cookies	Si
MSTG-58	Es posible hacer “Fingerprinting” del Servidor y/o el Sistema Operativo	Si

Fuente: El autor.

6.6 DIVULGACIÓN DE RESULTADOS

Por tratarse de una información que expone las debilidades de seguridad de la aplicación Prism y el sistema de backend asociado, los resultados del análisis realizado serán entregados directamente a la empresa BarcodeApps, se hará la recomendación que la información sea marcada como confidencial y que sea tratada como tal, a su vez la empresa será la responsable de su respectivo tratamiento y divulgación.

Adicionalmente se entrega documento correspondiente al proyecto a la Universidad Nacional Abierta y A Distancia, este documento reposará en el repositorio de la institución. La información del documento solo debe ser consultada para fines académicos y en ningún momento puede ser utilizada para atentar en contra de la información de la empresa BarcodeApps, de sus clientes, u otras entidades o personas relacionadas directa o indirectamente con ella.

7. CONCLUSIONES

El estudio de los riesgos establecidos en el Proyecto de seguridad móvil de la OWASP, permitió conocer los principales problemas que se pueden encontrar en una aplicación móvil y los impactos que estos pueden traer a nivel técnico y de negocio. El estudio fue clave para afrontar el reto de realizar el análisis de seguridad de la aplicación Prism Android y del sistema de backend asociado, ya que brindó las bases necesarias para establecer un plan de trabajo, sin que se perdiera el foco de la investigación.

Al revisar cada uno de los riesgos establecidos en el Proyecto de Seguridad móvil de la OWASP, se fueron identificando los principales vectores de ataques utilizados por los atacantes y el nivel de explotabilidad, y a partir de ello se construyó una lista de chequeo, teniendo en cuenta el sistema operativo al que está dirigida la aplicación, y la necesidad de BarcodeApps de conocer qué tan protegidos están los datos que la aplicación almacena e intercambia con el sistema de backend asociado. Al final la lista de chequeo resultó con 58 casos de pruebas, 42 casos a revisar en la aplicación y 16 en el servidor.

Durante el proceso quedó evidenciando la gran utilidad e importancia que tuvieron las herramientas utilizadas, que independientemente del hecho que todas fueron gratuitas, permitieron realizar un profundo análisis de la aplicación móvil, para llegar a obtener resultados rápidos, coherentes, y respaldados. Sin lugar a dudas, las pruebas automáticas que ofrecen estas herramientas, reducen la cantidad de tiempo que se tarda un analista en probar una aplicación, y le permiten capturar las evidencias necesarias para mostrar a los clientes.

De las 19 herramientas definidas inicialmente, se destacan ADB y el mismo IDE de Android Studio, que permitieron encontrar problemas relacionados con la exposición y almacenamiento de datos sensibles sin encriptar; Burp Suite, para encontrar problemas relacionados con la transmisión de datos sensibles a través de la red y el manejo inadecuado de sesiones; ApkTools Dex2jar, y Androguard, para hacer ingeniería inversa; Drozer, para manipular el código de la aplicación y saltar sus procesos de autenticación; y finalmente W3af y Nmap, para encontrar problemas en el servidor.

El nivel de seguridad de la aplicación Prism Android y del sistema de backend asociado, es considerablemente bajo, ya que de los 58 casos de prueba analizados, 46 representan una vulnerabilidad. Se encontraron problemas en 9 de los 10 riesgos establecidos en el proyecto OWASP, con la salvedad que en el riesgo M6 - Broken Cryptography (Criptografía rota)-, donde se hizo una búsqueda de posibles vulnerabilidades relacionadas con la implementación de mecanismos de encriptación, no se reportaron problemas, no porque la aplicación no implemente algoritmos de encriptación desaconsejados, sino porque en la misma no se implementa ninguno. El almacenamiento y transferencia de datos sensibles se hace sin encriptar la información, lo que representa problemas aún mayores. Estos problemas se reportaron en los resultados de las pruebas relacionadas con los riesgos M2 - Insecure Data Storage (almacenamiento de datos inseguro) y M3 - Insufficient Transport Layer Protection (Protección insuficiente en la capa de transporte).

Es necesario que la empresa BarcodeApps tome las medidas necesarias para corregir cada uno de los problemas reportados, y realizar análisis de seguridad periódicos (por lo menos una vez al año) y/o a medida que se incluyen nuevas funcionalidades críticas.

La empresa también debe tener en cuenta todas las recomendaciones dadas en la ejecución de nuevos proyectos, e incluir la seguridad en el inicio y en todo el ciclo de vida de desarrollo del software, pues entre más temprano se identifique una falla de seguridad en el software, más rápida y económica será su mitigación.

Desde las primeras fases del ciclo de desarrollo de software se deben establecer requerimientos y controles de seguridad verificables y medibles, los cuales surgen a partir de la definición de los requerimientos funcionales y los posibles impactos que estos tendrán en la seguridad de la información, teniendo en cuenta la naturaleza de la misma, la arquitectura de la aplicación y las plataformas en las que ésta se desplegará, los perfiles de usuarios y los diferentes sistemas de autenticación, entre otros.

Todo el trabajo llevado a cabo permitió afianzar y aplicar los conocimientos adquiridos durante el periodo académico de la especialización y cumplir con uno de los requisitos para obtener el título como Especialista en Seguridad Informática, pero también para acumular la experiencia necesaria para seguir realizando este tipo de pruebas de penetración en aplicaciones móviles, ya no de manera

académica, sino como un servicio ofrecido a las empresas de desarrollo de Software en Colombia, en las que se está empezando a ver la seguridad como un factor crucial para la calidad del software desarrollado. Esto último, fue una de las principales motivaciones para hacer el planteamiento de la propuesta de grado, y ahora que es un hecho, significa una gran satisfacción.

8. RECOMENDACIONES Y TRABAJOS FUTUROS

Dependiendo del grado de aceptación de los resultados y la implementación de las recomendaciones por parte de la empresa, se estudiará la posibilidad de realizar un análisis de seguridad más exhaustivo del conjunto de servidores en los que se alojan los web services y otros sistemas web relacionados con la aplicación Prism para Android. Mediante un proceso similar al mostrado en el presente documento, se espera planear y ejecutar un plan de pruebas basado en el Top 10 de riesgos establecidos por la OWASP para el análisis de seguridad de sistemas web, con lo que sin duda se reforzarán los resultados obtenidos durante el análisis de seguridad del sistema de backend asociado a la aplicación Prism para Android.

Por otro lado, los resultados obtenidos en esta investigación proporcionan un punto de partida para que otros profesionales continúen el desarrollo de proyectos futuros que aporten a mejorar la seguridad en los dispositivos móviles. En primera instancia se propone el análisis de seguridad de aplicaciones en versiones más recientes de Android o en otros sistemas operativos móviles, y muy interesante puede resultar el análisis de aplicaciones híbridas, las cuales están teniendo un crecimiento exponencial.

BIBLIOGRAFÍA

ALCALDÍA MAYOR DE BOGOTÁ. LEY ESTATUTARIA 1581 DE 2012. 2012. [En línea][Citado el 20 de octubre, 2017]. Disponible en internet: http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=49981.

Androbugs Framework. 2016. [En línea][Revisado el 12 de septiembre, 2016]. Disponible en internet: https://github.com/AndroBugs/AndroBugs_Framework.

ANDRODEITY. Arquitectura de Android. 2011. [En línea][Revisado el 12 de mayo, 2016]. Disponible en internet: http://androideity.com/2011/07/04/arquitectura-de-android/.

ANDROID DEVELOPERS. Application Element. [En línea][Revisado el 12 de diciembre, 2016]. Disponible en internet: https://developer.android.com/guide/topics/manifest/application-element.html

--------. Android Debug Bridge. [En línea][Citado el 11 de septiembre, 2016]. Disponible en internet:https://developer.android.com/studio/command-line/adb.html.

--------. Android Studio. [En línea][Citado el 12 de mayo, 2016]. Disponible en internet: https://developer.android.com/studio/index.html?hl=es-419

--------. Biblioteca de compatibilidad. [En línea][Citado el 13 de mayo, 2016]. Disponible en internet: https://developer.android.com/topic/libraries/support-library/index.html

--------. Manifest Permissions. 2016. [En línea][Citado el 15 de mayo, 2016]. Disponible en internet: https://developer.android.com/reference/android/Manifest.permission.html

--------. Paneles de control. [En línea][Citado el 16 de mayo, 2016]. Disponible en https://developer.android.com/about/dashboards/index.html.

--------. Security Tips. [En línea][Citado el 18 de octubre, 2016]. Disponible en internet: https://developer.android.com/training/articles/security-tips.html.

-------. Shrink Your Code and Resources. [En línea][Citado el 14 de diciembre, 2016]. Disponible en internet: https://developer.android.com/studio/build/shrink-code.html#shrink-code

ANDROID EXPERTOS. ¿Android stock o modificado? ¿Cuáles son las diferencias?. 2014. [En línea][Citado el 16 de mayo, 2016]. http://www.androidexperto.com/aprender-android/android-stock-modificado-diferencias/.

ANDROIDPIT. ¿Qué es el número del firmware?. [En línea][Citado el 16 de mayo, 2016]. Disponible en internet: http://www.androidpit.es/que-es-numero-firmware

APKTOOL. [En línea][Citado el 16 de mayo, 2016]. Disponible en internet: https://ibotpeaches.github.io/Apktool.

AVENDAÑO, Marco. Android Security Fundamentales. 2015.[En línea][Citado el 16 de mayo, 2016]. Disponible en internet: http://es.slideshare.net/Marcoviaweb/android-security-54094428.

BARCODEAPPS. Prism Features and Glance. [En línea][Citado el 21 de octubre, 2017]. Disponible en internet: http://prism.basisinventory.com/.

BORGHELLO, Cristian. OWASP Mobile Security Project (Español). 2015 Disponible en: http://blog.segu-info.com.ar/2015/03/owasp-mobile-security-espanol.html

BRITO, Cristian. Metodologías para desarrollar software seguro. 2013. [En línea][Citado el 16 de mayo, 2016]. Disponible en internet: http://recibe.cucei.udg.mx/revista/es/vol2-3/pdf/computacion05.pdf?ver=24062013

BROWN, M. Secure software development: Why the development world awoke to the challenge. Information Security Technical Report, Vol. 13, No. 1, pp. 40 – 43.

CARTER, Jonathan y SINGH, Milan. OWASP Mobile Application Security Guide. . [En línea][Citado el 25 de mayo, 2016] Disponible en internet: https://drive.google.com/file/d/0BxOPagp1jPHWczhwYjRQNzZIekU/view.

CERDA, Alberto. Guías Legales Editores. 2011. [En línea][Citado el 29 de mayo, 2016]. Disponible en internet https://www.derechosdigitales.org/wp-content/uploads/guias_legales_editores.pdf.

CHELL, D. ERASMUS, T, & COLLEY, S. The Mobile Application Hacker's Handbook. [En línea][Citado el 16 de octubre, 2016] Disponible en internet: https://books.google.com.co/books?id=lgNhBgAAQBAJ&pg=PA139&lpg=PA139&dq=application+background+screenshot&source=bl&ots=lLmx0bbgpB&sig=LDlPnrNx_VAbi-mESTBAufFMXRk&hl=es-419&sa=X&ved=0ahUKEwi7ifG26-fPAhVHYiYKHVx1DAE4FBDoAQhBMAU#v=onepage&q=application%20background%20screenshot&f=false.

COLORADO, Pedro y TORRES, Inírida. Análisis de seguridad de aplicaciones móviles nativas para el sistema operativo Android versión Jelly Bean 4.1.2 en dispositivos móviles Smartphone. 2015. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: repository.unad.edu.co/bitstream/10596/3412/1/40186727.pdf.

DECOMPILING ANDROID, Top 10 Mobile Security Risks. 2014. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: http://www.decompilingandroid.com/mobile-app-security/top-10-mobile-security-risks/.

DESNOS, Antony. AndroGuard. [En línea][Citado el 20 de septiembre, 2016] Disponible en internet: https://github.com/androguard/androguard.

EBAY INC. Barcode Scanning SDK for iOS and Android Documentation. 2012. [En línea][Citado el 20 de mayo, 2016] Disponible en internet: http://redlaser.com/developers/documentation/.

GARCÍA, Jorge. Lo que debes saber sobre el contrato de escrow. 2014. [En línea][Citado el 23 de mayo, 2016] Disponible en internet: http://mprende.co/legal/lo-que-debes-saber-sobre-el-contrato-de-escrow.

GITHUB. Mobile Security Framework. 2016. [En línea][Citado el 17 de noviembre, 2016] Disponible en internet: https://github.com/ajinabraham/Mobile-Security-Framework-MobSF.

GOGI, Rahul. What are the main differences between stock Android and CyandohendMod. [En línea][Citado el 17 de mayo, 2016] Disponible en internet: https://www.quora.com/What-are-the-main-differences-between-stock-Android-and-CyanogenMod.

GOOGLE. Acuerdo de Distribución para Desarrolladores de Google.2016. [En línea][Citado el 12 de junio, 2016] Disponible en internet: Playhttps://play.google.com/intl/all_es/about/developer-distribution-agreement.html.

INFOSEC INSTITUTE. INTRODUCTION TO DROZER. 2014. [En línea][Revisado el 23 de septiembre, 2016] Disponible en internet: http://resources.infosecinstitute.com/android-hacking-security-part-13-introduction-drozer/.

INGENIERÍA COGNITIVA. Arquitectura general Android. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: http://ingenieriacognitiva.com/developer/cursos/AndroidBasico/chapters/c2.php.

INTERNATIONAL DATA CORPORTATION. Smartphone Volumes Expected to Rebound in 2017 with a Five-Year Growth Rate of 3.8%, Driving Annual Shipments to 1.53 Billion by 2021, According to IDC. 2017. [En línea][Citado el 21 de octubre, 2017]Disponible en internet: https://www.idc.com/getdoc.jsp?containerId=prUS42334717.

LEYES Y REGLAMENTOS / COMUNIDAD ANDINA DE NACIONES. Decisión 351 de 1993 Régimen común sobre Derecho de Autor y derechos conexos. 1993. [En línea][Citado el 12 de junio, 2016] Disponible en internet: http://www.cerlalc.org/derechoenlinea/dar/leyes_reglamentos/Colombia/Dec_andina_351.htm.

LEYES Y REGLAMENTOS / ESPAÑA. Capítulo XI De los delitos relativos a la propiedad intelectual e industrial, al mercado y a los consumidores. 1995. [En línea][Citado el 12 de junio, 2016] Disponible en internet: http://www.cerlalc.org/derechoenlinea/dar/leyes_reglamentos/Espana/Ley_organica.htm.

LÓPEZ, Ilya. Top 10 de OWASP de vulnerabilidades en aplicaciones móviles. 2014. [En línea][Citado el 25 de mayo, 2016] Disponible en internet: http://www.welivesecurity.com/la-es/2014/02/26/top-10-owasp-vulnerabilidades-aplicaciones-moviles/.

CEBALLOS, Julián y MARULANDA, Cesar. Una Revisión de metodologías seguras en cada fase del ciclo de desarrollo de desarrollo de software. 2012. [En línea][Citado el 25 de mayo, 2016] Disponible en internet: http://web.usbmed.edu.co/usbmed/fing/v3n1/v3n1a2.pdf.

MINISTERIO DE COMERCIO, INDUSTRIA Y TURISMO. Decreto Número 1377 de 2013. 2013. [En línea][Citado el 21 de octubre, 2017]. Disponible en: https://www.mintic.gov.co/portal/604/articles-4274_documento.pdf.

MINTIC. Ley No 1273. 2009. [En línea][Citado el 12 de junio, 2016] Disponible en internet: http://www.mintic.gov.co/portal/604/articles-3705_documento.pdf.

MOVILION. Aplicaciones móviles híbridas: bomba de tiempo en materia de seguridad. 2014. [En línea][Citado el 27 de mayo, 2016] Disponible en internet: http://www.movilion.com/aplicaciones-moviles-hibridas-seguridad/.

NMAP ORG. Nmap (Network mapper). [En línea][Revisado el 17 de septiembre, 2016] Disponible en internet: https://nmap.org/.

OLARTE, Jorge y ROJAS, Miguel. La protección del derecho de autor y los derechos conexos en el ámbito penal. 2010. [En línea][Citado el 12 de junio, 2016] Disponible en internet: http://www.derechodeautor.gov.co/documents/10181/11769/La+proteccin+del+derecho+de+autor+y+los+derechos+conexos+en+el+ambito+penal+sep+15+de+2010.pdf/75686fc1-c9be-4dc3-b1d5-efcd5f4be949.

ORACLE. JDWP. 2016. [En línea][Citado el 17 de septiembre, 2016] Disponible en internet: https://docs.oracle.com/javase/8/docs/technotes/guides/troubleshoot/introclientissues005.html.

OWASP. Android Testing Cheat Sheet. 2016. [En línea][Revisado el 9 de octubre, 2016] Disponible en internet: https://www.owasp.org/index.php/Android_Testing_Cheat_Sheet.

-------. Category: Vulnerability. 2016. [En línea][Citado el 28 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Category:Vulnerability.

-------. Clickjacking. 2015. [En línea][Citado el 28 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Clickjacking.

-------. Sobre OWASP. 2014. [En línea][Citado el 30 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Sobre_OWASP.

-------. Mobile Checklist Final 2016. 2016. [En línea][Citado el 11 de octubre, 2016] Disponible en internet: https://drive.google.com/file/d/0BxOPagp1jPHWYmg3Y3BfLVhMcmc/view.

-------. Mobile Top 10 2014-M1. 2014. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M1.

-------. Mobile Top 10 2014-M2. 2014. [En línea][Citado el 16 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M2.

-------. Mobile Top 10 2014-M3. 2014. [En línea][Revisado el 16 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M3.

-------. Mobile Top 10 2014-M4. 2014. [En línea][Revisado el 16 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M4.

-------. Mobile Top 10 2014-M5. 2014. [En línea][Citado el 18 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M5.

-------. Mobile Top 10 2014-M6. 2014. [En línea][Citado el 19 de mayo, 2016] Disponible en internet: https://www.owasp.org/index.php/Mobile_Top_10_2014-M6.

-------. Mobile Top 10 2014-M7. 2014. [En línea][Revisado el 17 de mayo, 2016] Disponible en internet https://www.owasp.org/index.php/Mobile_Top_10_2014-M7.

-------. Mobile Top 10 2014-M8. 2014. [En línea][Citado el 17 de mayo, 2016] Disponible en internet https://www.owasp.org/index.php/Mobile_Top_10_2014-M8.

-------. Mobile Top 10 2014-M9. 2014. [En línea][Citado el 20 de mayo, 2016] Disponible en internet https://www.owasp.org/index.php/Mobile_Top_10_2014-M9.

-------. Mobile Top 10 2014-M10. 2014. [En línea][Revisado el 21 de mayo, 2016] Disponible en internet https://www.owasp.org/index.php/Mobile_Top_10_2014-M10.

-------. OWASP Mobile Security Project - Security Testing Guide. 2013. [En línea][Revisado el 16 de septiembre, 2016] Disponible en internet https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Security_Testing_Guide

OWASP Project. 2014. [En línea][Revisado el 17 de mayo, 2016] Disponible en internet https://www.owasp.org/index.php/Category:OWASP_Project.

OWASP Mobile Security Project. 2016. [En línea][Revisado el 17 de mayo, 2016] Disponible en internet https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks

OWASP. Mobile Security Project - M-Tools. 2016. [En línea][Revisado el 19 de septiembre, 2016] Disponible en internet https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=M-Tools

RSA. Phishing, Vishing and Smishing: Old Threats Present new Risks. [En línea][Citado el 29 de mayo, 2016] Disponible en internet https://www.rsa.com/content/dam/rsa/PDF/h11933-wp-phishing-vishing-smishing.pdf

PICO, José. Amenazas de seguridad en comunicaciones de datos móviles. 2011. [En línea][Revisado el 29 de mayo, 2016] Disponible en internet: http://es.slideshare.net/eventoscreativos/amenazas-de-seguridad-en-comunicaciones-de-datos-mviles).

PORTSWIGGER. Burp Suite. [En línea][Revisado el 22 de septiembre, 2016] Disponible en internet: https://portswigger.net/burp/.

RAMÍREZ, Mauricio. La Seguridad En Aplicaciones Móviles: Estrategias En El Mundo Actual. [En línea][Revisado el 26 de mayo, 2016] Disponible en internet http://datateca.unad.edu.co/contenidos/233016/Articulos/La_Seguridad_en_Aplicaciones_Moviles_Estrategias_en_el_Mundo_Actual_Gabriel_Ramirez.pdf.

-------. Seguridad en aplicaciones móviles. 2013. [En línea][Citado el 22 de mayo, 2016] Disponible en internet http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/index.html.

RACCIATI. Hernán. Modelado de Amenazas, una introducción. 2012. [En línea][Revisado el 22 de mayo, 2016] Disponible en internet https://www.owasp.org/images/8/8e/HRacciatti_ModeladodeAmenazas.pdf.

ROUSE, Margaret. Jailbreaking. 2013. [En línea][Citado el 27 de mayo, 2016] Disponible en internet: http://whatis.techtarget.com/definition/jailbreaking.

RYDSTEDT, Gustav. Framing Attacks on Smart Phones and Dumb Routers: Tap-jacking and Geo-localization Attacks. [En línea][Revisado el 19 de mayo, 2016] Disponible en internet: http://seclab.stanford.edu/websec/framebusting/tapjacking.pdf.

SCOTT, Alexander. Android Security: Adding Tampering Detection to Your App. [En línea][Revisado el 27 de agosto, 2016] Disponible en internet https://www.airpair.com/android/posts/adding-tampering-detection-to-your-android-app.

SECRETARÍA GENERAL DE LA ALCALDÍA MAYOR DE BOGOTÁ D.C. Circular Conjunta 1 de 2006 Procuraduría General de la Nación 2006. 2006. [En línea][Revisado el 7 de julio, 2016] Disponible en internet: http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=21006.

SECRETARÍA GENERAL DE LA ALCALDÍA MAYOR DE BOGOTÁ D.C. Decreto 1360 de 1989 Nivel Nacional. 1989. [En línea][Citado el 12 de julio, 2016] Disponible en internet: http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=10575.

SILGADO, Felipe. Riesgos de seguridad de la información en el uso de dispositivos móviles para aplicaciones empresariales. 2014. [En línea][Revisado el 16 de mayo, 2016] Disponible en internet: https://www.pwc.com/co/es/assets/document/mision.pdf.

SOCKET MOBILE. Scan API Reference. 2016. [En línea][Revisado el 16 de octubre, 2016] Disponible en internet: http://www.socketmobile.com/docs/default-source/developer-documentation/scanapi.pdf?sfvrsn=2.

SOLIDPADD. Man-in-the-Phone Attacks (Man-in-the-Mobile/MitMo Attacks). [En línea][Revisado el 28 de mayo, 2016] Disponible en internet: http://www.solidpass.com/threats/man-in-the-phone-mitp-attacks.html.

Sourceforge. Dex2jar. 2016. [En línea][Revisado el 23 de septiembre, 2016] Disponible en internet: https://sourceforge.net/projects/dex2jar/.

SQUARE INC. Picasso - A powerful image downloading and caching library for Android. 2013. [En línea][Revisado el 16 de septiembre, 2016] Disponible en internet: http://square.github.io/picasso/.

TAYLOR, A. JD-GUI. 2015. [En línea][Revisado el 16 de septiembre, 2016] Disponible en internet: https://github.com/java-decompiler/jd-gui.

TARLOGIC. Auditoría de aplicaciones móviles. [En línea][Revisado el 16 de septiembre, 2016] Disponible en internet: https://www.tarlogic.com/servicios/auditoria-de-aplicaciones-moviles/.

PUJA S., SHELKE C.J. A Survey on different Attacks on Mobile Devices and its Security. 2014. [En línea][Citado el 18 de mayo, 2016] Disponible en internet: http://www.ijaiem.org/volume3issue2/IJAIEM-2014-02-28-080.pdf.

THE APACHE SOFTWARE FOUNDATION. Apache HttpComponents. 2016. [En línea][Revisado el 11 de septiembre, 2016] Disponible en internet: https://hc.apache.org/

THE LLDB DEBUGGER. [En línea][Revisado el 16 de septiembre, 2016] Disponible en internet: http://lldb.llvm.org/remote.html.

THE POWER OF WINGS. Abusing the Intent URL Scheme Redux. 2015. [En línea][Citado el 22 de octubre, 2016] Disponible en internet: http://rotlogix.com/2015/05/09/the-power-of-wings-abusing-the-intent-url-scheme/.

TUTORIALESPOINT. Android – Architecture. [En línea][Revisado el 1 de julio, 2016] Disponible en internet: http://www.tutorialspoint.com/android/android_architecture.htm.

VAN, Rutger. iOS App Security - Backgrounding screenshot. 2013. [En línea][Revisado el 28 de octubre, 2016] Disponible en internet: https://technology.amis.nl/2013/05/03/ios-app-security-backgrounding-screenshot/

UNIVERSITY OF MARYLAND. FindBugs in Java Programs. 2015. [En línea][Revisado el 28 de septiembre, 2016] Disponible en internet: http://findbugs.sourceforge.net/.

Wa3f. 2013. [En línea][Revisado el 2 de octubre, 2016] Disponible en internet: http://w3af.org/.

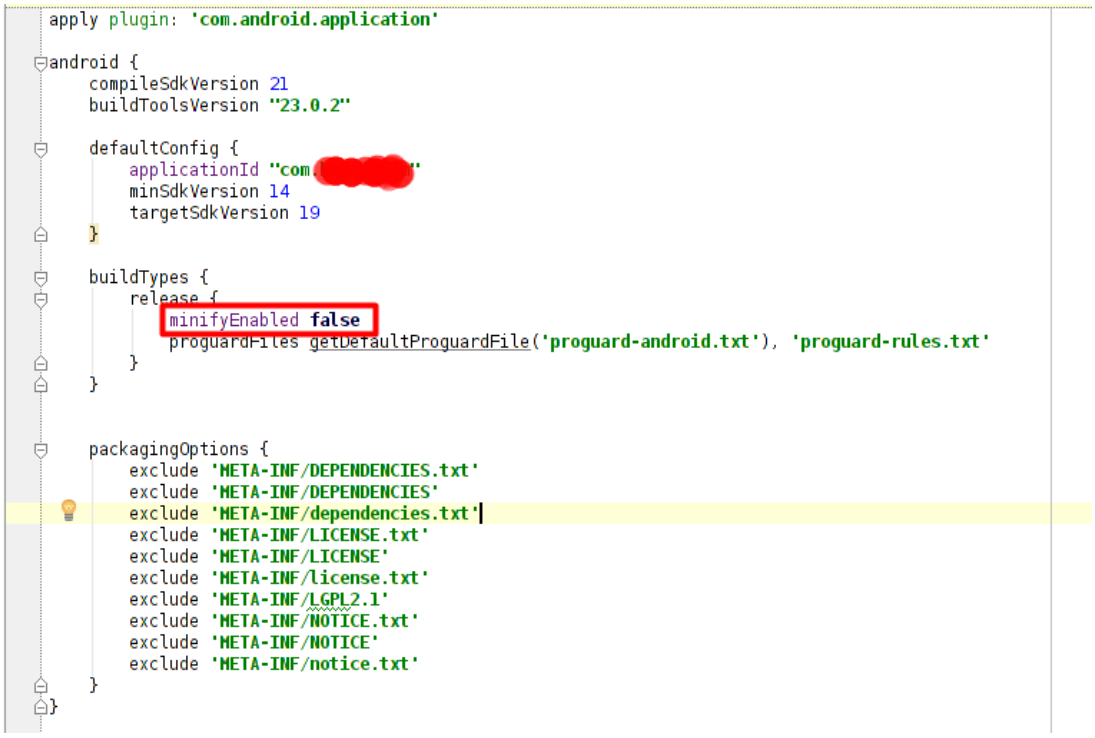
ZANDBERGUEN, F. URL Schemes for IOS and Android. 2014. [En línea][Revisado el 21 de octubre, 2016] Disponible en internet: http://fokkezb.nl/2013/08/26/url-schemes-for-ios-and-android-1/.

ANEXOS.

Anexo A. Resultados de ejecución de las pruebas.

- **MSTG-01. La aplicación es vulnerable a ataques de Ingeniería Inversa.** Al revisar uno de los archivos de configuración del proyecto proporcionado por BarcodeApps, se evidencia que la opción **minifyEnabled** se encuentra establecida a false, por lo que al generarse el APK, éste no es optimizado con Proguard, que es una herramienta que además de detectar librerías, clases, métodos, y/o atributos innecesarios en el proyecto; también ofusca el código de la aplicación, convirtiendo los nombres de clases, métodos, y/o atributos en nombres generalmente de un solo carácter, lo que dificulta los procesos de ingeniera inversa.

Figura 28. Configuración propiedad minifyEnabled.



```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "23.0.2"

    defaultConfig {
        applicationId "com[redacted]"
        minSdkVersion 14
        targetSdkVersion 19
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.txt'
        }
    }


    packagingOptions {
        exclude 'META-INF/DEPENDENCIES.txt'
        exclude 'META-INF/DEPENDENCIES'
        exclude 'META-INF/dependencies.txt'
        exclude 'META-INF/LICENSE.txt'
        exclude 'META-INF/LICENSE'
        exclude 'META-INF/license.txt'
        exclude 'META-INF/LGPL2.1'
        exclude 'META-INF/NOTICE.txt'
        exclude 'META-INF/NOTICE'
        exclude 'META-INF/notice.txt'
    }
}
```

Fuente: El Autor.

Al realizar una decompilación de la aplicación se puede evidenciar lo descrito en el párrafo anterior. Prácticamente todas las clases pudieron ser decompiladas, e incluso se puede leer gran parte de la lógica de la aplicación, por ejemplo en la clase encargada de gestionar la autenticación de usuario:

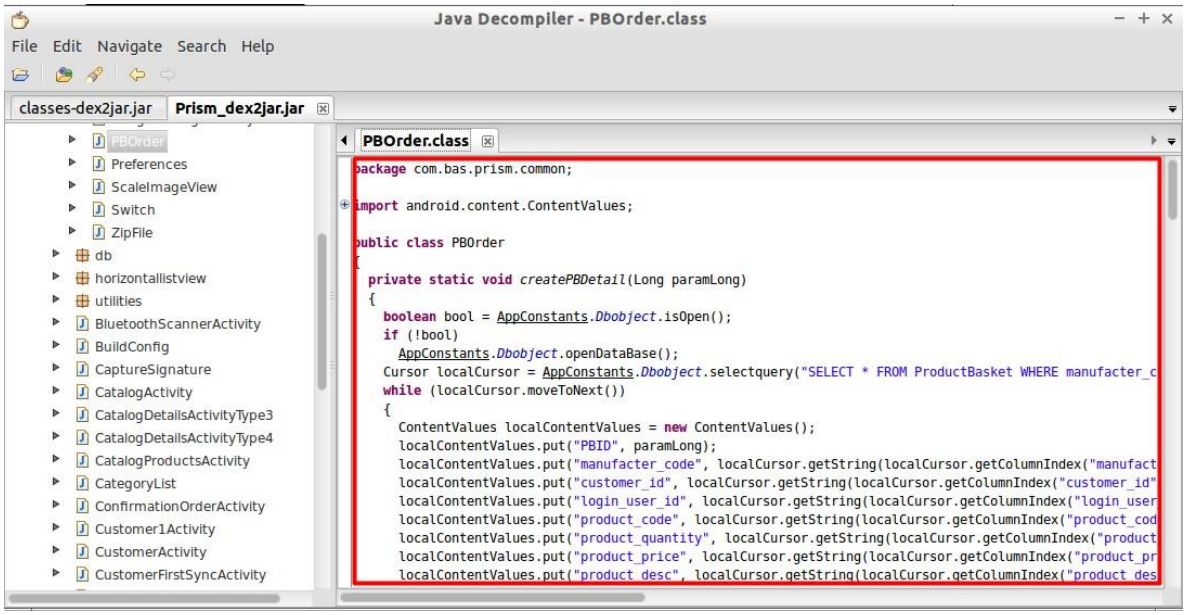
Figura 29. Código no ofuscado en clase del Tipo Activity.

Fuente: El Autor.

Como se puede evidenciar en la figura anterior, hay gran parte del código con la lógica del negocio dentro de una clase del Tipo Activity, donde solo debería ubicar el código encargado de gestionar los componentes gráficos de la aplicación y los diferentes eventos desencadenados sobre ellos.

Adicionalmente se encontraron clases que no son del tipo Activity en las que también se encuentra lógica perfectamente legible:

Figura 30. Código no ofuscado en clase del tipo Object.



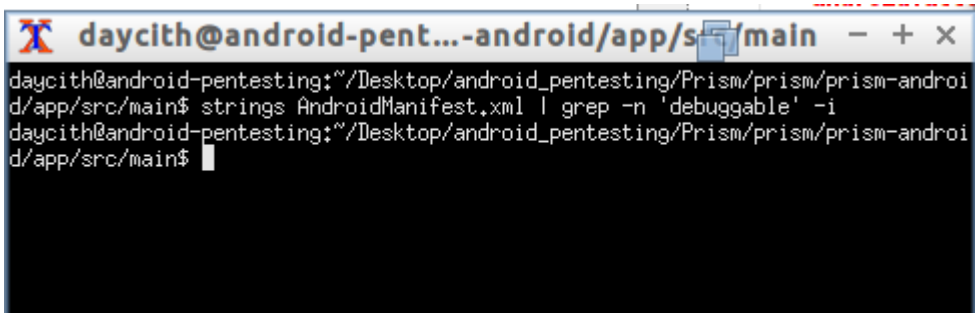
Fuente: El autor.

Recomendaciones. Optimizar el bytecode de la aplicación utilizando la herramienta proguard, la cual viene integrada en Android, sólo se debe activar la propiedad **minifyEnabled** en el archivo build.gradle de la aplicación. Una mejor alternativa a la opción anterior es utilizar una herramienta de optimización o protección más avanzada, como DexGuard.

Remover código con la lógica del negocio de las clases del tipo Activity, pues generalmente las herramientas de ofuscación generalmente no ofuscan su código. Este tipo de clases deberían contener solo el código para mostrar información al usuario y capturar los eventos desencadenados por estos.

- **MSTG-02. La aplicación es depurable.** En el archivo AndroidManifest.xml no se encuentra establecida la propiedad Android:debuggable:

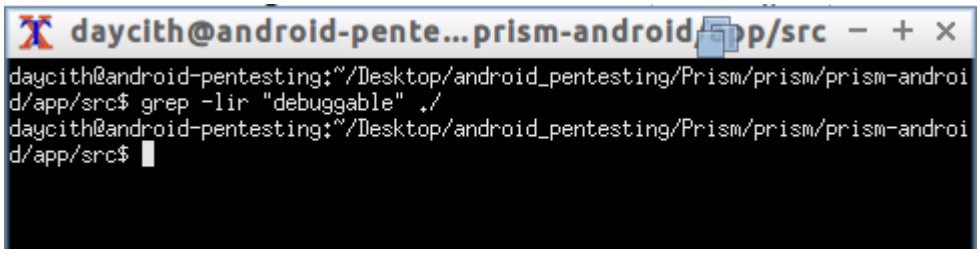
Figura 31. Verificación Propiedad Android:debuggable.



Fuente: El autor.

Tampoco se encuentra configuración y/o verificación de la propiedad en tiempo de ejecución:

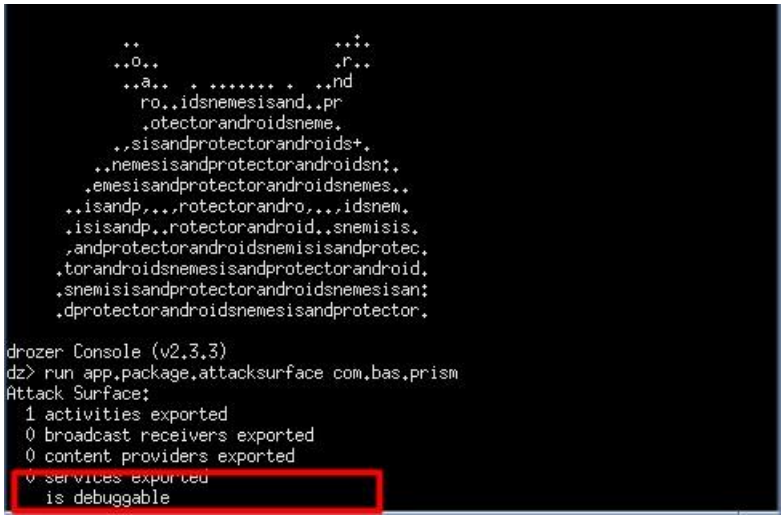
Figura 32. Búsqueda propiedad debuggable en el código.



Fuente: El autor.

Basado en lo anterior se podría pensar que la aplicación no es depurable, pues el valor por defecto es false. No obstante, al realizar un análisis con drozer, la herramienta indica que la aplicación es depurable:

Figura 33. Aplicación depurable de acuerdo escaneo automático.



Fuente: El autor.

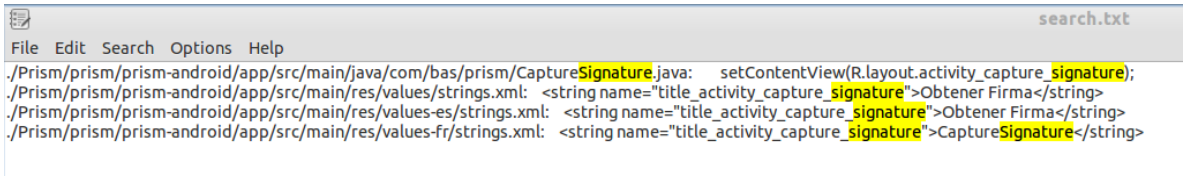
Lo que indica que el APK proporcionado por la empresa es una versión de depuración, y no es una versión oficial debidamente firmada.

Recomendaciones. Distribuir un APK de la aplicación debidamente firmado y optimizado, utilizando herramientas para optimización y ofuscación de código.

- **MSTG-03. Falta de controles de Checksum y detección de modificación de código.** Como se mencionó en la prueba anterior (ver Figuras 31 y 32), no se establece la propiedad debuggable en el archivo AndroidManifest.xml; pero tampoco se realiza verificación de la misma, lo que podría resultar en un problema de seguridad, en el caso en que un atacante decompile la aplicación, cambie el valor de la propiedad android:debuggable a true, y la vuelva a recompilar y ejecutar.

Por otra parte, dentro de la aplicación no se encuentra ningún fragmento de código en el que se trate de verificar la firma de la aplicación. Se encontró que existe una actividad en la que básicamente se permite al usuario seleccionar una imagen y adjuntarla como firma de una orden:

Figura 34. Búsqueda verificación firma de la aplicación.



Fuente: El autor.

Pero esto hace parte de la lógica del negocio de la aplicación, no corresponde a ninguna intención de protección frente a modificaciones de códigos no autorizados.

Como se evidenció durante la fase de recopilación de la información, la aplicación pudo ser ejecutada en el emulador, lo que implica que dentro de la aplicación no se realiza ninguna verificación del entorno de ejecución. La aplicación debería verificar si se está ejecutando en el emulador y tomar las medidas necesarias.

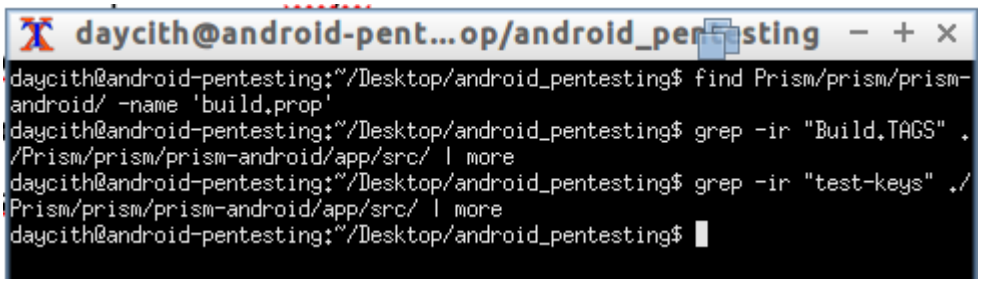
El hecho anterior también evidencia que en la aplicación no se verifica el instalador de la misma. Una aplicación en Android contiene información de su instalador, esto permite a sus desarrolladores definir cuáles son los métodos de instalación permitidos y validarlos dentro de la aplicación.

Recomendaciones. Implementar controles de Checksum dentro de la aplicación, que permitan verificar si ésta ha sido modificada y firmada por personas ajenas a la empresa, e interrumpir su ejecución en caso de detectar algún comportamiento sospechoso.

- **MSTG-04 La aplicación se puede ejecutar en un dispositivo rooteado.** Verificación de test-keys: Si la etiqueta test-keys se encuentra establecida en el archivo /system/build.prop del dispositivo, es un indicativo que el kernel del dispositivo ha sido firmado por un desarrollador no oficial, y que el dispositivo se encuentra rooteado.

Durante la revisión hecha en la aplicación no se encontró ninguna verificación de esta propiedad:

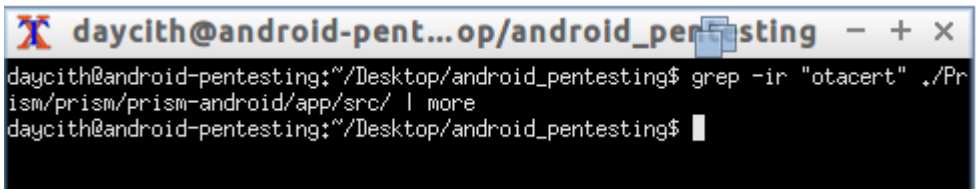
Figura 35. Búsqueda verificación test-keys.



Fuente: El autor.

Verificación de certificados OTA: Otro indicio de que el dispositivo se encuentra rooteado es la presencia de certificados OTA, que generalmente se trata del archivo ubicado en la ruta /etc/security/otacerts.zip.

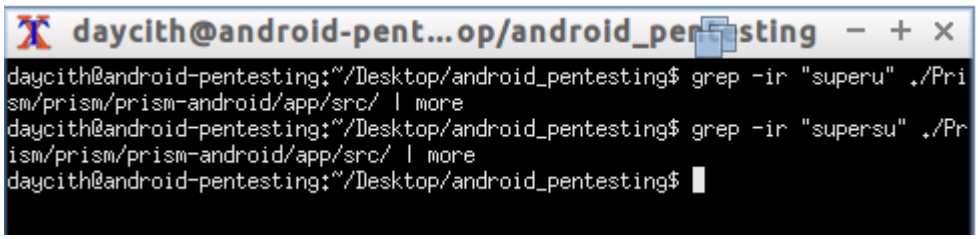
Figura 36. Búsqueda verificación certificados OTA.



Fuente: El autor.

Búsqueda de algunos APK´s conocidos: no se encontraron validaciones de presencia de aplicaciones que solo se ejecutan en dispositivos rooteados, por ejemplo com.noshufou.android.su, com.thirdparty.superuser, eu.chainfire.supersu, y com.koushikdutta.superuser:

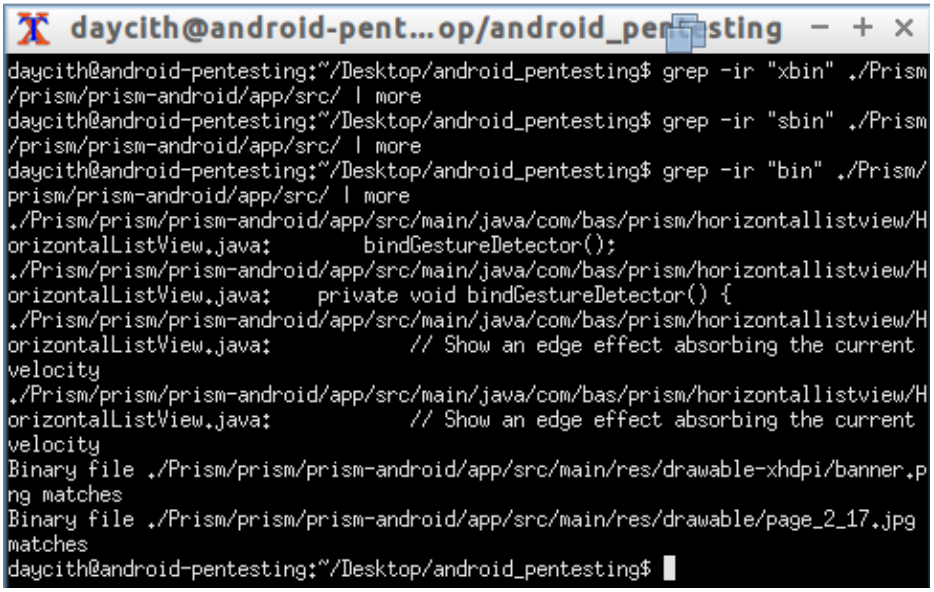
Figura 37. Búsqueda validación APK dispositivo rooteado.



Fuente: El autor.

Se buscaron comandos como “xbin”, “su”, “sbin”, “system”: La aplicación no valida la presencia de binarios SU y tampoco intenta ejecutar comando SU:

Figura 38. Búsqueda validación binarios SU.



Fuente: El autor.

Recomendaciones. Verificar que el dispositivo en el que se va a ejecutar la aplicación no se encuentre rooteado, mediante las siguientes acciones:

- Validar la etiqueta test-keys en el archivo /system/build.prop del dispositivo.
- Validar la presencia de certificados OTA.
- Verificar la existencia de aplicaciones que solo se ejecutan en dispositivos rooteados: com.noshufou.android.su, com.thirdparty.superuser, eu.chainfire.supersu, com.koushikdutta.superuser.
- Verificar la presencia de binarios SU e Intentar ejecutar comandos SU.

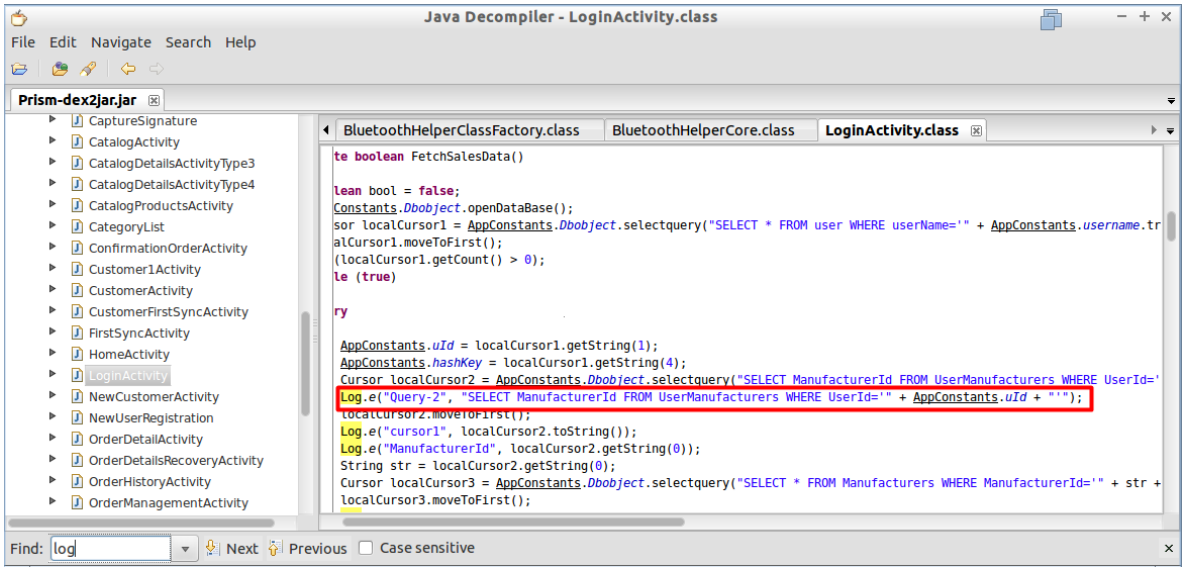
En caso en el que alguna de las pruebas sea positiva, interrumpir la ejecución de la aplicación.

➢ **MSTG-05. Almacenamiento de datos sensibles en archivos de Logs.** Dentro de la aplicación se encontraron muchos casos en los que se almacenan datos sensibles en el log sin ser encriptados. Los más importantes se detectaron en el código de la clase correspondiente al Activity de Login. En esta clase se pudo observar que se almacena en el archivo de log las URL´s de autenticación tanto de Sales Rep (vendedores) como de Customers (clientes), y algunas consultas ejecutados en la base de datos, en las que incluso se utilizan credenciales de usuarios. Esto se evidencia en las siguientes figuras:

Figura 39. Almacenamiento de url Login Customer en el log.

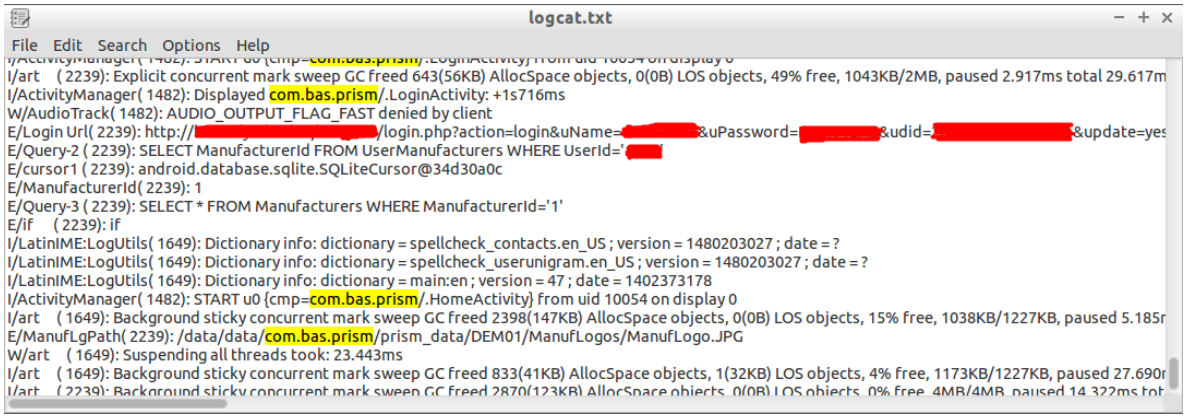
Fuente: El autor.

Figura 40. Almacenamiento del query login para autenticación local de Vendedores.



Fuente: El autor.

Figura 41. Datos sensibles en el log del dispositivo.



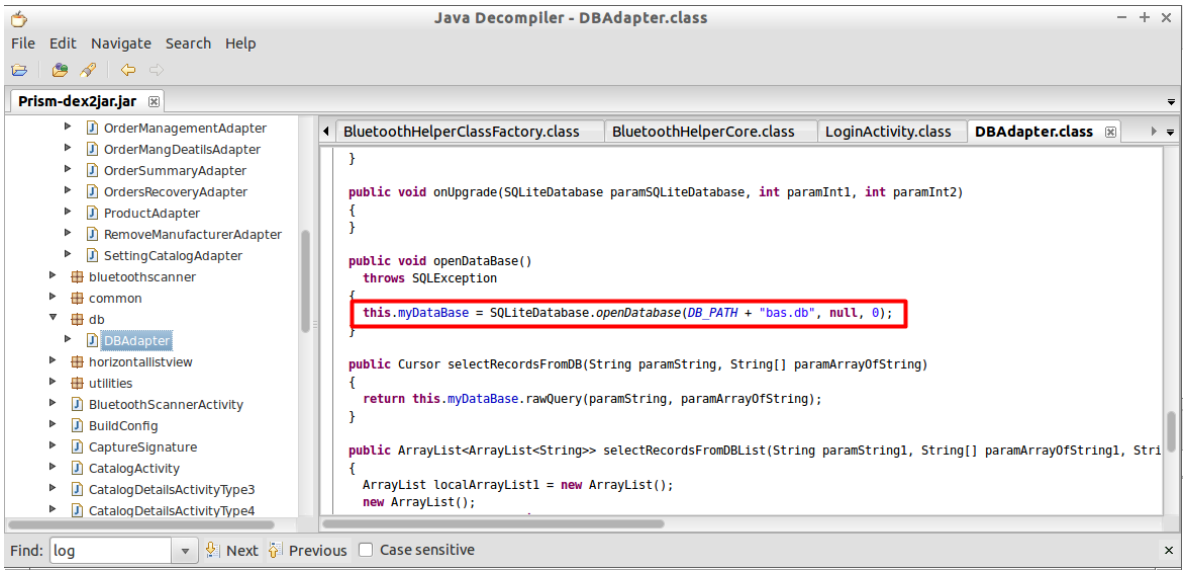
Fuente: El autor.

Recomendaciones. Eliminar del código de la aplicación toda sentencia en la que se almacene información sensible en el log del dispositivo. Si se requiere hacer seguimiento del comportamiento de la aplicación, entonces hacerlo solo para una versión de pruebas.

- **MSTG-06. Credenciales quemadas en el código (Hard coded).** No se identifican datos de credenciales, tokens, o llaves de encriptación quemadas en el código.

- **MSTG-07. Credenciales son almacenadas en la base de datos sin ser encriptadas.** La aplicación hace uso de una base de datos con nombre bas.db, como se muestra en la siguiente figura:

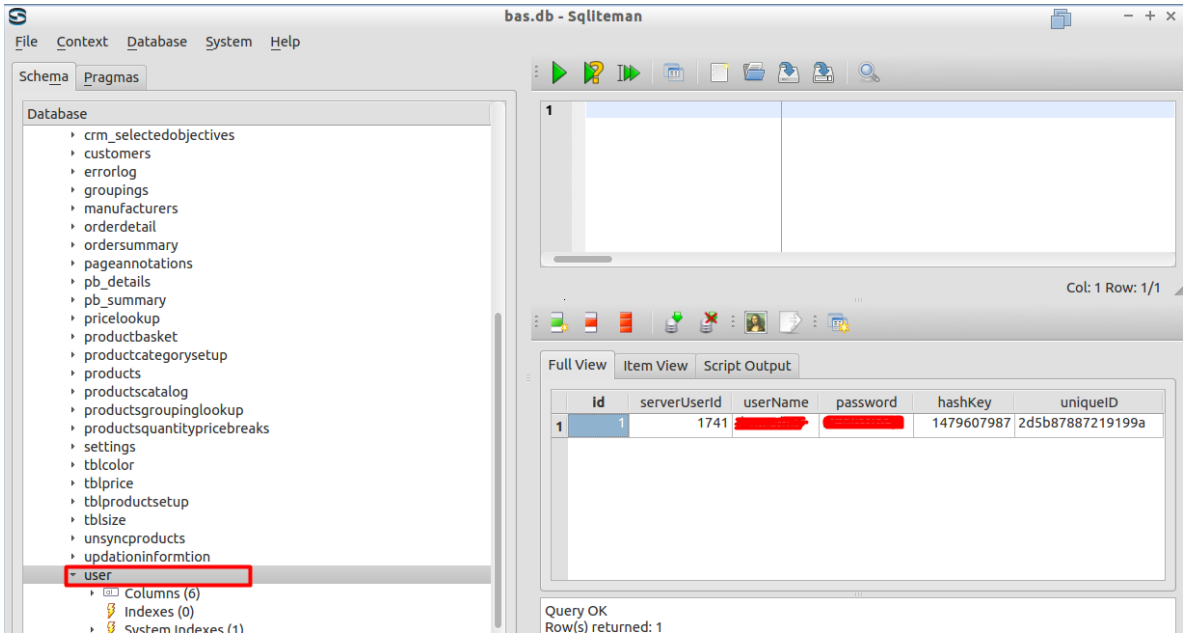
Figura 42. Ubicación archivo de Base de datos.



Fuente: El autor.

Al realizar una revisión de las tablas de la base de datos y la información que en estas se almacenan, se encontró información sensible sin ser encriptada. En la tabla “user” se almacenan registros de usuarios correspondientes a vendedores, si se analiza un registro de esta tabla, y se toman los valores de los campos userName y password, es posible autenticarse en la aplicación sin ningún inconveniente.

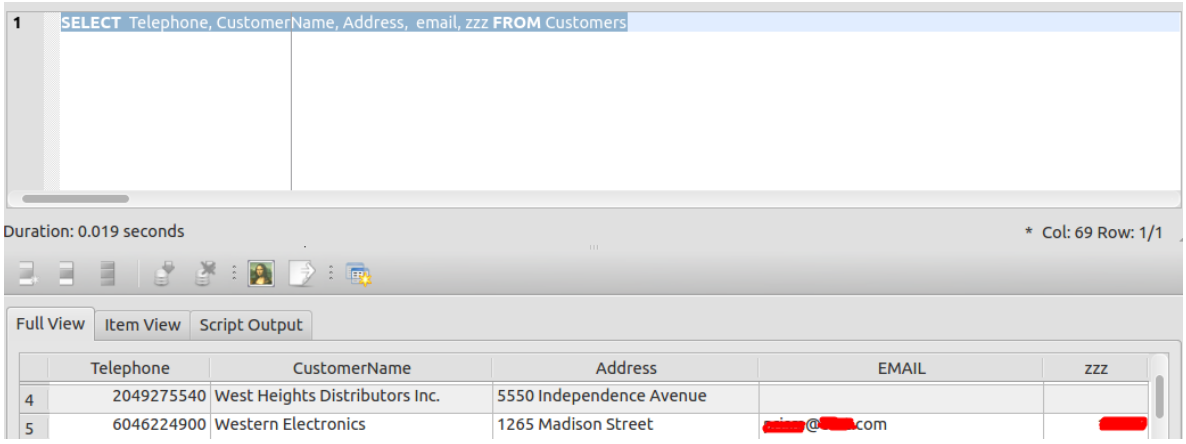
Figura 43. Ubicación de información sensible de vendedores en la base de datos.



Fuente: El autor.

En la tabla Customers se almacena información sensible de los clientes, como nombre, teléfono, dirección, el email, y su contraseña:

Figura 44. Ubicación de información sensible de clientes en la base de datos.



Fuente: El autor.

Como se muestra en la siguiente figura, los campos email y zzz son los utilizados para realizar la autenticación de los clientes dentro de la aplicación:

Figura 45. Identificación consulta autenticación de clientes.

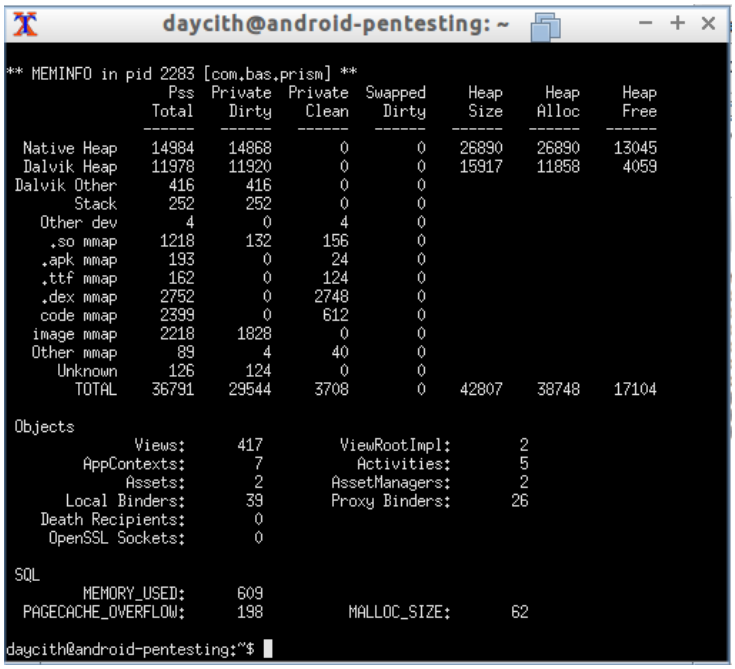
Fuente: El autor.

Recomendaciones. Si es estrictamente necesario el almacenamiento de datos sensibles en una base de datos local, se deben implementar mecanismos de encriptación, utilizando las alternativas proporcionadas por el mismo lenguaje de programación JAVA, o mediante el uso de librerías externas como:

- wxSQLite
- SQLiteCrypt
- botansqlite3
- SQLCipher

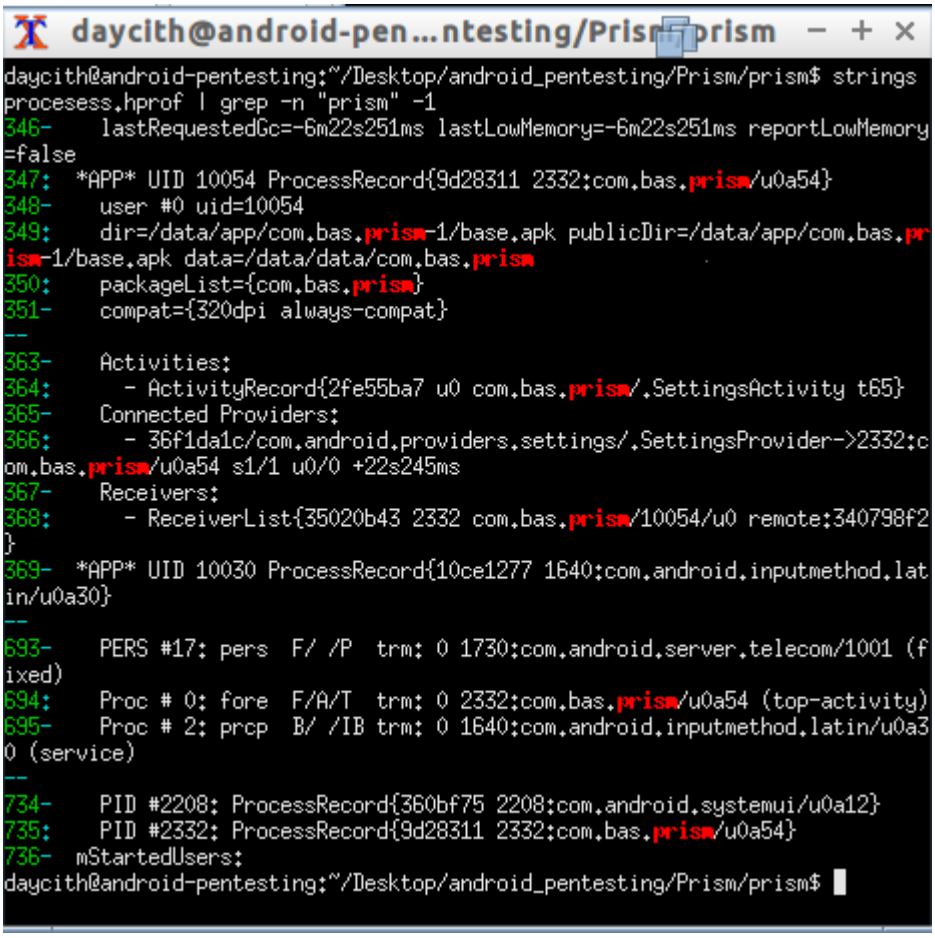
- **MSTG-08. Acceso a Información sensible después de realizar un volcado de memoria.** Se realizaron comprobaciones de la memoria del dispositivo y no se identifica el almacenamiento de información sensible una abierta o cerrada la aplicación:

Figura 46. Información de la memoria.



Fuente: El autor.

Figura 47. Búsqueda Información sensible en la memoria.



Fuente: El autor.

- **MSTG-09. Almacenamiento de datos sensibles fuera del sandbox de la aplicación.** Durante el análisis estático se detecta que dentro de la funcionalidad que permite enviar una solicitud de soporte técnico, se adjunta la base de datos de la aplicación. Para esto se crea un archivo ZIP que es ubicado dentro de la carpeta de descargas, que a su vez se ubica dentro de la memoria externa del dispositivo:

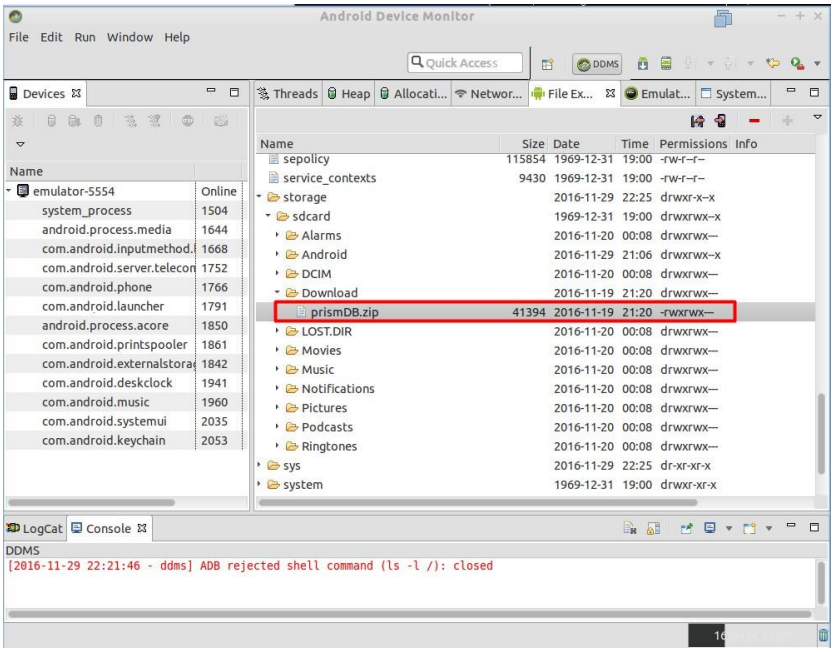
Figura 48. Código copia archivo base de datos a carpeta de descargas.

```
        }
    });

    btn_supportEmail.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {

            String db_path = "/data/data/com.bas.prism/databases/bas.db";
            String destzip = Environment.getExternalStoragePublicDirectory(
                    Environment.DIRECTORY_DOWNLOADS).getAbsolutePath()
                    + "/prismDB.zip";
            Log.e("destzip", destzip);
```

Fuente: El autor.

Durante una revisión dinámica de la aplicación, se ejecuta la funcionalidad de Soporte técnico, encontrando efectivamente el archivo ZIP con la base de datos:

Figura 49. Archivo ZIP con base de datos dentro de carpeta de descargas.



Fuente: El autor.

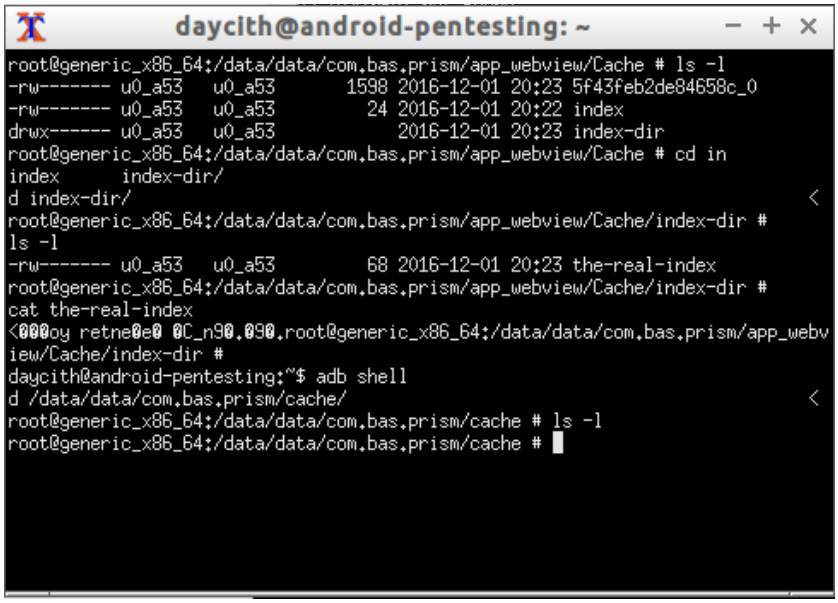
Lo que sin duda representa un problema de seguridad, ya que un atacante podría acceder a la base de datos directamente durante una navegación en el dispositivo, a través de una aplicación maliciosa, o conectando el dispositivo a un equipo de cómputo.

Recomendaciones. Si la lógica del negocio requiere el envío de la base de datos al personal de soporte técnico, se recomienda que ésta sea eliminada de la carpeta de descargas inmediatamente después de ser enviada.

Al tratarse de un archivo de base de datos, se recomienda que la información sensible que se almacena en ella sea encriptada, utilizando mecanismos de encriptación proporcionados por el mismo lenguaje de programación JAVA, o mediante el uso de librerías externas, las cuales fueron listadas anteriormente.

- **MSTG-10. Almacenamiento de datos sensibles en caché.** Después de una extensa interacción con la aplicación, se procedió a revisar el contenido de la carpeta caché, sin encontrar algún tipo de información sensible:

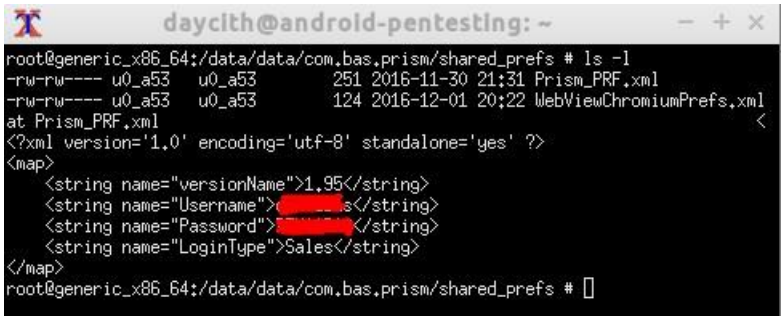
Figura 50. Contenido de la caché.



Fuente: El autor.

- **MSTG-11. Almacenamiento de datos sensibles en preferencias.** Al revisar el archivo de preferencias, se encontró que la aplicación guarda datos de usuarios y contraseñas sin ser encriptados.

Figura 51. Datos sensibles en Preferencias.



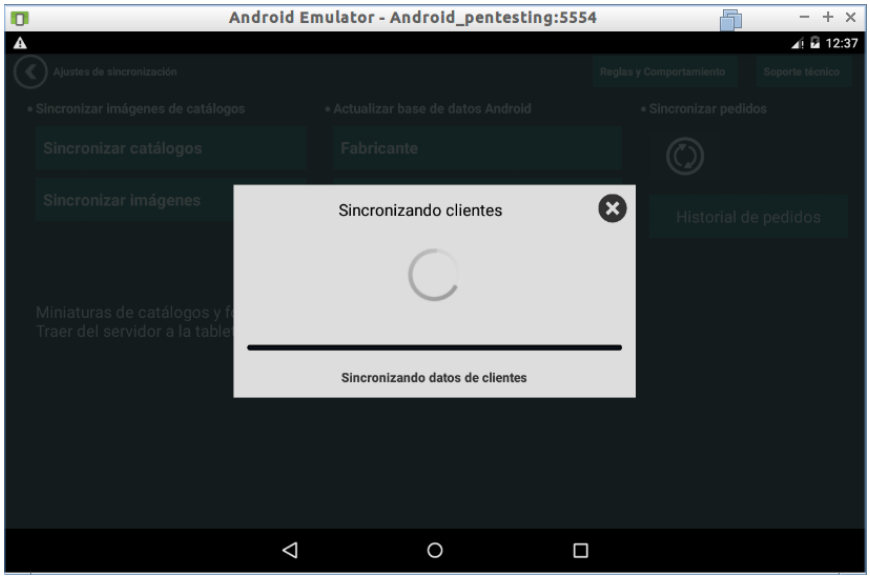
Fuente: El autor.

Recomendaciones. Si es estrictamente necesario la implementación de una funcionalidad para recordar las credenciales del usuario, se deben adoptar mecanismos de encriptación, utilizando las alternativas proporcionadas por el mismo lenguaje de programación JAVA, o mediante el uso de librerías externas como Secure-preferences, disponible en https://github.com/scottyab/secure-preferences.

Adicionalmente, se recomienda establecer el valor de la propiedad MODE_WORLD_READABLE a false, lo que impedirá que otras aplicaciones puedan acceder a las preferencias de la aplicación.

- **MSTG-12. Información sensible es enviada como texto legible a través de la red.**

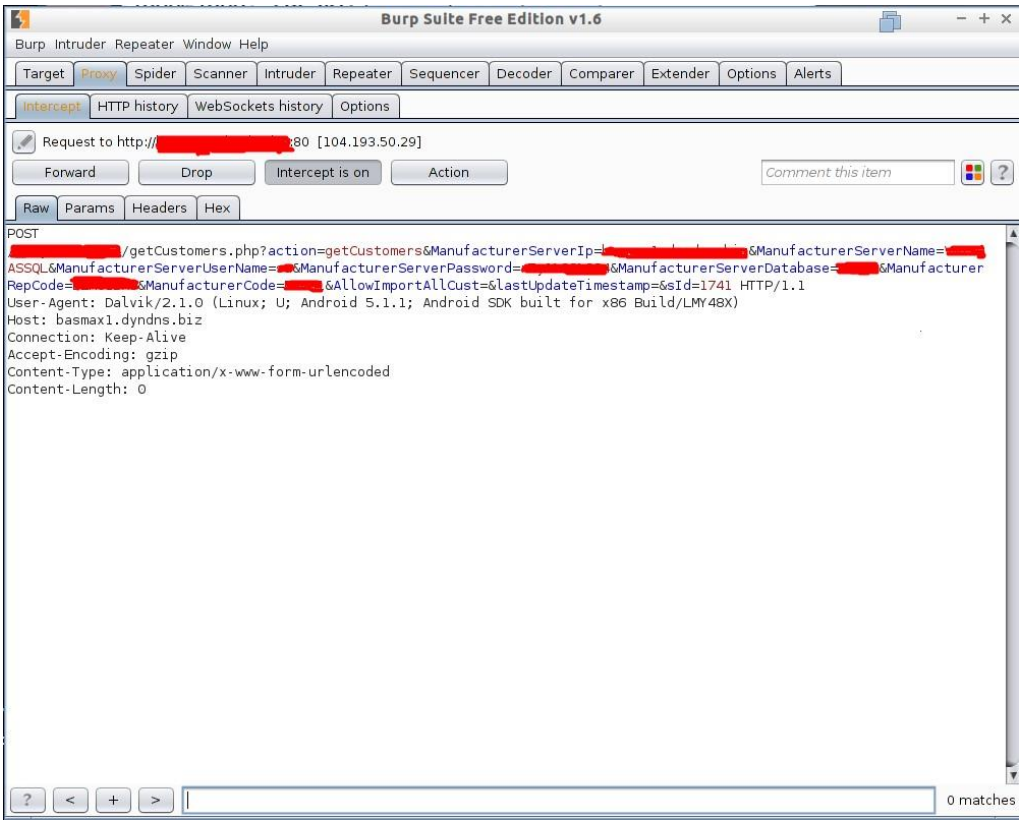
Figura 52. Proceso de sincronización de clientes.



Fuente: El autor.

Se analiza el tráfico de red de la aplicación y se determina que si se transmite información sensible por el protocolo HTTP en vez de HTTPS.

Figura 53. Transmisión de datos sensibles de Manufacturer y base de datos remota por HTTP.

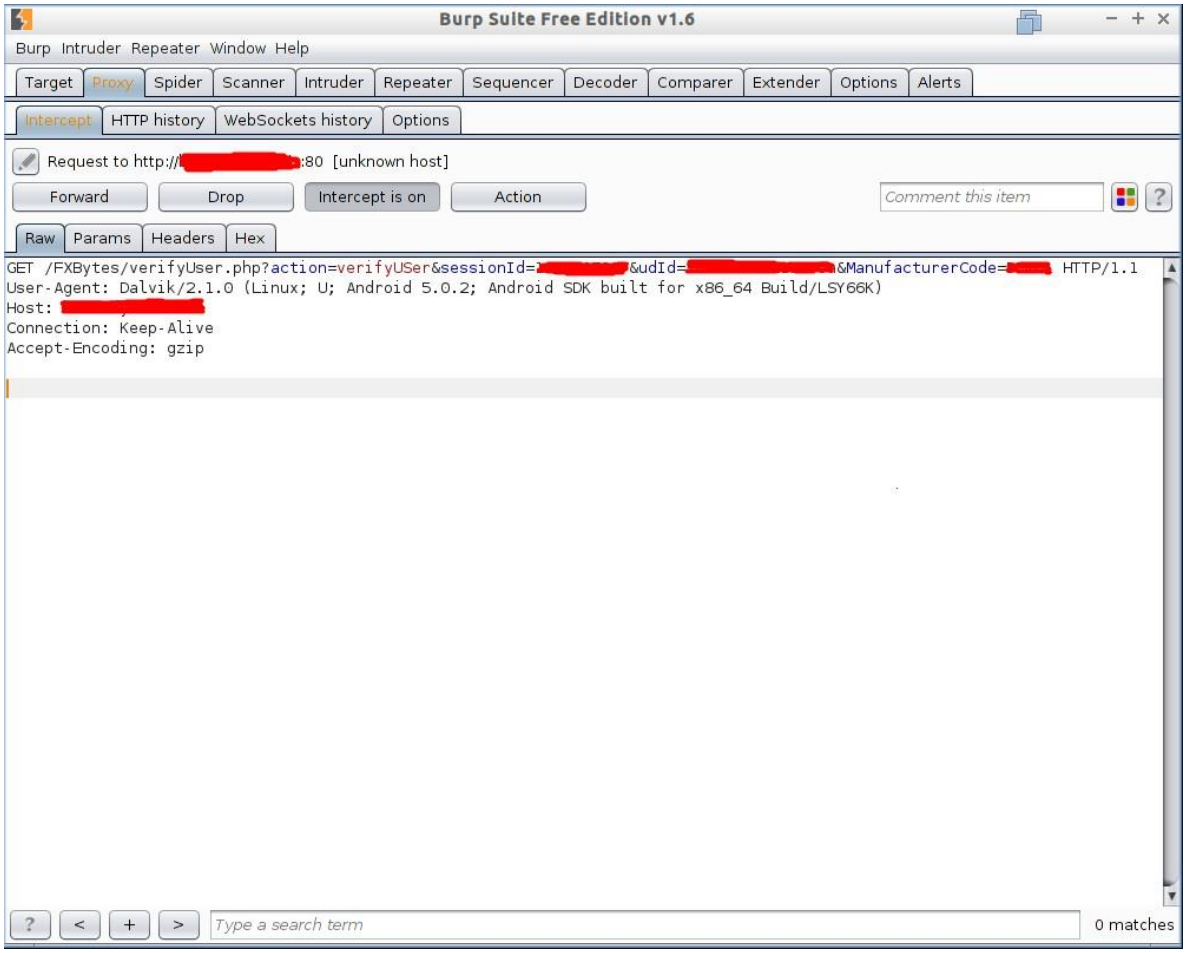


Fuente: El autor.

Dentro de la información sensible se encuentran datos del Manufacturer y de la base de datos remota. Esto pasa por lo menos en un 90% de los llamados a los Web Services.

Se notó también, que antes de realizar procesos de comunicación desde y hacia el servidor, se realiza una verificación del usuario, mediante un llamado a un Web Service específico, el problema es que en el proceso de verificación también se envían datos de la sesión y el identificador del dispositivo del usuario por el protocolo HTTP, como se muestra en la figura siguiente:

Figura 54. Transmisión de datos sensibles de usuario y dispositivo por HTTP.



Fuente: El autor.

Recomendaciones. Asumir que la capa de transporte es susceptible a una variedad de tipos de interceptación y manipulación, por lo que en lo posible, se debe eliminar el envío de los parámetros de conexión a la base de datos alojada en el servidor, esta información debería ser obtenida en los Web Services.

Dejar de transmitir cualquier tipo de información sensible por el protocolo HTTP, y aplicar SSL o TLS para realizar peticiones al servidor.

Validar que efectivamente todas las conexiones se hacen a través de un canal seguro, validar la identidad del servidor, y que su certificado sea expedido por una entidad certificadora reconocida y autorizada. Nunca se deben aceptar certificados auto-firmados.

Para el caso específico de Android, se deben asegurar que no existan instrucciones como org.apache.http.conn.ssl.AllowAllHostnameVerifier y SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER en el código de la aplicación.

Tener en cuenta que incluso las librerías para la encriptación SSL pueden presentar vulnerabilidades, por lo que sí es posible se debe implementar una capa de encriptación adicional antes de transferir información sensible por el canal SSL.

- **MSTG-13. Datos sensibles son enviados como un parámetro del tipo querystring.** En la figura 54 se evidencia también que se realizan peticiones GET, enviando en la URL parámetros con datos sensibles, como es la sesión del usuario e identificador del dispositivo.

Recomendaciones. Reducir la cantidad de parámetros enviados en las peticiones GET, y si esto no es posible, entonces enviarlos a través de peticiones por un canal seguro.

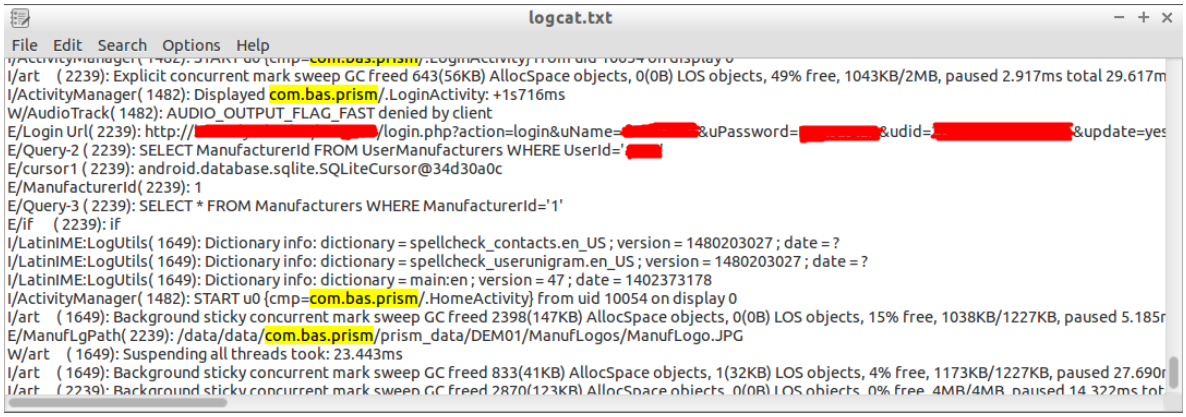
Además, se debe validar cada uno de los parámetros que son enviados desde la aplicación y recibidos en los Web Services. Un buen proyecto para este propósito en cualquier proyecto en Java, y por ende en Android, es el The Stinger HTTP request validation engine, el cual se encuentra disponible en http://sourceforge.net/projects/stinger.

- **MSTG-14. Protocolos de red inseguros.** Como se evidencia en las pruebas anteriores, todas las comunicaciones con el servidor se realizan a través del puerto HTTP, del cual sobra decir que es un protocolo de red inseguro.

Recomendaciones. Las mismas recomendaciones dadas en las pruebas MSTG-12 y MSTG-13.

- **MSTG-15. Exposición de datos en el log.** Dentro de la aplicación se encontraron muchos casos en los que se exponen datos sensibles en el log sin ser encriptados. Los más importantes se detectaron en el código de la clase correspondiente al Activity de Login. En esta clase se pudo observar que se imprime en el archivo de log las URL´s de autenticación tanto de Sales Rep (vendedores) como de Customers (clientes), y algunas consultas ejecutados en la base de datos, en las que incluso se utilizan credenciales de usuarios. Esto se evidencia en las siguientes figuras:

Figura 55. Exposición de datos sensibles en el archivo log del dispositivo.

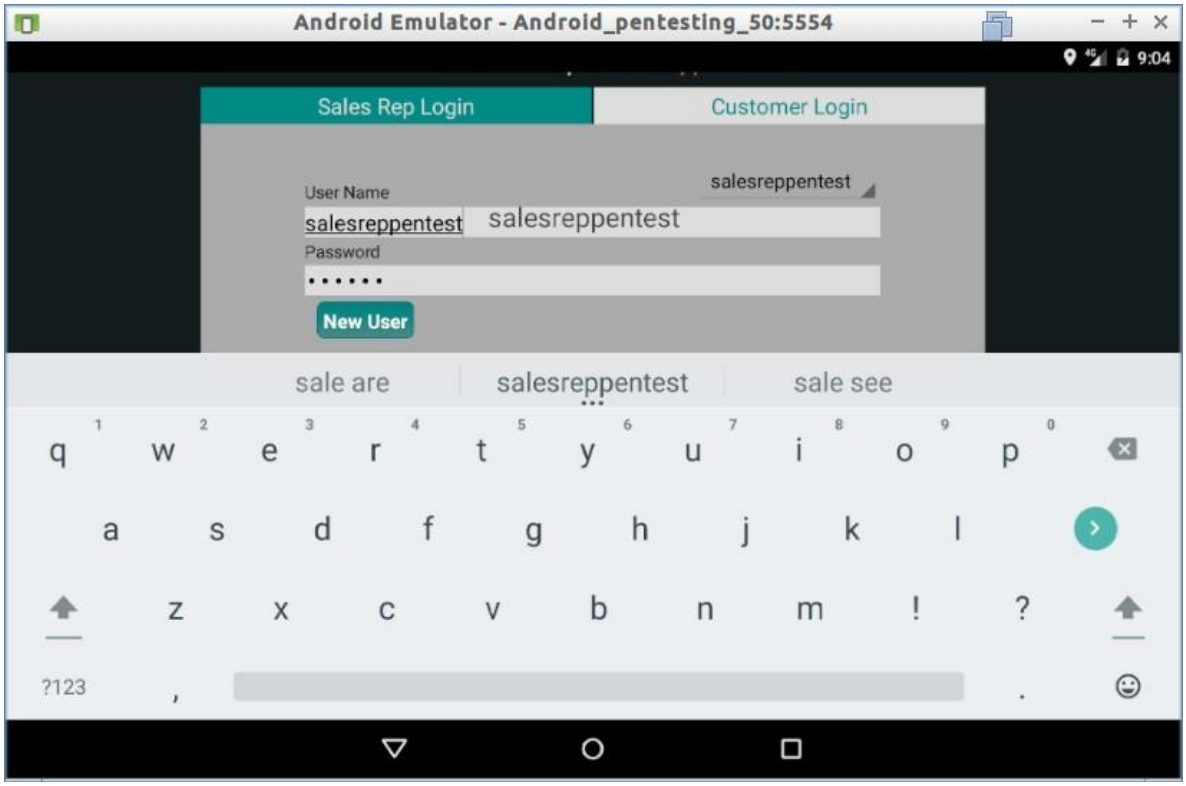


Fuente: El autor.

Recomendaciones. Eliminar del código de la aplicación toda sentencia en la que se almacene información sensible en el log del dispositivo. Si se requiere hacer seguimiento del comportamiento de la aplicación, entonces hacerlo solo para una versión de pruebas.

- **MSTG-16. Portapapeles (clipboard) no es deshabilitado.** Durante las pruebas realizadas se evidenció que en los campos de texto destinados para nombres de usuarios no se encuentra desactivada la opción de copiar y pegar:

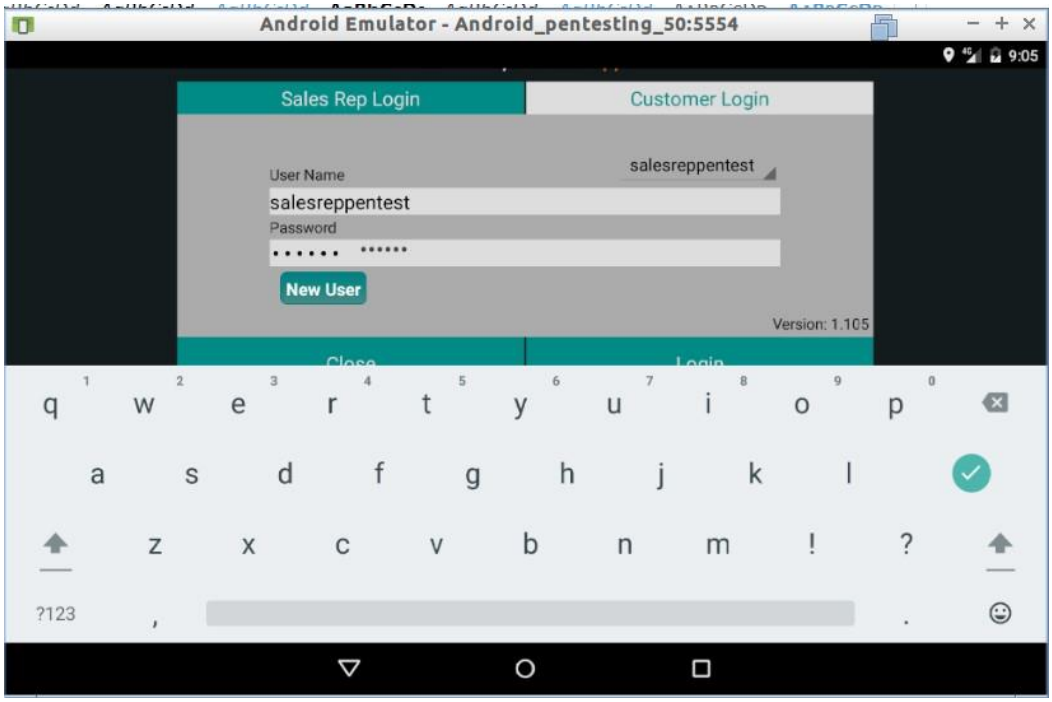
Figura 56. Portapapeles no deshabilitado en campo de nombre de usuario.



Fuente: El autor.

Para los campos de contraseñas se evidencia también que es posible pegar datos del portapapeles.

Figura 57. Portapapeles no deshabilitado en campo de contraseña.



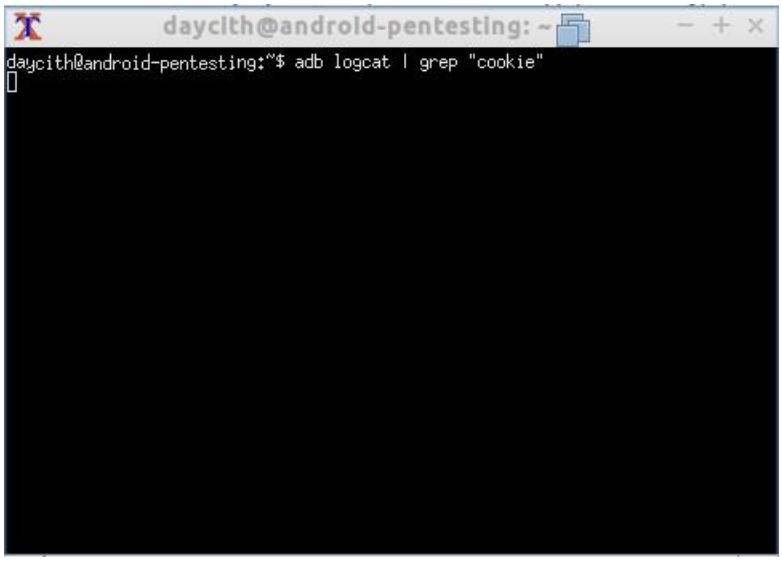
Fuente: El autor.

Recomendaciones. La recomendación obvia es deshabilitar el uso del portapapeles en los campos donde se debe ingresar información sensible, como nombres de usuarios y contraseñas.

- **MSTG-17. Application Backgrounding.** Durante la revisión de los archivos del dispositivo, en este caso del emulador, no se encontraron capturas de pantalla de la aplicación.

- **MSTG-18. "Url Caching" (peticiones y respuestas).** En el log del dispositivo no se evidencia el almacenamiento de cookies:

Figura 58. Busque de información de peticiones y respuesta en caché.



Fuente: El autor.

Y en el fragmento de código marcado en la siguiente figura, se evidencia la limpieza de la caché al cargar URL´s en los WebViews de la aplicación:

Figura 59. Limpieza de la caché en WebView.

```
progressBar = (ProgressBar) findViewById(R.id.progressBar1);

webView1.setWebChromeClient(new WebChromeClient() {
    @Override
    public void onProgressChanged(WebView view, int progress) {
        progressBar.setProgress(0);
        progressBar.setVisibility(View.VISIBLE);

        OrderHistoryActivity.this.setProgress(progress * 1000);

        progressBar.incrementProgressBy(progress);
        webView1.setVisibility(View.GONE);
        if (progress == 100) {
            webView1.setVisibility(View.VISIBLE);
            progressBar.setVisibility(View.GONE);

            webView1.clearCache(true);
            deleteDatabase("webview.db");
            deleteDatabase("webviewCache.db");
        }
    }
});
webView1.clearCache(true);
```

Fuente: El autor.

Lo que indica que la aplicación no almacena peticiones y respuestas en la caché.

- **MSTG-19. Permisos de aplicación inseguros.**

Figura 60. Permisos solicitados por la aplicación.

Fuente: El autor.

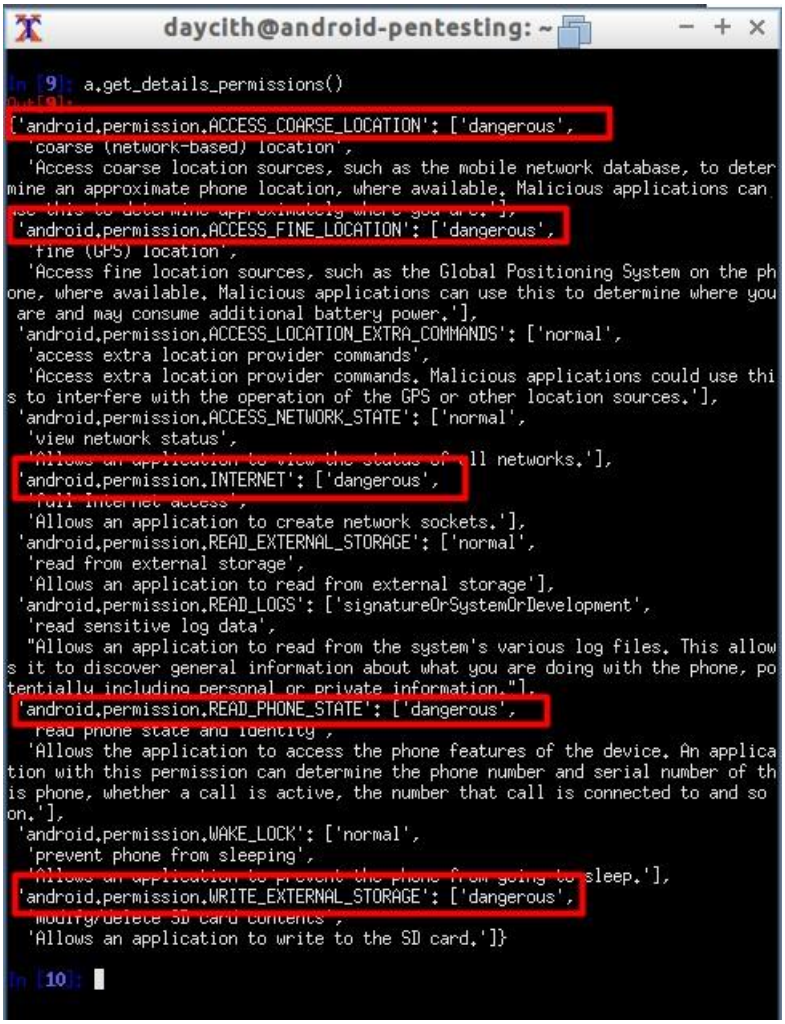
Como se muestra en la figura anterior, los permisos requeridos por la aplicación son los siguientes:

- INTERNET
- READ_EXTERNAL_STORAGE
- WRITE_EXTERNAL_STORAGE
- ACCESS_NETWORK_STATE
- WAKE_LOCK,
- ACCESS_COARSE_LOCATION
- ACCESS_FINE_LOCATION
- ACCESS_LOCATION_EXTRA_COMMANDS
- READ_LOGS

- READ_PHONE_STATE

De los cuales, son considerados como peligrosos los siguientes:

Figura 61. Detalles de los permisos solicitados por la aplicación.



Fuente: El autor.

ACCESS_COARSE_LOCATION: Una aplicación maliciosa podría usar este permiso para determinar la ubicación actual del usuario.

ACCESS_FINE_LOCATION: al igual que el permiso anterior, permite conocer la ubicación actual del usuario, pero adicionalmente puede causar un aumento del consumo de batería mayor.

READ_PHONE_STATE: Una aplicación con este permiso puede determinar el número de teléfono y su serial, e incluso el número telefónico con el que se tiene una llamada activa.

WRITE_EXTERNAL_STORAGE: usado para escribir en la memoria externa del dispositivo.

De acuerdo a la revisión hecha, los permisos ACCESS_COARSE_LOCATION y ACCESS_FINE_LOCATION se requieren para capturar la ubicación desde la que se realiza una orden.

Por su parte, el permiso WRITE_EXTERNAL_STORAGE es requerido en varias funcionalidades en la que es necesario interactuar con la memoria externa del dispositivo, la memoria SD. Sin embargo, como se vio en la prueba MSTG-09, en la funcionalidad que permite solicitar soporte técnico al personal de la empresa, se copia la base de datos de la aplicación a la tarjeta SD y se deja ahí, sin ser eliminada. Esto implica un problema de seguridad, ya que otras aplicaciones pueden acceder a la memoria externa, incluso una persona puede acceder a ella a través del explorador de archivos del dispositivo.

El permiso READ_PHONE_STATE, implica un problema de seguridad en la aplicación. Un atacante podría aprovechar que la aplicación no implementa una adecuada protección de su código y que tiene concedido este permiso, para para incluir fragmentos de código malicioso, en los que se acceda al número telefónico o a las llamadas realizadas en el dispositivo móvil.

A parte del hecho anterior, se encuentra un problema mayor e innecesario. En la prueba MSTG-26 se mostrará también que la aplicación utiliza uno de los identificadores del dispositivo para llevar un control de las instalaciones de la aplicación para un mismo usuario, y para esto requiere del permiso

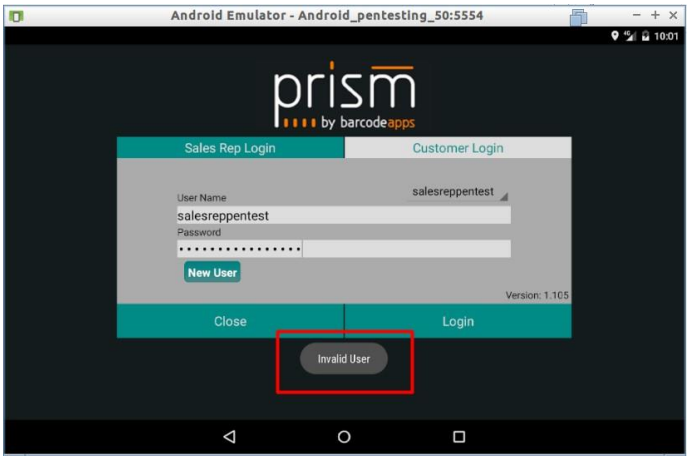
READ_PHONE_STATE. Si la aplicación utiliza un identificador de instancia en vez del identificador del dispositivo, no tendría que solicitar este permiso.

Recomendaciones. Generalmente, los desarrolladores de una aplicación deben esforzarse por definir la menor cantidad de permisos posible sin dejar de cumplir con los requisitos del negocio y de seguridad. En este sentido, se debe dejar de utilizar el identificador del dispositivo físico y utilizar un identificador de instancia de la aplicación o en su defecto un GUID; después de esto, se debe dejar de solicitar el permiso READ_PHONE_STATE.

Lo ideal es que la aplicación no tenga que almacenar información en la memoria externa del dispositivo y que el permiso WRITE_EXTERNAL_STORAGE no sea solicitado, pero en caso que sea estrictamente necesario, se deben adoptar mecanismos de encriptación de los archivos, por lo menos en los que contengan algún tipo de información sensible. En Android, es posible obtener un nivel de seguridad básico utilizando las clases del paquete javax.crypto.

- **MSTG-20. Bloqueo de cuenta no implementado.** Se llevaron a cabo varios intentos de autenticación con credenciales inválidas para un usuario vendedor, obteniendo como resultado un mensaje de “Invalid User”:

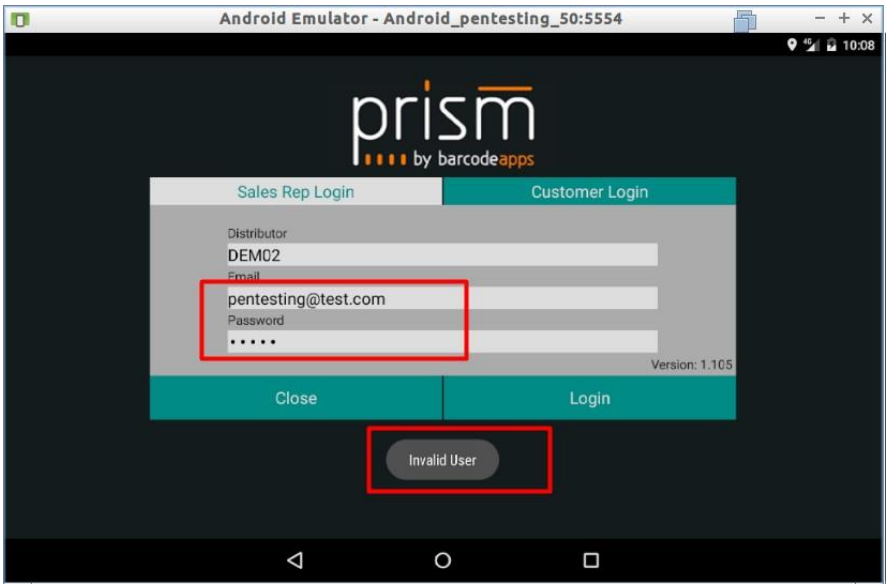
Figura 62. Bloqueo de cuenta no implementado para vendedores.



Fuente: El autor.

Lo mismo para un usuario cliente:

Figura 63. Bloqueo de cuenta no implementado en clientes.



Fuente: El autor.

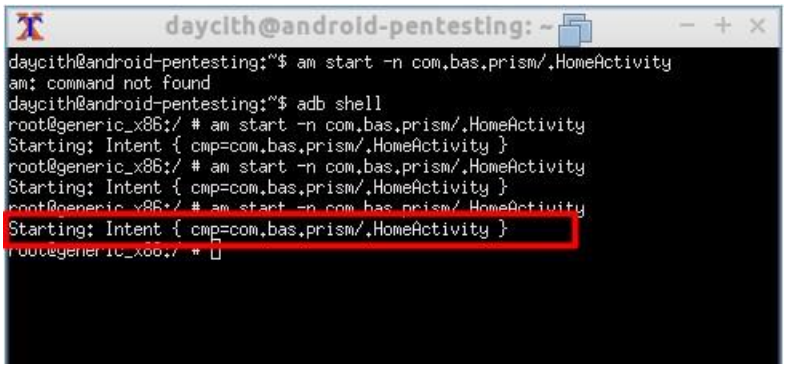
Pero luego de más de 10 intentos en ambos tipos de autenticación, la aplicación siguió mostrando el mensaje de “Invalid User” sin realizar un bloqueo de cuenta o desactivación de la autenticación por exceso de intentos fallidos.

Recomendaciones. La aplicación debe controlar el número de intentos de autenticación fallidos (máximo 10 intentos) y deshabilitar la cuenta del usuario por lo menos temporalmente.

Adicionalmente se debe cerrar la sesión o bloquear la interfaz de usuario después de cierto tiempo de inactividad (no más de 5 minutos).

- **MSTG-21. La autenticación se puede evadir.** Debido a que la aplicación es depurable, fue posible iniciar la actividad HomeActivity sin pasar por la autenticación:

Figura 64. Arrancar actividad, para evadir autenticación.

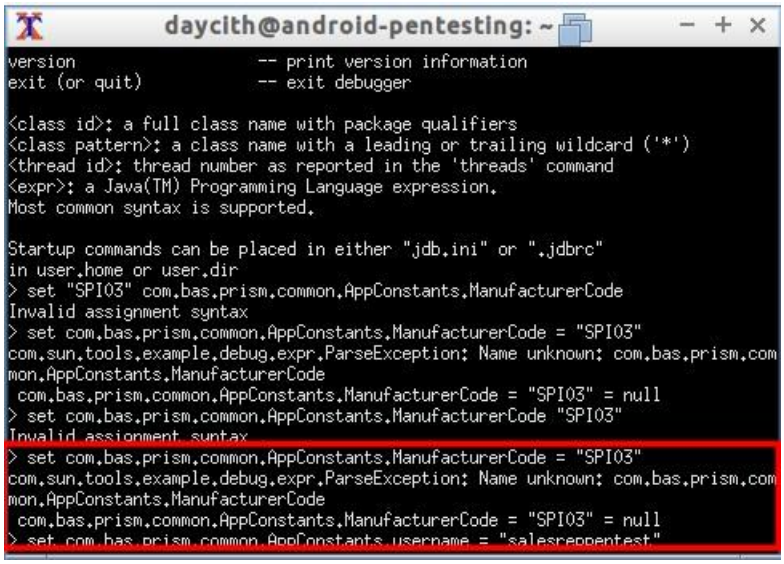


Fuente: El autor.

Inicialmente no se cargaron productos, clientes y órdenes, ya que esto depende del usuario autenticado. Frente a esta situación se optó por dos alternativas:

Establecer en tiempo de ejecución el valor del ManufacturerCode y el usuario vendedor:

Figura 65. Asignación de valores en tiempo de ejecución.

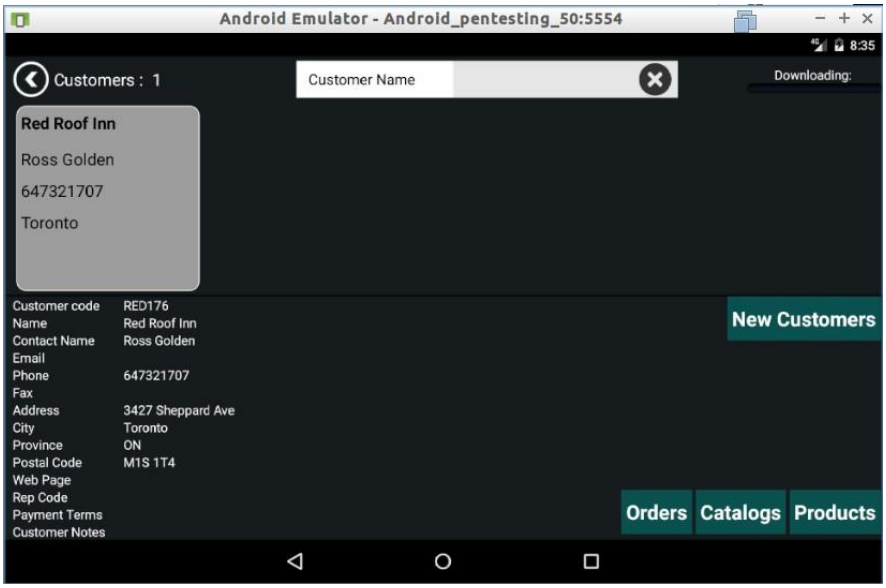


Fuente: El autor.

Iniciar sesión dentro de la aplicación con las credenciales correctas de un usuario vendedor, “Cerrar sesión” mediante el botón “Logout”, y luego iniciar una de las actividades de la aplicación.

En ambas situaciones fue posible arrancar la actividad HomeActivity y continuar la navegación por la sección de clientes:

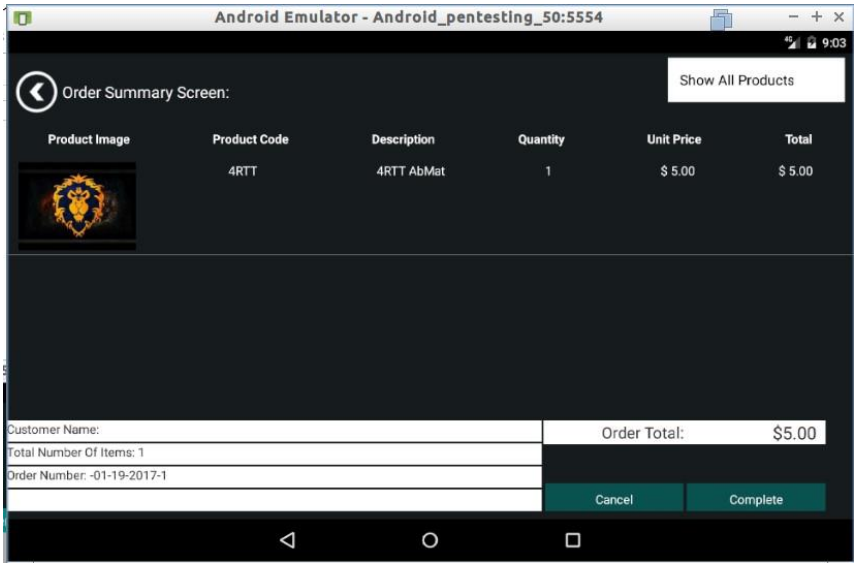
Figura 66. Selección de cliente, luego de evadir autenticación.



Fuente: El autor.

Agregar productos a una orden e incluso completarla:

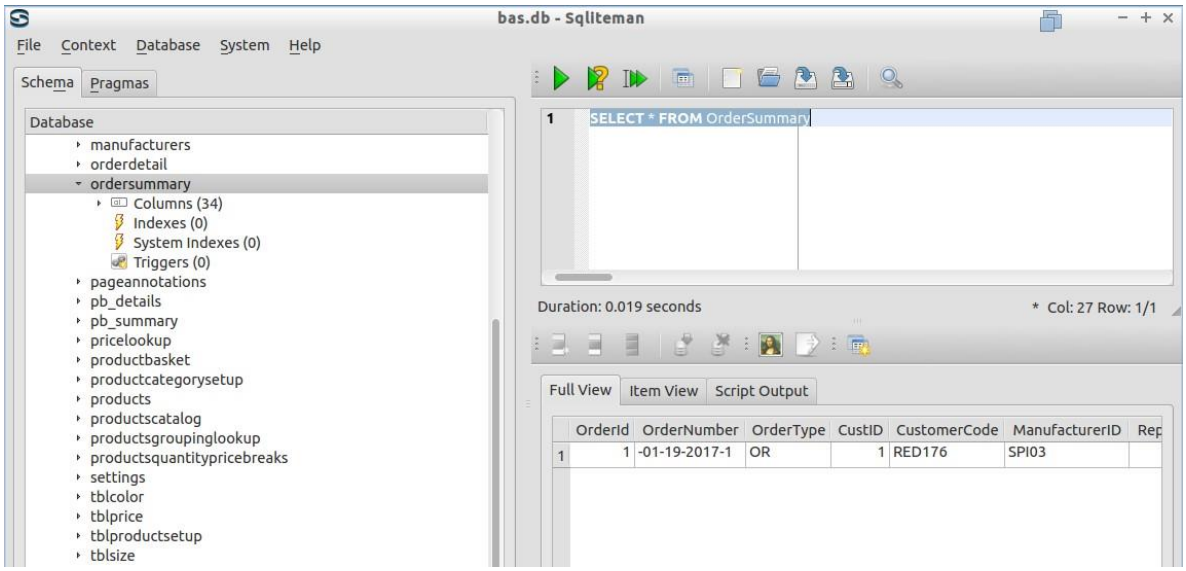
Figura 67. Completando orden, luego de evadir autenticación.



Fuente: El autor.

En la siguiente figura se evidencia la orden en la base de datos:

Figura 68. Orden en base de datos, luego de evadir autenticación.



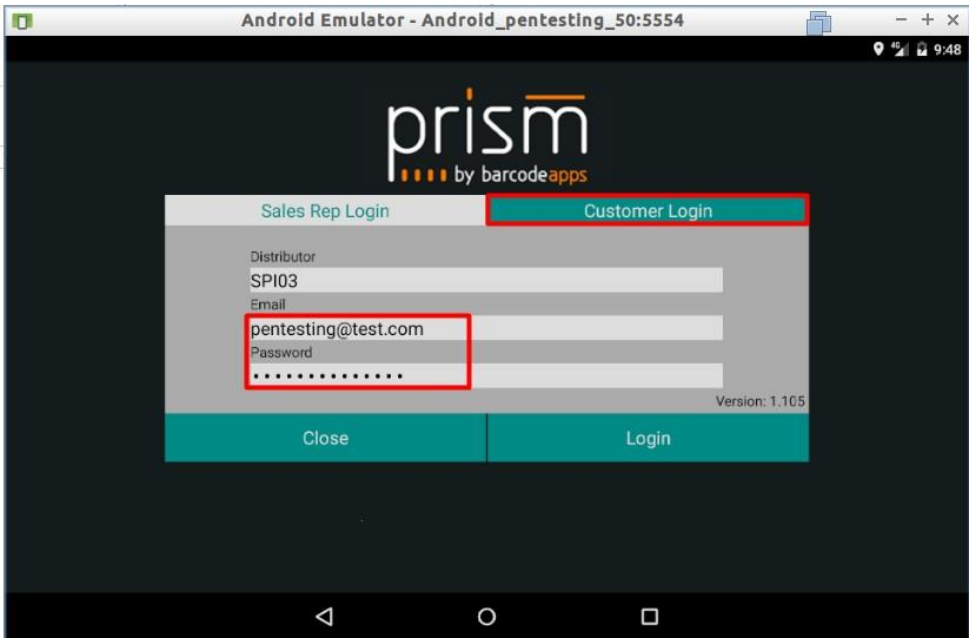
Fuente: El autor.

Recomendaciones. Los desarrolladores de la aplicación deben tener en cuenta que un atacante podría burlar el sistema de autenticación de la aplicación móvil, sobre todo si la aplicación es depurable, como en este caso. La recomendación es que en cada Actividad, o por lo menos en las que tengan un acceso restringido, se verifique la identidad del usuario, si está autorizado para ingresar a dicha, y a ejecutar las acciones que dentro de ella se pueden llevar a cabo.

Por el lado del servidor se debe verificar también la identidad del usuario que realiza las peticiones y si está autorizado para llevar a cabo la acción solicitada.

- **MSTG-22. Escalamiento de privilegios.** Mediante un proceso similar a la prueba anterior, se pudo evidenciar que luego de ingresar a la aplicación como un cliente:

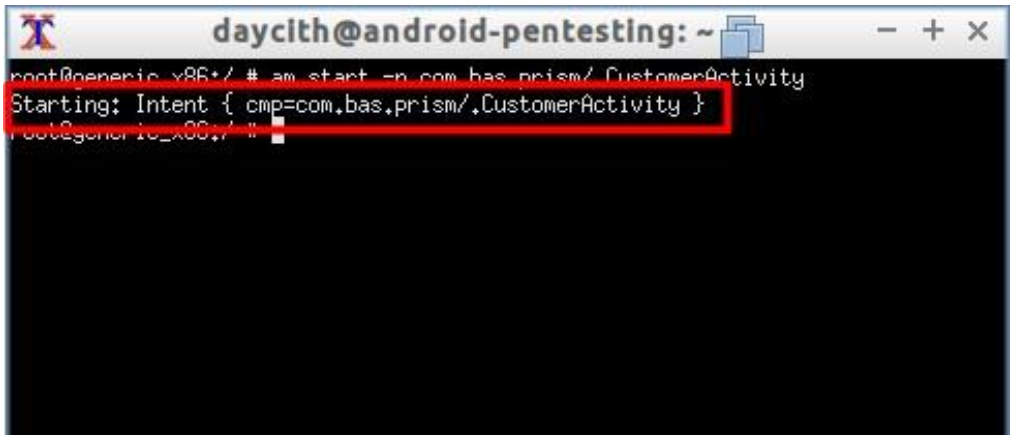
Figura 69. Autenticación como cliente para escalamiento de privilegios.



Fuente: El autor.

Y luego de esto, arrancar la actividad encargada de listar los clientes dentro de la aplicación, a la cual solo deberían tener acceso los usuarios vendedores:

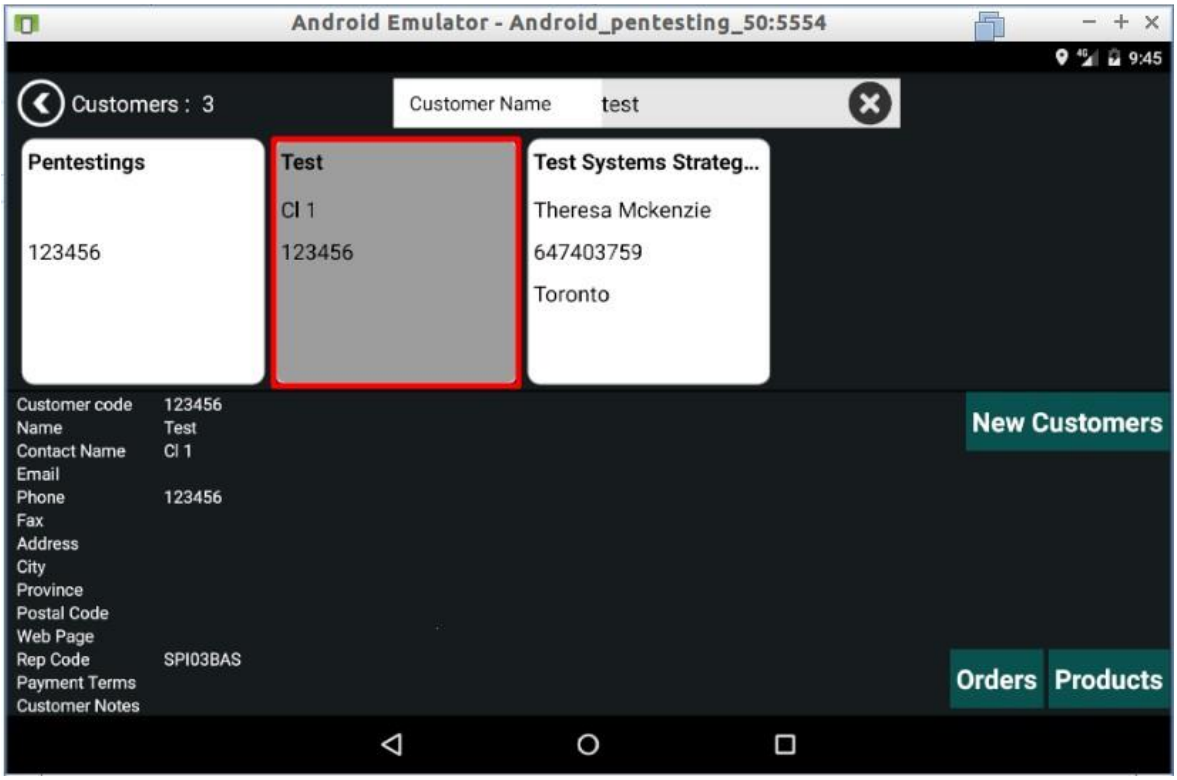
Figura 70. Arrancar actividad para escalar privilegios.



Fuente: El autor.

Perfectamente fue posible escoger un cliente diferente al “Penstesting” (el cliente con el que se ingresó a la aplicación):

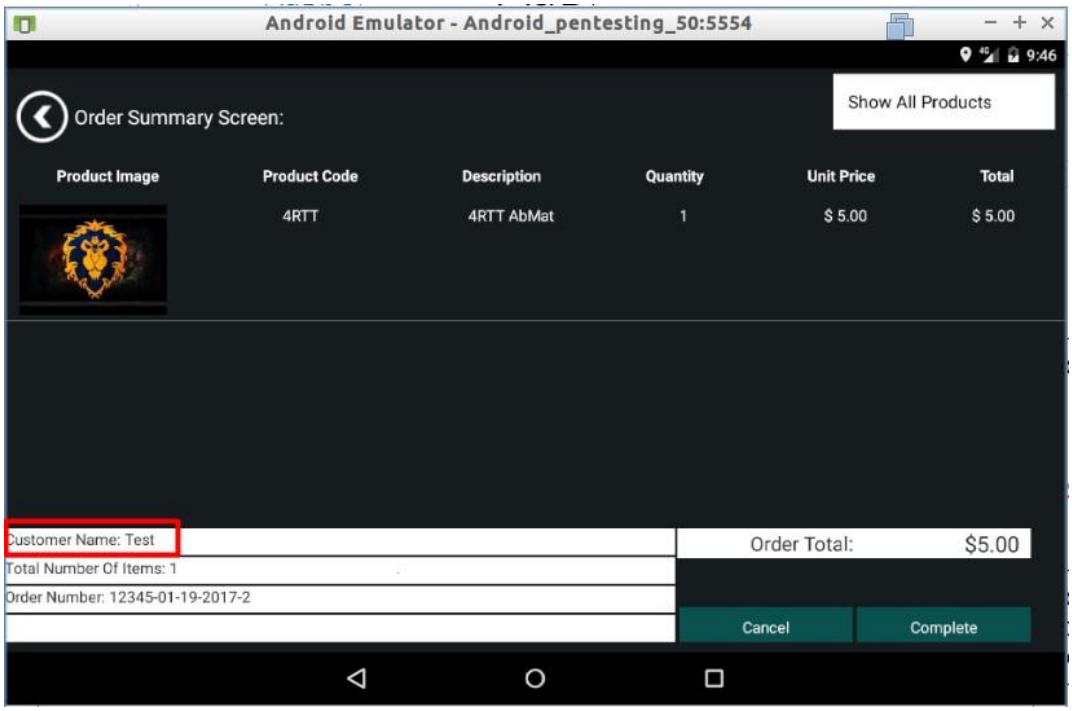
Figura 71. Selección de otro cliente autenticado como cliente.



Fuente: El autor.

Y de ahí en adelante navegar dentro de la aplicación y completar una orden a nombre del otro cliente:

Figura 72. Completando orden, luego de escalar privilegios.



Fuente: El autor.

Recomendaciones. Se debe evitar que la aplicación se pueda depurar, llevando a cabo las recomendaciones dadas en la prueba MSTG-02.

Al igual que para la prueba anterior, la recomendación es que en cada Actividad, o por lo menos en las que tengan un acceso restringido, se verifique la identidad del usuario, si está autorizado para ingresar a dicha actividad, y a ejecutar las acciones que dentro de ella se pueden llevar a cabo.

Al cerrar la aplicación o finalizar la sesión de un usuario, asegurarse de hacer una limpieza de los datos del último usuario logueado. Durante la ejecución de las pruebas se evidenció que luego de un usuario vendedor salía de la aplicación y posteriormente entraba un cliente, éste último podía seleccionar otro cliente y completar una orden a nombre del otro cliente seleccionado, si mediante un ataque binario se arrancaba la actividad CustomersActivity.

- **MSTG-23. No implementación o implementación no apropiada de una sección para cambiar contraseñas.** La aplicación no implementa un sistema para realizar cambio de contraseñas.

Recomendaciones. La aplicación debe implementar un sistema de bloqueo de cuentas luego de cierta cantidad de intentos de autenticación fallidos, y un sistema que le permita cambiar su contraseña tanto en el sistema de almacenamiento remoto como local.

- **MSTG-24. Políticas de contraseñas débiles.** Para los usuarios de pruebas creados:

Usuario vendedor:

Nombre de usuario: SalesRepPentest

Contraseña: SalesRepPentest

Usuario Cliente (Customer):

Email: pentesting@test.com

Contraseña: Pentesting2016

Se detectó que no se distingue entre mayúsculas y minúsculas en ninguno de los campos del formulario de Login. Se pudo ingresar exitosamente a la aplicación utilizando las contraseñas **salesrepentest** y **pentesting2016** para ingresar como vendedor y cliente respectivamente.

Recomendaciones. Además de un adecuado bloqueo de cuenta por cantidad de intentos fallidos y un sistema de cambio de contraseña, es necesario que la

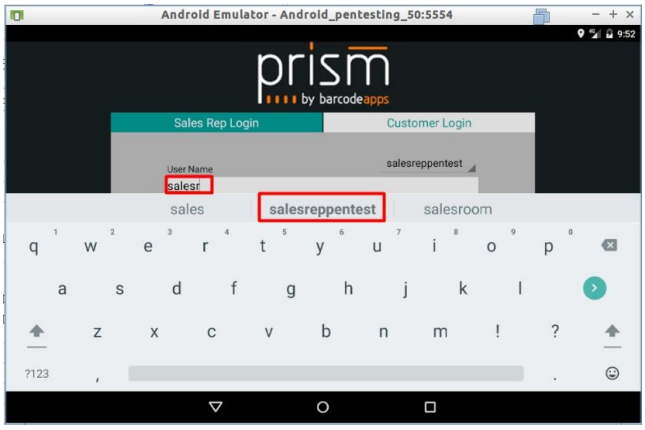
aplicación maneje fuertes políticas de contraseñas. Políticas de contraseñas fuertes implican, entre otros, los siguientes criterios:

- Distinción entre mayúsculas y minúsculas.
- Contraseñas deben ser lo suficiente largas, por ejemplo 6 caracteres mínimos.
- Deben contener además de números y letras, por lo menos un carácter especial.
- Imposibilidad para utilizar una contraseña ya utilizada por el usuario anteriormente.

Si lo anterior no es posible, se debería implementar un sistema de autenticación adicional.

- **MSTG-25. Opción de autocompletar no está deshabilitada.** Como se observa en la siguiente figura, si un usuario agrega alguna de sus credenciales al diccionario del dispositivo, le aparecerá como sugerencia posteriormente en los campos de la ventana de autenticación, lo que representa un problema de seguridad:

Figura 73. Opción de autocompletar no deshabilitado.

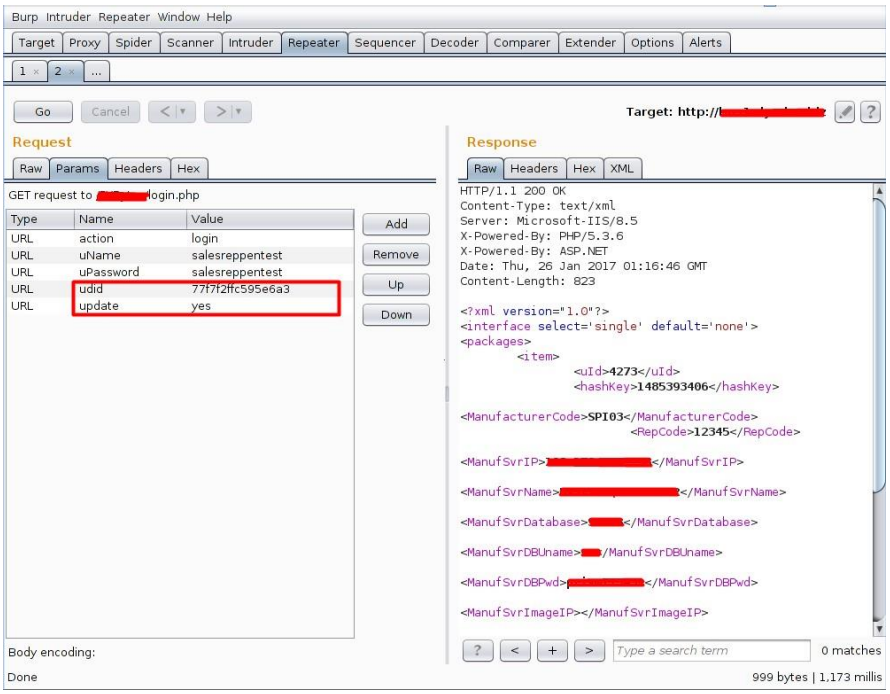


Fuente: El autor.

Recomendaciones. Deshabilitar las sugerencias en los campos en los que se debe ingresar datos sensibles, en este caso nombres de usuarios y contraseñas. Para el caso específico de Android, mediante el atributo InputType.

- **MSTG-26. Uso de IMEI/UDID falsos para la autenticación.** Cuando un usuario ingresa por primera vez a la aplicación, se realiza un proceso de autenticación contra un Web Service ubicado en el servidor. La petición realizada contiene, además del usuario y la contraseña, el identificador del dispositivo:

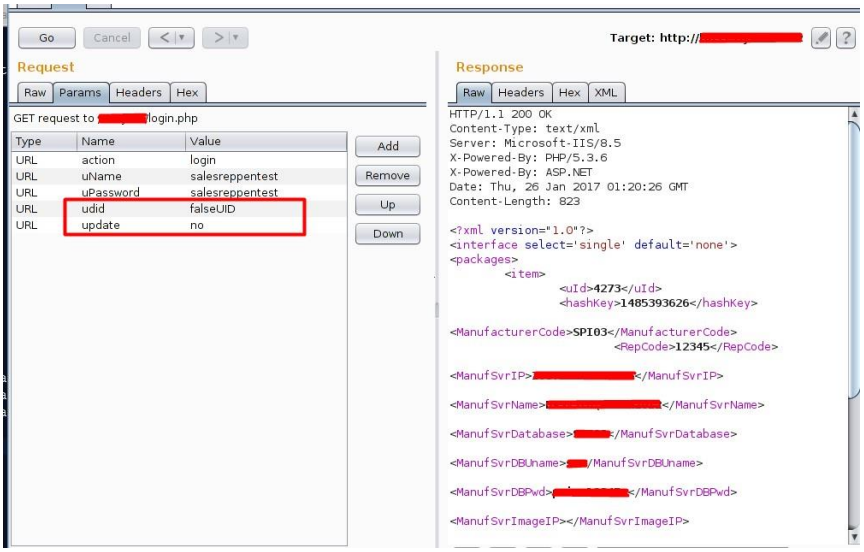
Figura 74. Envío de identificador de dispositivo para autenticación.



Fuente: El autor.

Si posteriormente se replica la petición al Web Service, con un valor de UDID falso se autentica sin ningún problema:

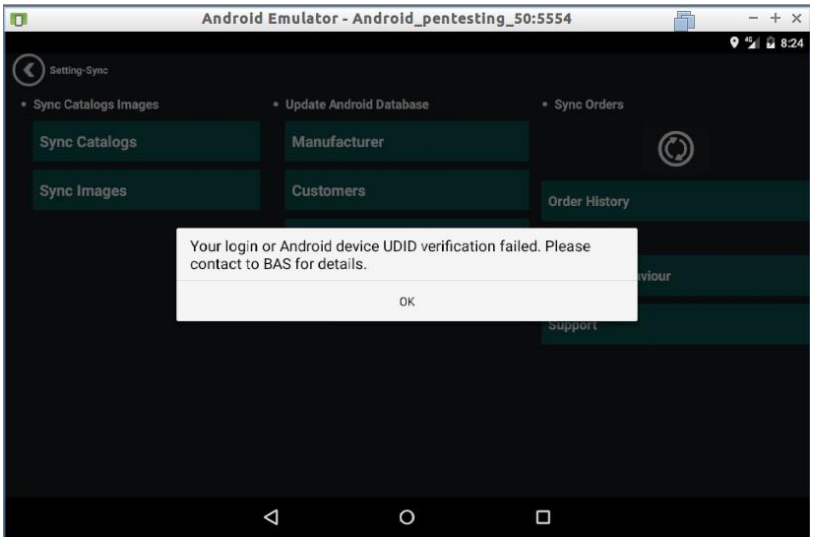
Figura 75. Uso de identificador falso para autenticación.



Fuente: El autor.

Lo que implica que se invalida el identificador del dispositivo logueado previamente:

Figura 76. Invalidación de identificador de dispositivo.



Fuente: El autor.

En la siguiente figura se muestra el fragmento de código en el que se captura el identificador del dispositivo:

Figura 77. Captura identificador del dispositivo.

```
intentFilter.addAction("CLOSE_ALL");
broadcastReceiver = (BroadcastReceiver) (context, intent) -> { LoginActivity.this.finish(); };
registerReceiver(broadcastReceiver, intentFilter);
copyDb();

AppConstants.UDID = Settings.Secure.getString(getContentResolver(),
        Settings.Secure.ANDROID_ID);
AppConstants.Dbobject = DBAdapter
        .getDBAdapterInstance(LoginActivity.this);

user = (EditText) findViewById(R.id.username);
password = (EditText) findViewById(R.id.password);
distributor_cust = (EditText) findViewById(R.id.distributor_cust);
username_cust = (EditText) findViewById(R.id.username_cust);
```

Fuente: El autor.

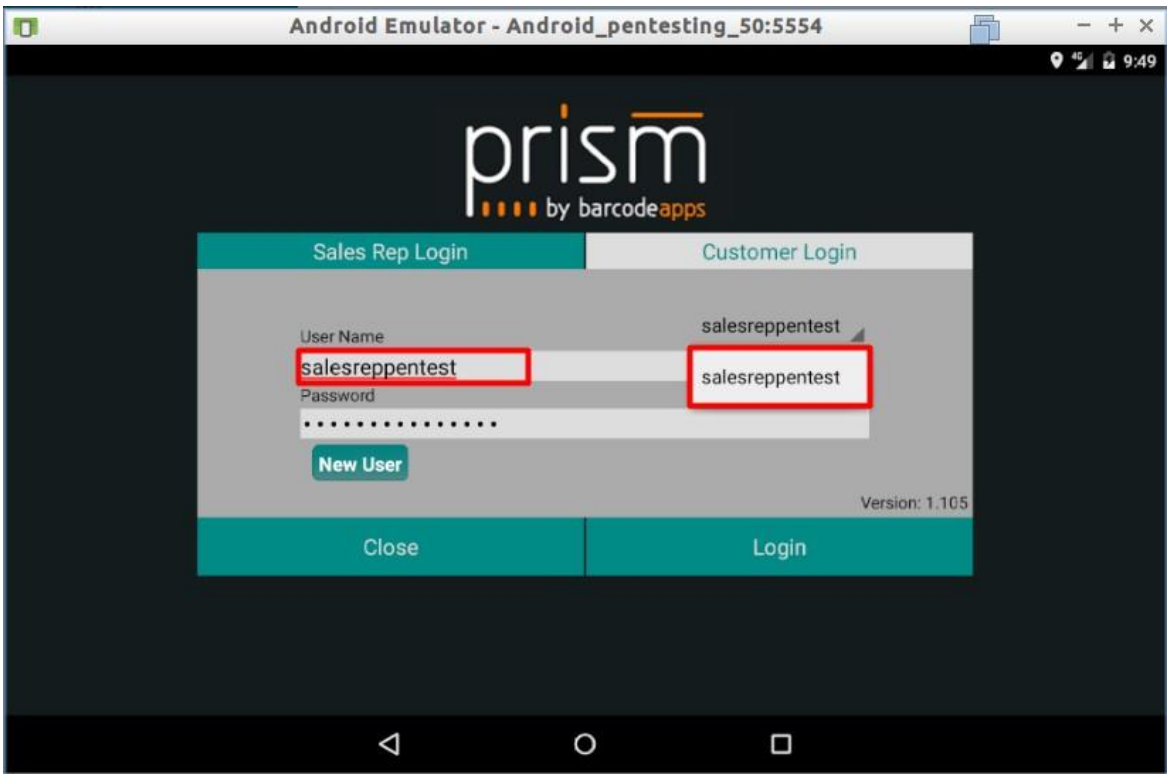
Recomendaciones. En toda aplicación móvil se debe evitar la captura de identificadores del dispositivo, especialmente para realizar procesos de autenticación. Los identificadores de dispositivos en ocasiones pueden estar asociados a información personal de los usuarios.

De acuerdo la revisión hecha al código de la aplicación, se observó que más que identificar el dispositivo en el que ésta se ejecuta, es decir el dispositivo físico en sí; lo que se busca es identificar o controlar el número de instalaciones que se pueden hacer de la aplicación. Esto perfectamente es posible en Android, cuando se usa un ID de instancia o se crea un GUID propio en el momento de la instalación de la aplicación. De acuerdo a los desarrolladores de Android, El ID de instancia fue diseñado explícitamente para este fin; su ámbito se limita a la app, por lo que no se puede usar para realizar un seguimiento de usuarios en diferentes apps y se restablece tras reinstalar la app. En los casos poco comunes en los cuales un ID de instancia no sea suficiente, también se puede usar un GUID.

- **MSTG-27. Opción “Recordar credenciales” activa.** Se pudo constatar que la aplicación por defecto recuerda todos los usuarios que se han autenticado en la

aplicación. En la siguiente figura se muestra como permite autenticarse como vendedor seleccionando una de las opciones de la lista:

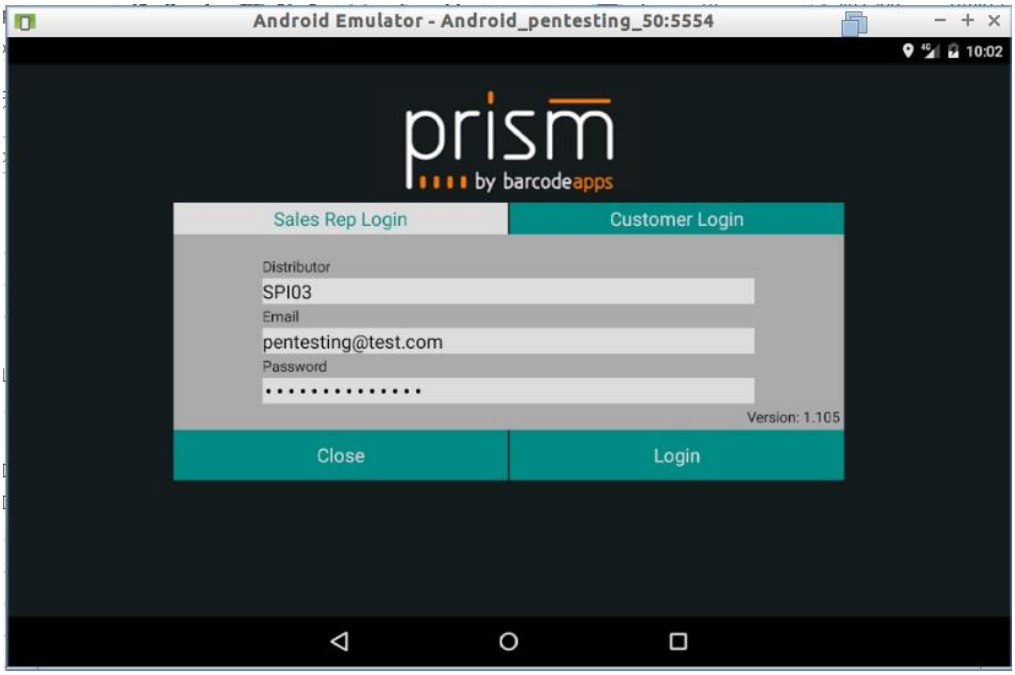
Figura 78. Opción de recordar credenciales para vendedores.



Fuente: El autor.

Y en la siguiente figura se muestra como la aplicación recuerda las credenciales de un usuario cliente:

Figura 79. Opción de recordar credenciales para clientes.



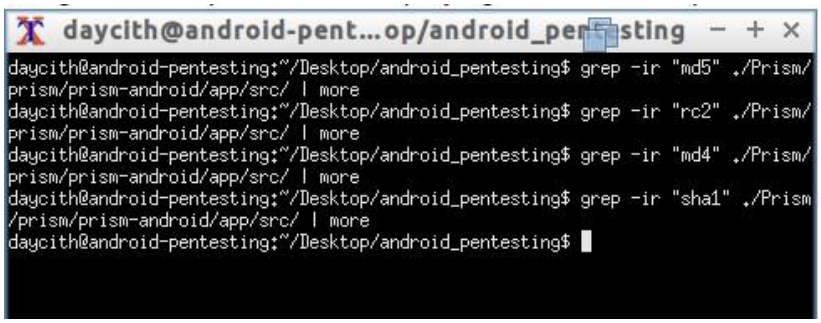
Fuente: El autor.

Esto resulta en una vulnerabilidad, pues en caso en que un atacante tenga acceso a un dispositivo en el que un usuario se haya autenticado correctamente, podrá ingresar a la aplicación sin ningún inconveniente y llevar a cabo acciones a nombre de su víctima, es decir del usuario autenticado previamente.

Recomendaciones. Es completamente comprensible que uno de los requerimientos no funcionales más comunes en aplicaciones móviles es que el usuario no tenga que ingresar sus credenciales cada vez que necesite entrar a la aplicación. Esto resulta conveniente para la comodidad del usuario, pero también implica un problema de seguridad, en casos en donde se presenta pérdida o robo del dispositivo móvil. Ante este hecho los desarrolladores no podrán hacer mucho, pero lo que si pueden hacer es encriptar apropiadamente las credenciales, y ojalá implementar mecanismos que permitan impedir que se siga utilizando la aplicación en el dispositivo.

- **MSTG-28. Aplicación hace uso de Algoritmos de encriptación débiles.** Al realizar una búsqueda de algoritmos de encriptación dentro de la aplicación, no se encontró ningún algoritmo desaconsejado, pero tampoco ningún algoritmo de encriptación seguro:

Figura 80. Búsqueda de algoritmos de encriptación débiles.



Fuente: El autor.

Algo peor que el uso de algoritmos de encriptación débiles, es la no utilización de estos, aun cuando se maneja información que debería estar protegida, por ejemplo la información de usuarios que es almacenada en la base de datos local y/o transferida por un canal de comunicación no seguro.

- **MSTG-29. Envío de información en claro a través de túnel SSL.** Durante la revisión del código de la aplicación e interceptar las diferentes peticiones realizadas a los Web Services, no se encontraron peticiones a través de túneles SSL. No porque no se realice comunicación entre aplicación y el servidor, sino porque toda la comunicación se realiza por el protocolo HTTP, lo que implica un problema mayor.

Recomendaciones. Asumir que la capa de transporte es susceptible a una variedad de tipos de interceptación y manipulación, por lo que, en lo posible, se debe eliminar el envío de los parámetros de conexión a la base de datos alojada en el servidor, esta información debería ser obtenida en los Web Services.

Dejar de transmitir cualquier tipo de información sensible por el protocolo HTTP, y aplicar SSL o TLS para realizar peticiones al servidor.

Validar que efectivamente todas las conexiones se hacen a través de un canal seguro, validar la identidad del servidor, y que su certificado sea expedido por una entidad certificadora reconocida y autorizada. Nunca se deben aceptar certificados auto-firmados.

Para el caso específico de Android, se deben asegurar que no existan instrucciones como org.apache.http.conn.ssl.AllowAllHostnameVerifier y SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER en el código de la aplicación.

Tener en cuenta que incluso las librerías para la encriptación SSL pueden presentar vulnerabilidades, por lo que en la medida de lo posible, se debe implementar una capa de encriptación adicional antes de transferir información sensible por el canal SSL.

- **MSTG-30. Almacenamiento de claves de encriptación localmente.** Durante la revisión del código de la aplicación no se evidenciaron claves de encriptación almacenadas localmente.

- **MSTG-31. La aplicación es vulnerable a XSS.** En las actividades donde se hace uso de WebViews, se encontró la propiedad setJavaScriptEnabled establecida a true, que como su nombre lo indica, permite la ejecución de código JavaScript en el navegador cargado en este tipo de componentes, lo que podría hacer que la aplicación sea a vulnerables a ataques del tipo XSS:

Figura 81. Ejecución de JavaScript activa en WebViews.

Fuente: El autor.

No se encontró ningún fragmento de código que realice una verificación de la URL a cargar en el Webview, y tampoco ningún filtro XSS, lo cual podría implicar un problema de seguridad en el caso en el que un atacante pueda lograr cambiar la URL a cargar en el WebView, la cual es establecida en el fragmento de código marcado en la siguiente figura:

Figura 82. Código con establecimiento de URL en WebView.

Fuente: El autor.

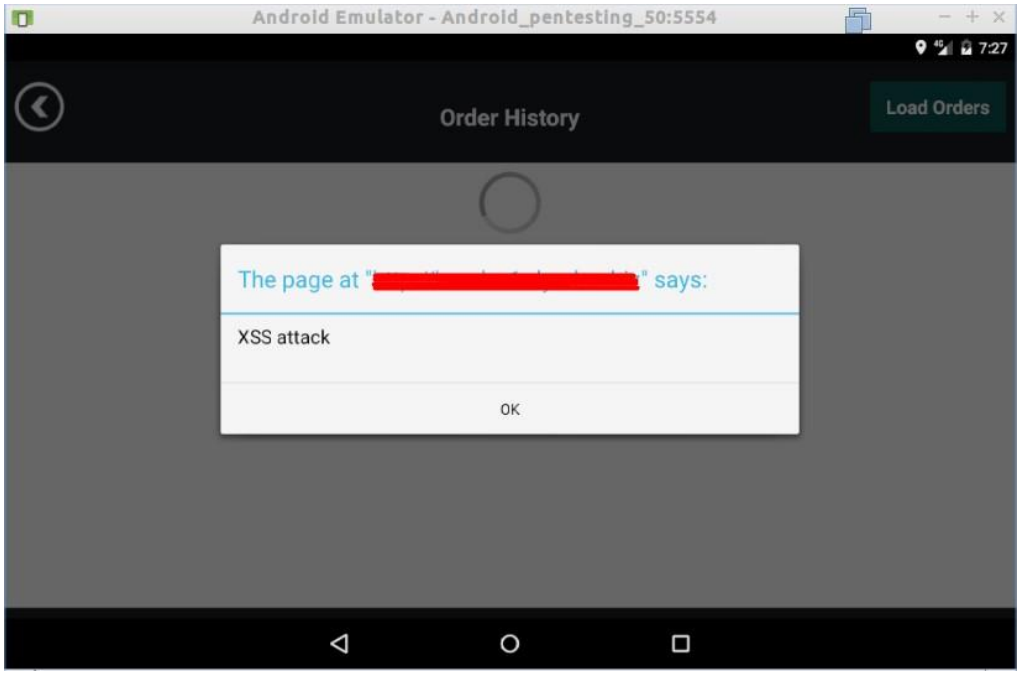
Como se observa en la figura anterior, la URL a cargar en el WebView se establece en la línea 91 de la clase analizada. Aprovechando el hecho de que la aplicación es depurable, se puede reestablecer el valor en la línea de código que le procede, indicando una URL con un código JavaScript malicioso:

Figura 83. Reescritura de URL a cargar en WebView

Fuente: El autor.

La aplicación cargará la nueva URL, ejecutándose también el código JavaScript malicioso:

Figura 84. Ejecución de código malicioso en WebView.



Fuente: El autor.

En este caso solo se ejecuta una alerta creada con código JavaScript, pero se debe tener en cuenta que en Android es posible invocar funciones del API mediante JavaScript (como se hace en aplicaciones híbridas bajo un ambiente “controlado”).

Un ataque por XSS también podría utilizar vulnerabilidades en los WebView para borrar información útil del dispositivo móvil, como los contactos, los identificadores de correo electrónico, y los números de teléfono.

Recomendaciones. La primera medida para tratar de prevenir ataques XSS en aplicaciones móviles Android es desactivar la ejecución de código JavaScript, mediante la instrucción webview.getSettings().setJavaScriptEnabled(false);

Si para el buen funcionamiento de la aplicación es necesario la ejecución de código JavaScript, entonces se deben implementar filtros XSS, como el proporcionado por

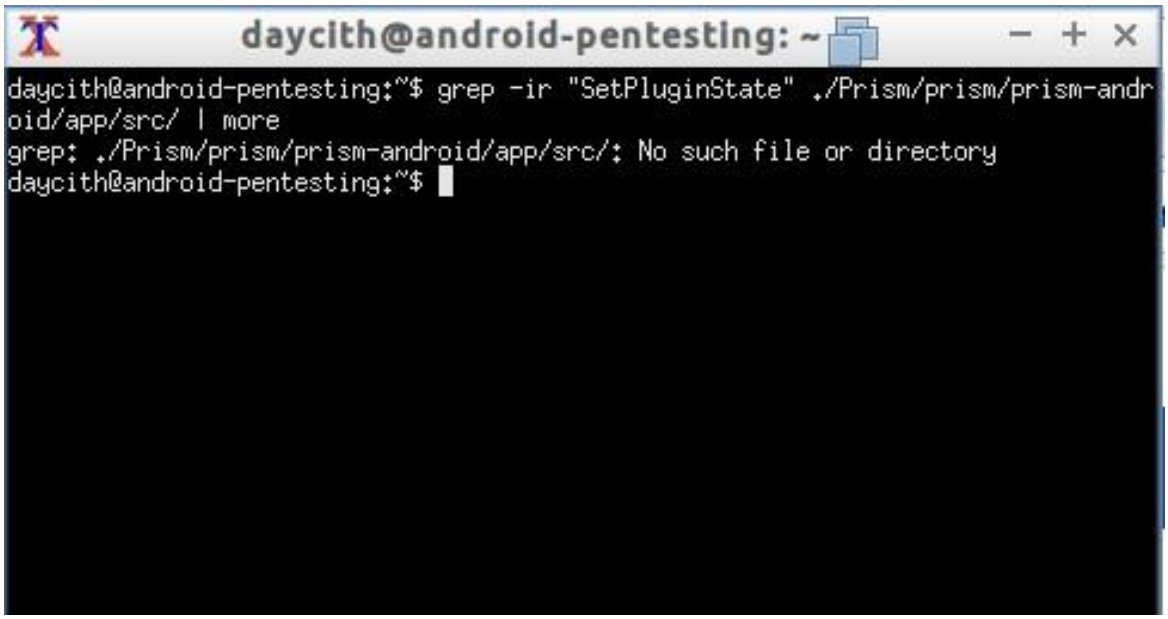
OWASP Java Encoder Project, el cual se encuentra disponible en https://www.owasp.org/index.php/OWASP_Java_Encoder_Project.

Otra buena medida de seguridad es validar cada URL que se intente cargar en el WebView antes de realizar dicha carga, con lo que se reduce el riesgo de ejecutar o mostrar contenido de sitios malintencionados.

- **MSTG-32. Validaciones del lado del cliente pueden ser evadidas.** Como se evidenció en la prueba MSTG-31, en las actividades donde se hace uso de WebViews, se encontró la propiedad setJavaScriptEnabled establecida a true, lo que podría representar una vulnerabilidad a XSS.

Para los WebViews utilizados no se encontró la activación de plugins, lo que indica que en el navegador no podrán ser ejecutados plugins.

Figura 85. Búsqueda activación de plugins en WebViews.



Fuente: El autor.

Adicionalmente, para los WebView no se encontró establecida la propiedad setAllowFileAccess, la cual en Android por defecto se encuentra establecida a true:

Figura 86. SetAllowFileAccess no deshabilitada en WebViews.

Fuente: El autor.

Lo que implica que el WebView tendrá acceso al sistema de archivos del dispositivo, hecho que podría ser aprovechado por un atacante para visualizar o leer el contenido de un archivo del sistema. Como se verá en la prueba MSTG-34, la aplicación no hace uso de Contents Providers, por lo que se puede decir que la aplicación no es susceptible de sufrir una Inclusión local de archivos a través de Contents Providers, pero esto si es posible a través de la modificación en tiempo de ejecución de la URL a cargar en la el WebView, como se verá en la prueba MSTG-35.

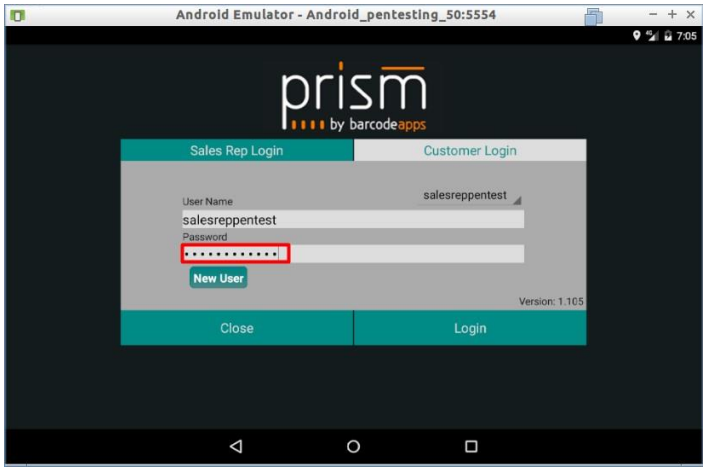
Recomendaciones. En primer lugar, se debe establecer el valor false a la propiedad setAllowFileAccess.

La aplicación debe validar la URL que se está intentando cargar en los WebViews.

- **MSTG-33. Inyecciones SQL.** Se detecta en el login una inyección SQL que permite autenticarse fácilmente dentro de la aplicación. A continuación, se

muestra como ingresando la cadena **x' or 'x'='x'** en el campo de contraseña es posible autenticarse como el usuario vendedor SalesRepPentest:

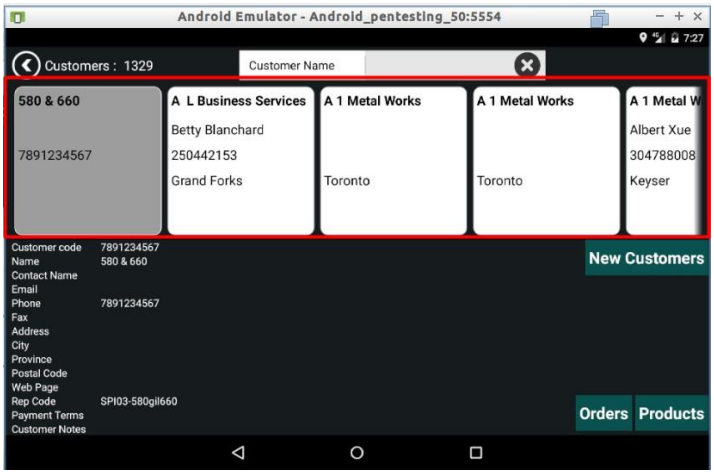
Figura 87. Inyección SQL en autenticación.



Fuente: El autor.

Una vez presionado el botón de Login se puede navegar en la aplicación y seleccionar un cliente para luego capturar un pedido a su nombre:

Figura 88. Listado de clientes luego de inyección SQL en autenticación.



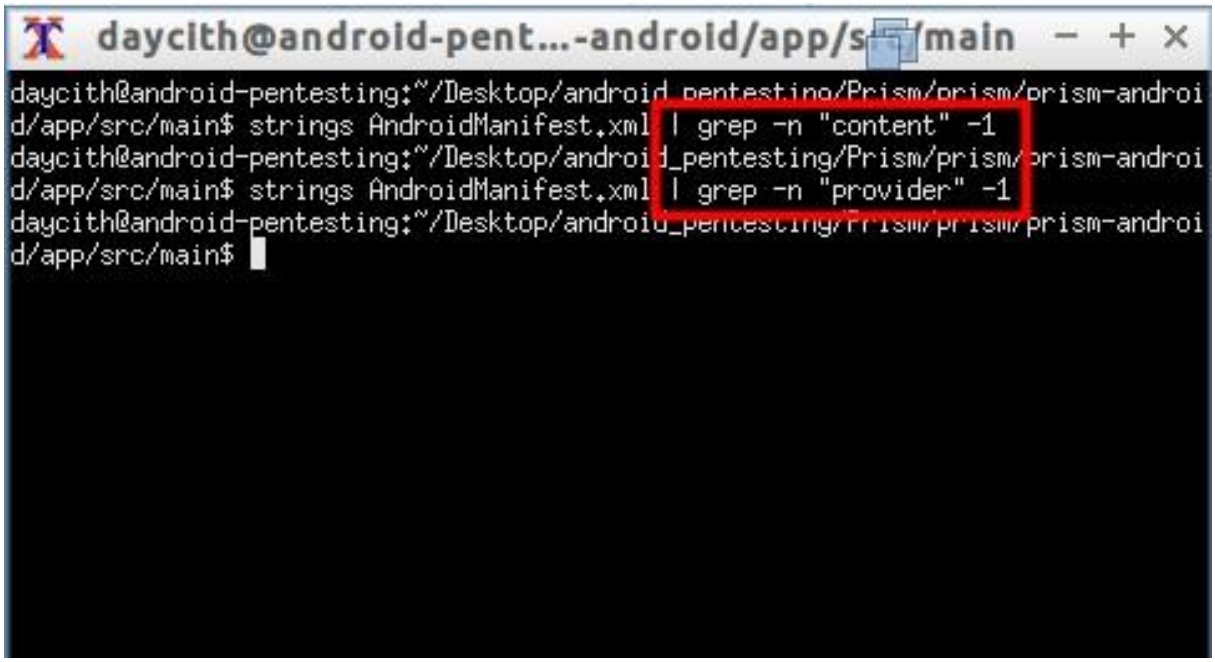
Fuente: El autor.

Este hecho indica que la aplicación es altamente vulnerable frente a ataques de inyecciones SQL.

Recomendaciones. Los desarrolladores de la aplicación deben asegurarse que todas las consultas realizadas a la base de datos estén parametrizadas, especialmente en aquellas construidas tomando los datos ingresados por el usuario.

- **MSTG-34. Fuga de datos en Content Providers.** Luego de realizar una búsqueda en el archivo AndroidManfest.xml, no se encontró uso de Content Providers:

Figura 89. Búsqueda de Content Providers en el código de la aplicación.



Fuente: El autor.

Lo que se ratifica al realizar un scanner de la aplicación:

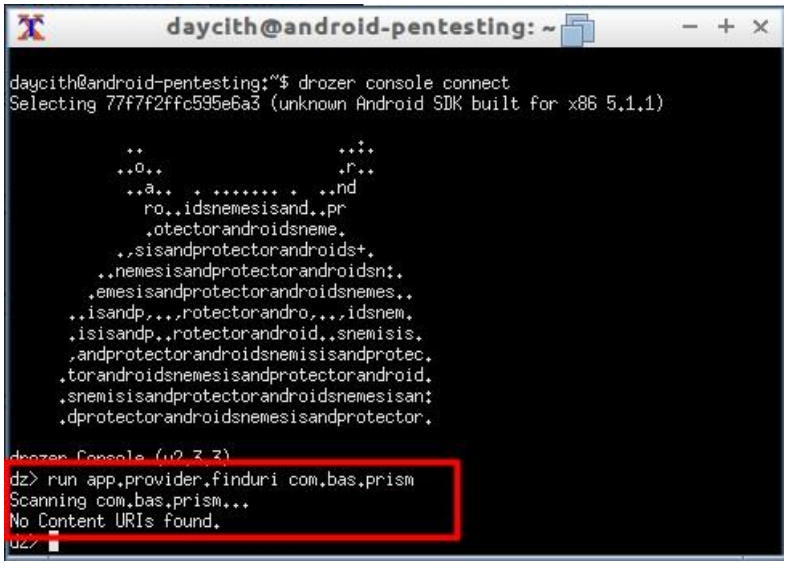
Figura 90. Escaneo de aplicación en busca de Content Providers.

Fuente: El autor.

Esto indica que la aplicación tiene pocas posibilidades de presentar fuga de datos por el mal uso de Content Providers.

- **MSTG-35. Inclusión local de archivos (LFI).** Como se mencionó anteriormente, la inclusión local de archivos no es posible a través de Content Providers:

Figura 91. Escaneo de aplicaciones para encontrar Content Providers URI.



Fuente: El autor.

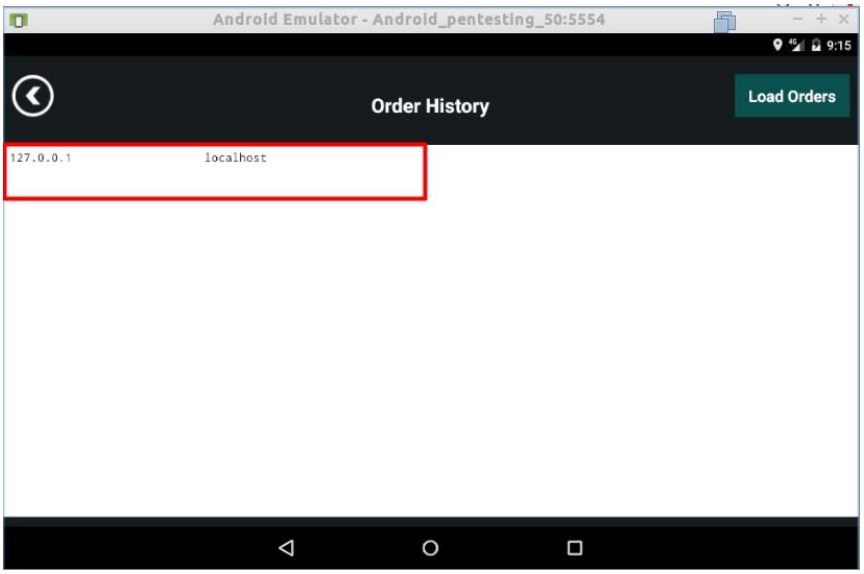
Sin embargo, aprovechando el hecho de que la aplicación es depurable, y que no se realiza una adecuada validación de la URL a cargar en los WebViews, se procedió a cambiar la URL en tiempo de ejecución:

Figura 92. Establecimiento de Ruta de archivo del sistema operativo.

Fuente: El autor.

Como se muestra en la figura anterior, se le indica a la aplicación que cargue el archivo /etc/hosts, propio de cualquier sistema operativo Linux. Gracias a esto se carga el contenido de dicho archivo en el WebView:

Figura 93. Visualización contenido del archivo hosts en WebView.



Fuente: El autor.

Recomendaciones. En primer lugar, se debe establecer el valor false a la propiedad setAllowFileAccess.

La aplicación debe validar la URL que se está intentando cargar en los WebViews antes de hacerlo.

- **MSTG-36. Modificación de URL.** En la prueba MSTG-31 quedó evidenciado como la URL que se debe cargar en el WebView es determinada en el código de la aplicación, basado en la variable estática publica **AppConstants.ManufSvrIP**, cuyo valor proviene de los parámetros enviados por uno de los web services alojados en el servidor, que luego es almacenado en la base de datos, para posteriormente ser consultado desde la aplicación:

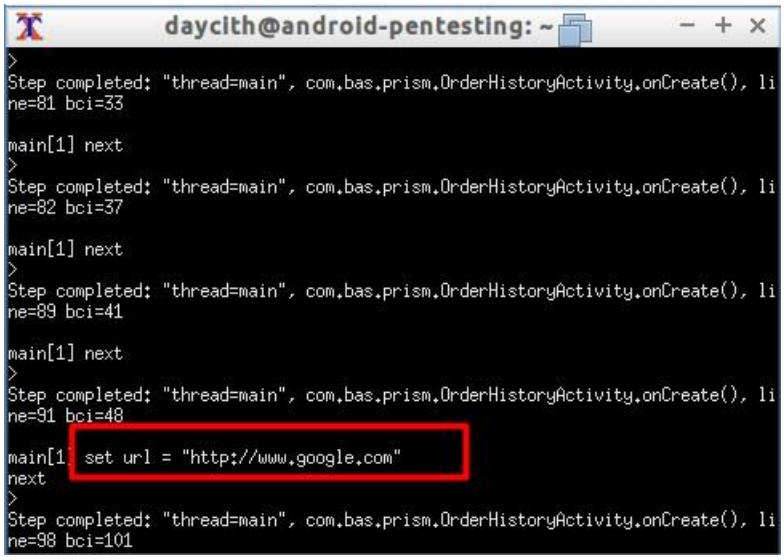
Figura 94. Establecimiento de URL a cargar en WebViews.

```
Cursor cursor2 = AppConstants.Dbobject
        .selectquery("SELECT * FROM Manufacturers WHERE ManufacturerId='"
                + ManufacturerId + "'");
cursor2.moveToFirst();
Log.e("Query-3",
        "SELECT * FROM Manufacturers WHERE ManufacturerId='"
                + ManufacturerId + "'");

AppConstants.ManufacturerID = cursor2.getString(0);
AppConstants.ManufacturerCode = cursor2.getString(1);
AppConstants.ManufacturerName = cursor2.getString(2);
AppConstants.ManufacturerAddress = cursor2.getString(3);
AppConstants.ManufacturerCity = cursor2.getString(4);
AppConstants.ManufacturerStateProv = cursor2.getString(5);

AppConstants.RepCode = cursor2.getString(34);

AppConstants.ManufSvrIP = cursor2.getString(35);
AppConstants.ManufSvrName = cursor2.getString(36);

AppConstants.ManufSvrDatabase = cursor2.getString(37);

AppConstants.ManufSvrDBUname = cursor2.getString(38);

AppConstants.ManufSvrDBPwd = cursor2.getString(39);
```

Fuente: El autor.

Un atacante podría establecer este valor modificando la respuesta del web Service, alterando el valor en la base de datos, o en tiempo de ejecución aprovechando que la aplicación es depurable, como se muestra en la siguiente figura:

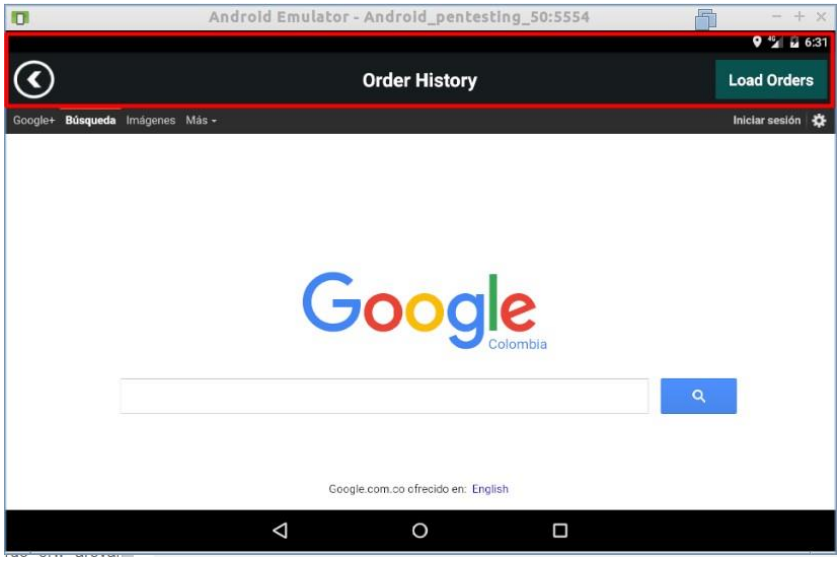
Figura 95. Modificación de URL en tiempo de ejecución.



Fuente: El autor.

Lo que se hizo fue simplemente indicar la URL de Google.com:

Figura 96. Carga de URL modificada en WebView.



Fuente: El autor.

Google es un sitio inofensivo, pero un atacante podría indicarle a la aplicación que cargue una URL que induzca al usuario a ingresar cualquier tipo de información sensible, o a realizar cualquier otro tipo de acción dañina.

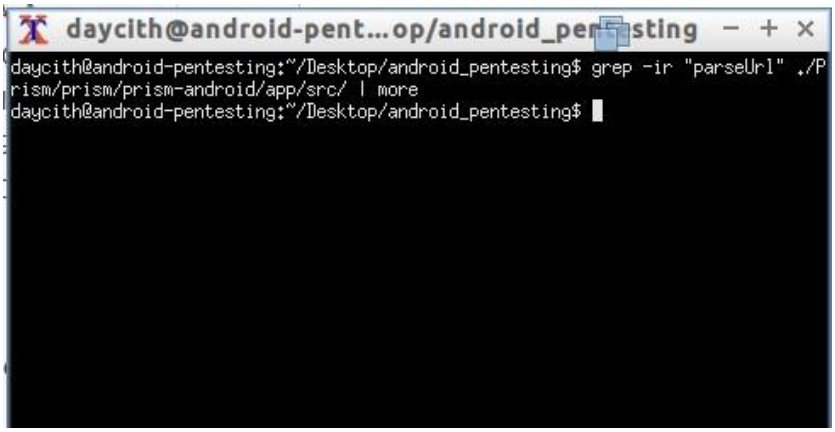
Recomendaciones. La aplicación debe validar la URL que se está intentando cargar en los WebViews antes de hacerlo.

- **MSTG-37. Abuso de componentes a través de IPC Intents.** Como se evidenció en la prueba MSTG-34, en la aplicación no se hace uso de Content Providers, lo que implica también que no existen aparentes riesgos por el uso de Content Providers y posibles abusos de los mismos, derivados de la posible transferencia de datos sensibles entre ellos.

- **MSTG-38. Uso indebido de URL Schemes.** Como se muestra en la Figura 90, la aplicación no hace uso de Servicios, Content Providers o Broadcast Receivers exportados, lo cual implica que es poco probable que la aplicación tenga un uso indebido de URL Schemes.

Tampoco se encontró uso de la función parseUrl dentro de la aplicación, la cual suele ser uno de los puntos cruciales en un problema de abuso de URL Schemes.

Figura 97. Búsqueda de “análisis” de URL dentro de la aplicación.



Fuente: El autor.

- **MSTG-39. Fijación de sesión.** Luego de interceptar y analizar la comunicación entre la aplicación y el servidor, no se encontraron respuestas en las que se estableciera un identificador de sesión en la cabecera “Set-cookie”.

- **MSTG-40. La aplicación no implementa una opción para cerrar sesión.** A pesar de que en la aplicación existe un botón para finalizar sesión dentro de la aplicación:

Figura 98. Botón para finalizar sesión.

Fuente: El autor.

Se evidenció que tanto en la aplicación como en el lado del servidor no se invalida la sesión después de pasar un largo periodo de tiempo. Se dejó de usar la aplicación por más de 30 minutos, pudiendo continuar trabajando en ella al retomarla.

Adicionalmente, en las pruebas MSTG-44, MSTG-45, y MSTG-46 se mostrará cómo se puede acceder directamente a los web services sin necesidad de autenticación

y que no se realizar protección por timeout, lo que implica que no existe adecuado manejo de sesiones en el servidor.

Recomendaciones. Establecer controles de verificación e invalidación de sesiones en el lado del servidor, y no solo en la aplicación. En primera instancia, se debe eliminar flujo actual de verificación o validación de los usuarios, en el que desde la aplicación se realiza primero un llamado a un Web Service para verificar el usuario, y posteriormente se realiza la petición a un Web Service específico, dependiendo de la información requerida. En cada Web Service alojado en el servidor se debe validar la identidad del usuario y el origen de las peticiones.

Luego de implementar lo anterior, se deben establecer políticas para protección por largos periodos sin usar la aplicación o hacer llamados al servidor, teniendo como máximo una hora de inactividad.

Al finalizar sesión dentro de la aplicación, se debe realizar una adecuada limpieza de variables públicas estáticas que almacenan información relacionadas con el usuario que estuvo loggueado en la aplicación, lo que vendría siendo la sesión del usuario.

- **MSTG-41. Uso de Cookies persistentes.** Cuando un usuario ingresa a la aplicación por primera vez, se realiza una verificación de sus credenciales en el servidor, pero en este proceso no se establecen cookies que puedan ser utilizadas para la identificación de las peticiones posteriores, lo que puede ser considerado un problema mayor al que pueda derivar de una inadecuada rotación o la no eliminación de las cookies al momento de cambiar de estado o de usuario dentro de la aplicación.

Otro hecho a resaltar es que dentro de la misma aplicación no se realiza una adecuada limpieza de variables públicas estáticas que almacenan información relacionadas con el usuario que estuvo loggueado. En la pruebas MSTG-21 y MSTG-22 se evidenció que luego de “cerrar la sesión” de un usuario vendedor, fue posible evadir el sistema de autenticación y/o ingresar como un usuario cliente y posteriormente ejecutar funcionalidades basadas en la información del usuario vendedor, que supuestamente había cerrado sesión mediante el botón “Logout”.

Recomendaciones. Se debe eliminar la referencia directa que actualmente existe a los diferentes Web Services alojados en el servidor, e implementar un control adecuado de sesiones. La sesión del usuario debe ser destruida cada vez que se presenta algunas de las siguientes situaciones:

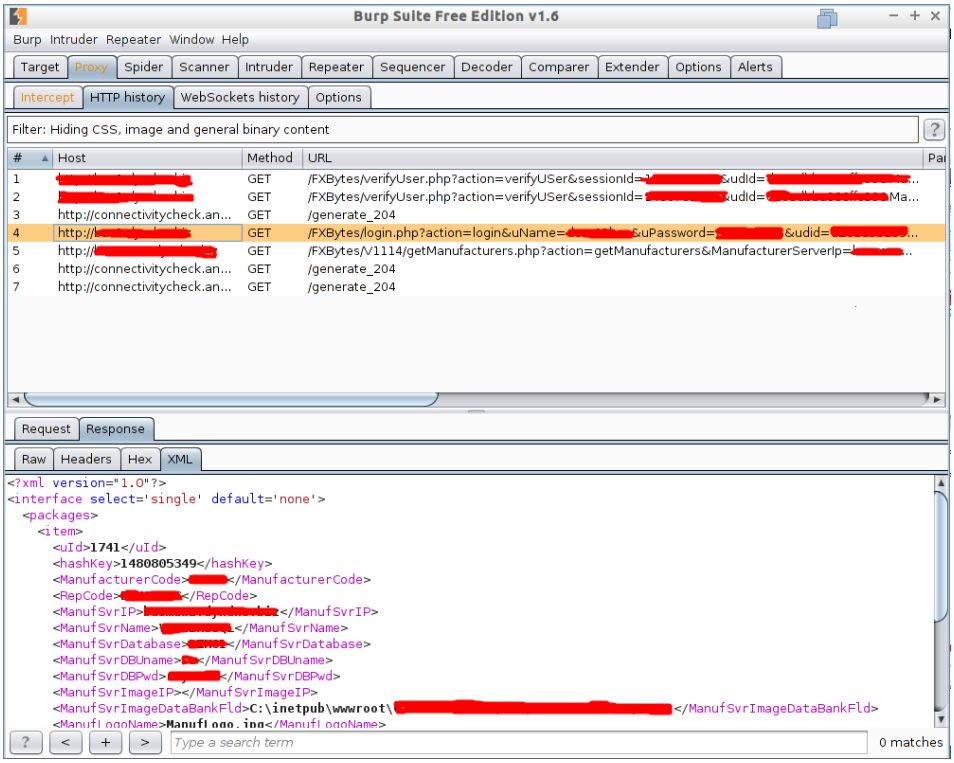
- Autenticación de un usuario.
- Cambio de usuario.
- Salida de la aplicación
- Tiempos de inactividad dentro de la aplicación y/o peticiones al servidor.

Del lado de la aplicación móvil se debe realizar una adecuada limpieza de variables públicas estáticas que almacenan información relacionadas con el usuario que estuvo loggueado.

- **MSTG-42. Tokens inseguros.** Luego de hacer una revisión del código de la aplicación no se encontraron tokens almacenados y/o transferidos entre la aplicación y los Web Services con los que ésta interactúa.

- **MSTG-43. Contraseñas legibles en respuestas desde Web Services.** Se interceptaron las peticiones realizadas al servidor y sus respectivas respuestas. En algunos casos se detectaron respuestas con datos sensibles perfectamente legibles. Como ejemplo de esto se destaca el caso en el que se ejecuta la aplicación por primera vez, en la que la autenticación del usuario se realiza contra el servidor. Si el proceso de autenticación es exitoso, el servidor envía un XML con datos de la compañía y de la base de datos remota:

Figura 99. Contraseñas legibles en respuestas del servidor.



Fuente: El autor.

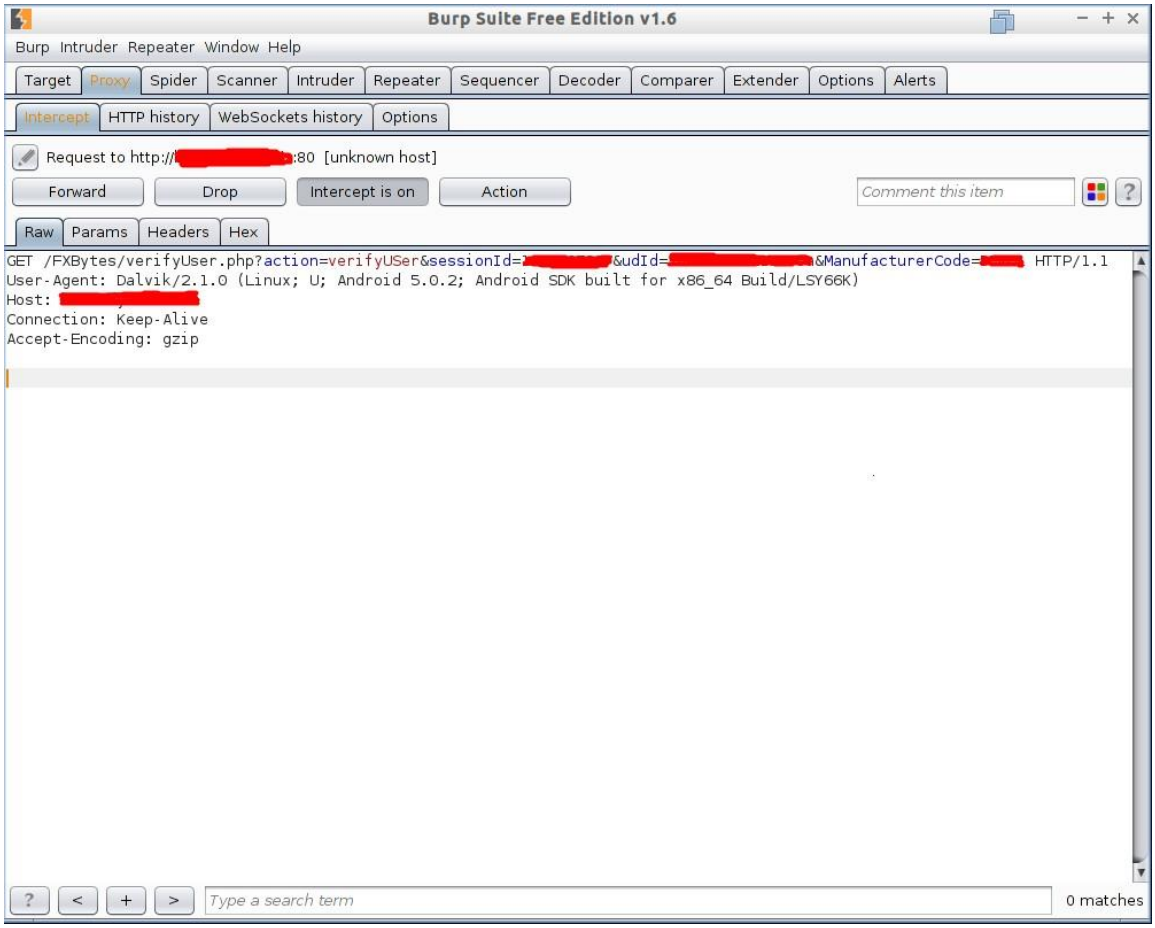
Si un atacante intercepta esta petición y la respuesta del servidor, se apoderará de información que le permitirá establecer conexión con la base de datos de la compañía y literalmente realizar las modificaciones que desee.

Recomendaciones. En primer lugar, el Web Service encargado de la autenticación en el servidor no debería enviar los parámetros de conexión a la base de datos remota, estos deben ser determinados del lado del servidor, basados en el identificador del usuario o de su compañía.

En segundo lugar, y como se ha mencionado en varias ocasiones, la conexión entre la aplicación y los Web Services en los que se realice envío de información sensible, debe hacerse bajo un canal de comunicación seguro.

- **MSTG-44. Referencia directa a recursos internos sin autenticación.** En el flujo de comunicaciones entre la aplicación y el servidor se intenta validar las peticiones realizadas a éste último. Cuando una funcionalidad especifica requiere la comunicación con el servidor, primero se envía una petición a un Web Service que verifica el usuario con el que se hará la siguiente petición al servidor:

Figura 100. Verificación de usuario en el servidor.

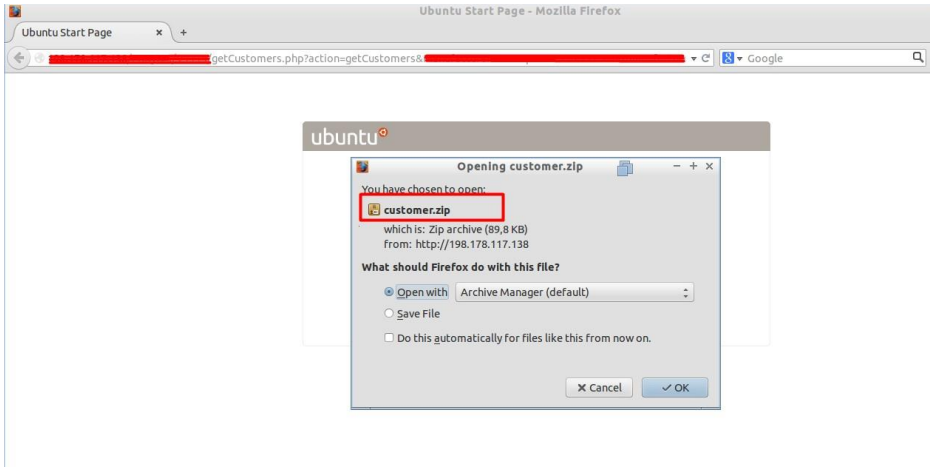


Fuente: El autor.

Si el servidor retorna una respuesta exitosa, la aplicación realizará la siguiente petición, de lo contrario mostrará un mensaje al usuario indicándole que el identificador del dispositivo es inválido. El problema de este método de validación es que si la siguiente petición se realiza desde una herramienta de pentesting, o

incluso desde un navegador web, el servidor llevará a cabo la funcionalidad solicitada, sin ningún tipo de validación de la identidad del usuario:

Figura 101. Referencia directa a recursos internos sin autenticación



Fuente: El autor.

Lo que implica que existe una referencia directa a los Web Services, sin ningún tipo de comprobación del origen de las peticiones en el lado del servidor.

Recomendaciones. Se debe eliminar la referencia directa que actualmente existe a los Web Services alojados en el servidor, e implementar un control adecuado de sesiones. La sesión del usuario debe ser destruida cada vez que se presenta algunas de las siguientes situaciones:

- Autenticación de un usuario.
- Cambio de usuario.
- Salida de la aplicación
- Tiempos de inactividad dentro de la aplicación y/o peticiones al servidor.

- **MSTG-45. Aplicación no tiene o implementa indebidamente un sistema de manejo de sesiones.** El hecho de que se pueda acceder a recursos del servidor sin necesidad de autenticación, implica también que no se implementa un adecuado sistema de manejo de sesiones.

Recomendaciones. Se debe eliminar la referencia directa que actualmente existe a los Web Services alojados en el servidor, e implementar un control adecuado de sesiones. La sesión del usuario debe ser destruida cada vez que se presenta algunas de las siguientes situaciones:

- Autenticación de un usuario.
- Cambio de usuario.
- Salida de la aplicación
- Tiempos de inactividad dentro de la aplicación y/o peticiones al servidor.

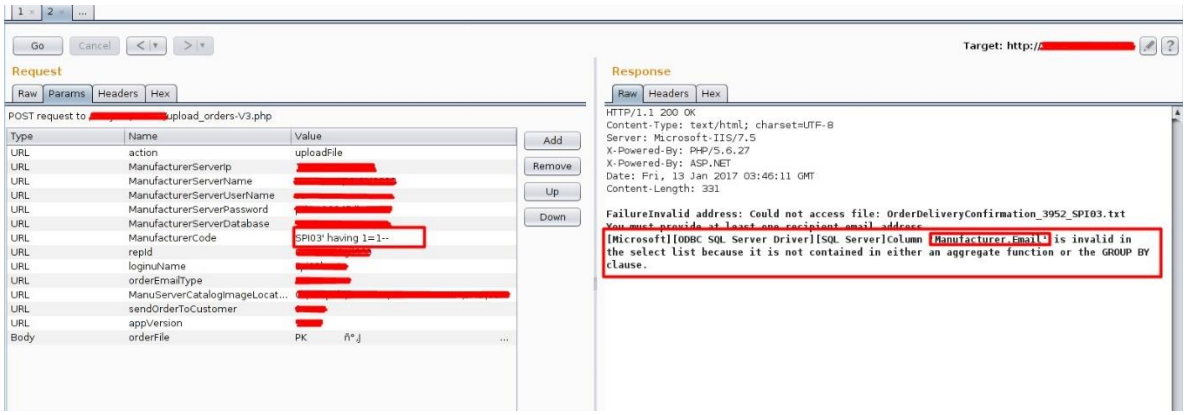
- **MSTG-46. Falta de protección adecuada por "Timeout".** Otro problema que repercute en el hecho de que se pueda acceder directamente a los recursos del servidor, sin ningún tipo de invalidación de sesión; es que no hay protección de los recursos del servidor en un tiempo determinado de inactividad dentro de la aplicación.

Recomendaciones. Se debe eliminar la referencia directa que actualmente existe a los Web Services alojados en el servidor, e implementar un control adecuado de sesiones. La sesión del usuario debe ser destruida cada vez que se presenta algunas de las siguientes situaciones:

- Autenticación de un usuario.
- Cambio de usuario.
- Salida de la aplicación
- Tiempos de inactividad dentro de la aplicación y/o peticiones al servidor.

- **MSTG-47. Validación incorrecta de entradas en el lado del servidor.** Al interceptar y volver a enviar algunas de las peticiones se evidenció que no se realiza una adecuada validación de entradas en el lado del servidor. En la siguiente figura se muestra la manipulación de la petición que la aplicación realiza para enviar órdenes al servidor:

Figura 102. Validación incorrecta de entradas en el lado del servidor – obtención nombre tabla.

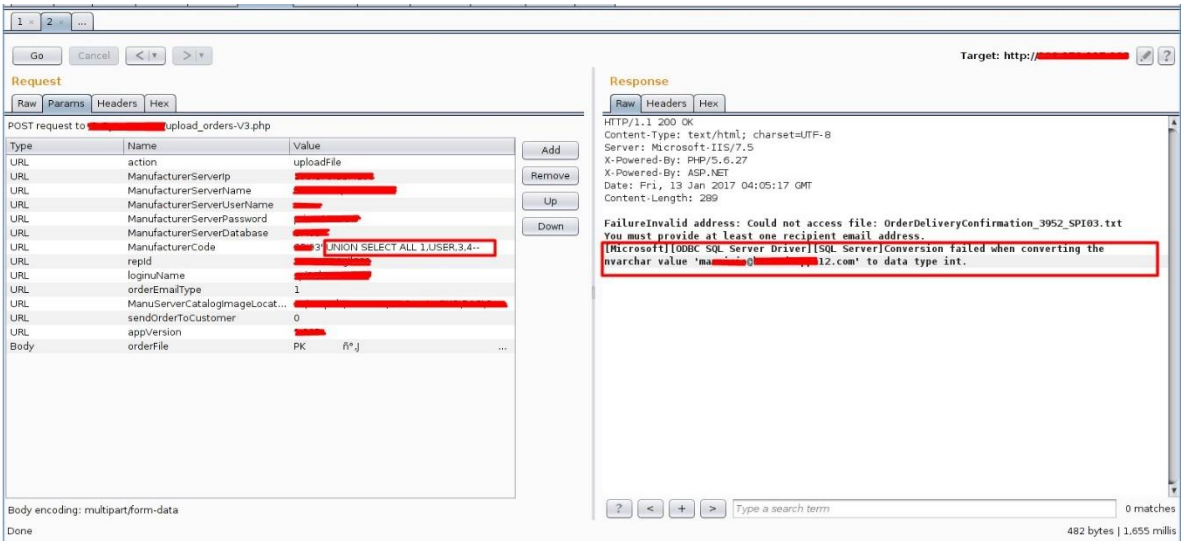


Fuente: El autor.

Al agregar la cadena de texto **having 1=1--** en el valor de uno de los parámetros, se obtiene un mensaje de error en el que se muestra información de la base de datos remota, específicamente la tabla Manufacturer y el campo Email.

Si en uno de los parámetros de la misma petición se agrega la cadena de texto **UNION SELECT ALL 1, USER, 3,4--** se obtiene información de uno de los usuarios almacenados en la base de datos:

Figura 103. Validación incorrecta de entradas en el lado del servidor – obtención usuario.



Fuente: El autor.

Esto implica que efectivamente no se realiza una adecuada validación de las entradas del lado del servidor, lo que en este caso en específico refleja también falta de protección frente a ataques de inyecciones SQL.

En pruebas posteriores se verá como en el servidor se presentan otros problemas derivados de una inadecuada validación de las entradas en el lado del servidor. En la prueba MSTG-49, se evidenciará cómo es posible subir archivos maliciosos al servidor, por no realizarse una validación del tipo de archivos que contiene el archivo ZIP que se envía al momento de sincronizar las órdenes; y en la prueba MSTG-50, se verá también que es posible hacer un ataque de XSS almacenado o persistente, por no validarse los datos ingresados por el usuario en la aplicación y que posteriormente se envía al servidor, también al momento de sincronizar órdenes o pedidos.

Recomendaciones. Básicamente todas las entradas en el servidor deben ser validadas utilizando “una lista blanca” de caracteres permitidos y realizar el escapado de caracteres especiales. El lenguaje de programación PHP proporciona funciones para este propósito. El OWASP también proporciona el ESAPI (Enterprise

Security API), una librería que encapsula una gran cantidad de funcionalidades para la validación de entradas.

Como quedó evidenciado, una de las consecuencias de la pobre validación de las entradas del lado del servidor es la posibilidad de realizar inyecciones SQL, por lo que también se recomienda utilizar sentencias preparadas para realizar todas las consultas en la base de datos del servidor.

- **MSTG-48. Se muestra información interna sensible en páginas de error.** En la prueba anterior se evidencia también como información sensible del servidor se muestra al producirse algunos errores, causados a su vez por la incorrecta validación de las entradas.

Recomendaciones. La primera medida de seguridad recomendada es realizar una adecuada validación de las entradas, con el objetivo de reducir la posibilidad que se produzcan los errores.

Como lo anterior no siempre será posible, se debe tratar de mostrar la menor cantidad de información del error presentado, por lo menos en los servidores de producción, esto debe hacerse en la configuración del servidor web y del archivo PHP.ini, a través de directivas como error_reporting y display_erros.

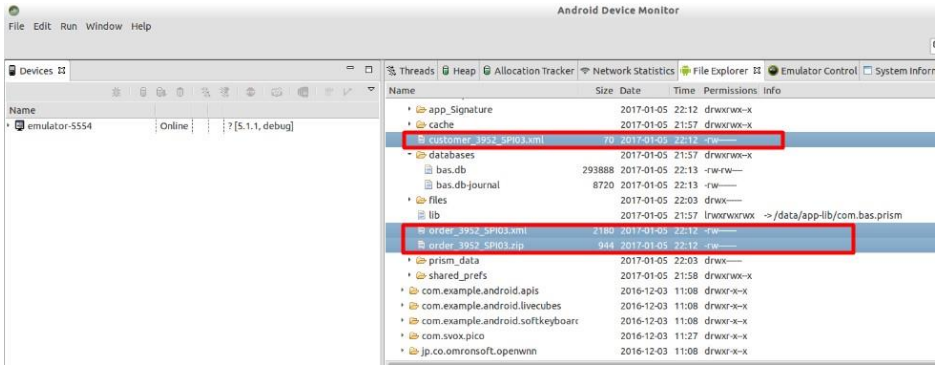
- **MSTG-49. Subida de archivos maliciosos en el servidor.** En la petición utilizada para enviar órdenes al servidor:

Figura 104. Sincronización de órdenes desde la aplicación.

Fuente: El autor.

Se detecta que se envía un archivo ZIP, con dos archivos XML's que contienen información de las órdenes a enviar y de los nuevos clientes, en caso en el que estos últimos sean creados desde la aplicación:

Figura 105. Archivos enviados al servidor en al subir órdenes.



Fuente: El autor.

Se encontró que en el servidor el archivo ZIP es descomprimido en la misma ruta donde se encuentra el Web Service que recibe la petición, por lo que se procedió a retomar el archivo ZIP de la petición anterior y agregarle un archivo con extensión PHP. Este archivo contiene la función phpinfo(), la cual permite obtener información sobre la versión de PHP, sus opciones de compilación y extensiones, información del servidor y entorno, etc.

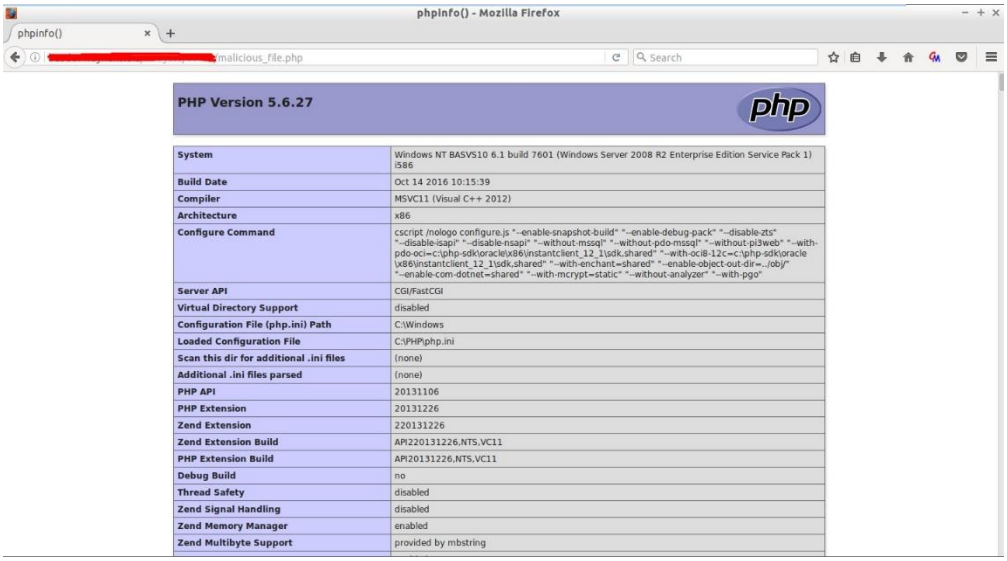
También se creó un pequeño script que consta de un simple formulario que tiene como destino la ruta del Web Service, con un campo del tipo File, para enviar el archivo ZIP que contiene el archivo “malicioso”:

Figura 106. Subida de archivo malicioso al servidor.

Fuente: El autor.

Luego de enviar el formulario con el archivo malicioso y obtener una respuesta exitosa, se procedió a acceder al archivo malicioso desde el navegador, logrando ver información sensible del servidor:

Figura 107. Ejecución de archivo malicioso en el servidor.



Fuente: El autor.

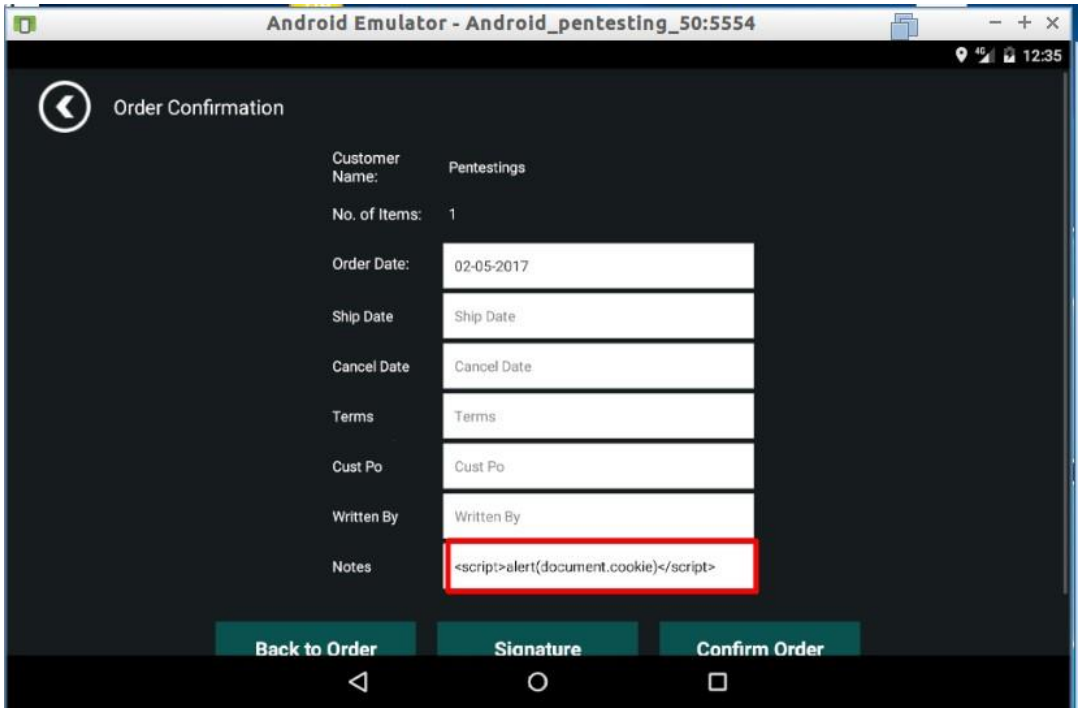
Si un atacante realiza este proceso, pero con un tipo de archivo ejecutable, los resultados pueden ser más comprometedores.

Recomendaciones. En lo posible se debe sustituir el intercambio de información entre la aplicación y el servidor a través de archivos XML comprimidos en archivos ZIP´s por JSON, un formato mucho más liviano y menos propenso a errores.

Antes de descomprimir el archivo ZIP en el servidor, se debe validar el tipo y el peso de los archivos que este contiene.

- **MSTG-50. XSS (Cross Site Scripting).** Desde la aplicación móvil se completó el flujo para crear una orden a nombre del cliente de prueba creado anteriormente, pero se ingresó una cadena “maliciosa” en uno de los campos disponibles en el formulario:

Figura 108. Ingreso Cadena de maliciosa en formulario de creación de orden.

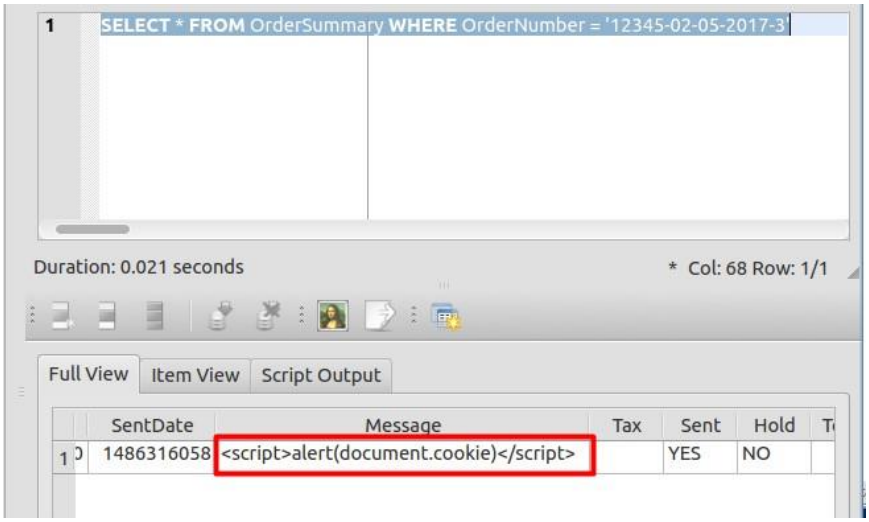


Fuente: El autor.

Como se muestra en la figura anterior, se ingresó la cadena “<script>alert(document.cookie)</script>”, que al ser ejecutada en un navegador web mostrará una alerta con la información de las cookies.

Luego de completar la orden se evidenció que en la base de datos se guarda la cadena sin ningún tipo de escapado de los caracteres especiales, lo que implica un problema de validaciones incorrecta del lado del cliente:

Figura 109. XSS almacenado en base de datos SQLite.



Fuente: El autor.

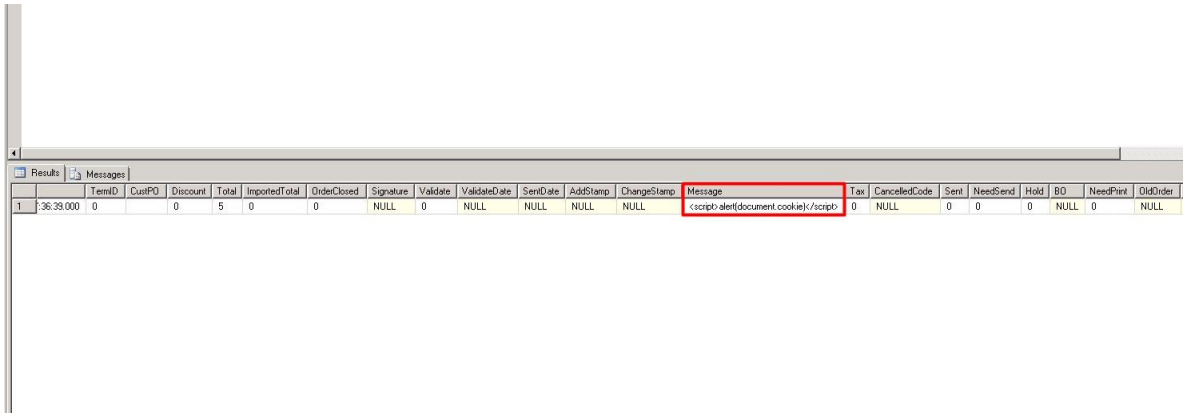
Si luego de completar la orden en el dispositivo móvil, esta se envía al servidor, se creará un archivo XML, en el que se evidencia la presencia de la cadena maliciosa ingresada en el formulario de la aplicación móvil:

Figura 110. Cadena Maliciosa en archivo XML de orden.

Fuente: El autor.

Archivo que al ser enviado al Web Service encargado de procesar la orden, se inserta un registro en la base de datos del servidor, lo que se convierte un ataque del tipo XSS almacenado o persistente, el más peligroso de todos los tipos de ataque por XSS:

Figura 111. Ejecución de archivo malicioso en el servidor.



Fuente: El autor.

Si esta orden es visualizada en algún navegador web, sin ningún tipo de filtro contra XSS, se convertirá en XSS reflejado, que en este caso solo mostrará la cookie del usuario logueado, pero podría contener un código JavaScript mucho más complejo y con resultados mucho más comprometedores.

Recomendaciones. La regla básica a seguir es escapar, codificar, o filtrar cada dato ingresado por el usuario. Si los datos no son ingresados por un usuario, pero se suministran a través del parámetro GET, también deben pasar por los filtros. Incluso un formulario POST puede contener vectores XSS. Por lo tanto, cada vez que se va a utilizar un valor variable en el sitio web, se debe pasar por un filtro contra XSS.

En primer lugar, se deben escapar caracteres especiales como:

- <
- >
- ‘’(comillas simples)
- “” (comillas dobles)
- &
- /
- Entre otros.

En PHP existen funciones como htmlspecialchars(), que ayudan en este propósito.

Pero algo mejor que escapar o filtrar una serie de caracteres específicos, es decir una lista negra, es manejar una “lista blanca”, en la que se definan los caracteres permitidos, cualquier otra cosa diferente será filtrada.

Los filtros XSS deben ser aplicados tanto en la aplicación móvil como en el conjunto de Web Services. En la prueba se evidenció que la cadena con el código malicioso fue ingresada en la aplicación móvil, almacenada en la base de datos local, e incluida en un archivo XML que posteriormente fue enviado al servidor, donde el XSS almacenado se convierte en una amenaza mayor.

El proyecto OWASP ESAPI ha producido un conjunto de componentes de seguridad reutilizables en varios idiomas, incluyendo rutinas de validación y escape para prevenir la manipulación de parámetros y la inyección de ataques XSS, este proyecto puede ser encontrado en la URL https://www.owasp.org/index.php/ESAPI.

- **MSTG-51. CORS (Cross Origin Resource Sharing).** El análisis realizado con las herramientas automáticas no arrojó la presencia de esta vulnerabilidad en el servidor. También se realizaron pruebas manuales, primero realizando una petición GET, enviando las siguientes cabeceras:

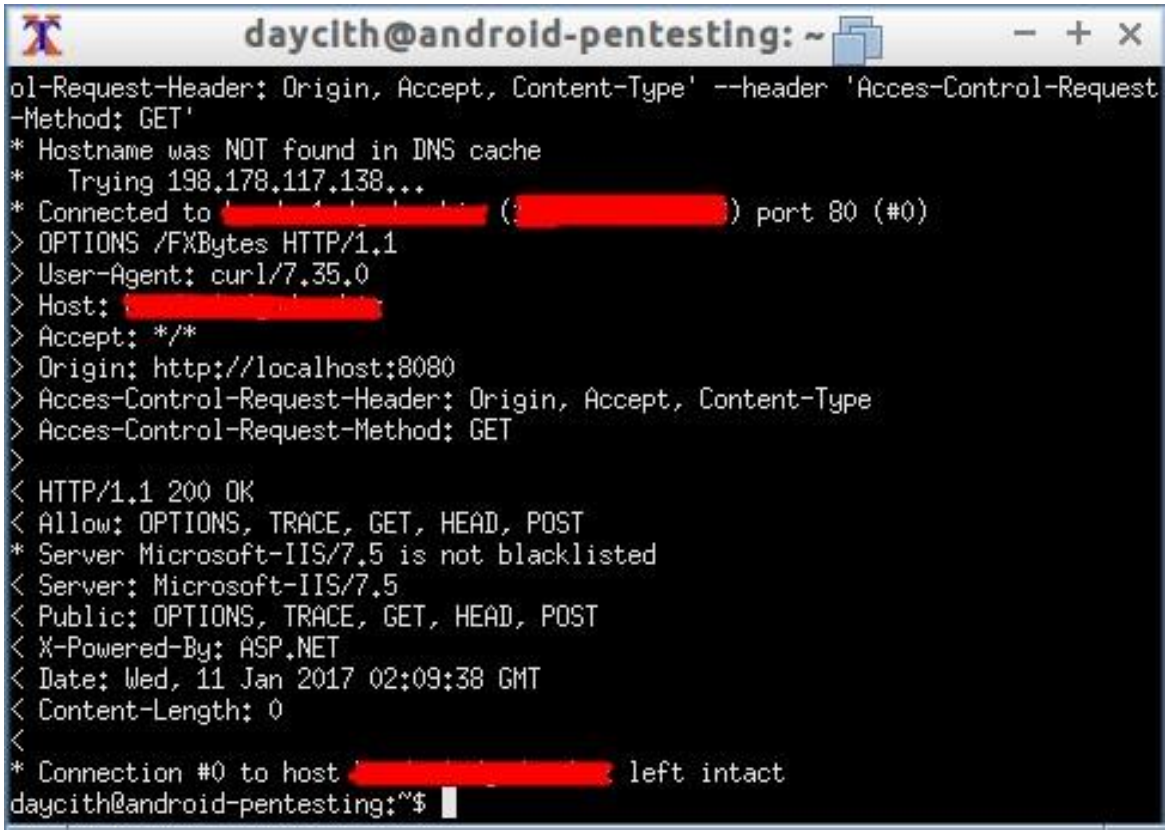
'Origin: http://localhost:8080'

'Acces-Control-Request-Header: Origin, Accept, Content-Type'

'Acces-Control-Request-Method: GET'

Para dicha petición se obtuvo una respuesta en la que no está presente la directiva **Access-Control-Allow-Origin** con el comodín ***:**

Figura 112. Validación CORS en el servidor.



Fuente: El autor.

Adicionalmente se crea un pequeño script que permite validar XSS y CORS. Al intentar ser cargado en el navegador web, enviándole como parámetro un link

externo al servidor se evidencia el error en la consola del navegador, producto de la restricción SOP establecida por defecto en los servidores web:

Figura 113. Validación CORS en el servidor desde el navegador.

Fuente: El autor.

- **MSTG-52. Es posible realizar peticiones HTTP diferentes a GET y POST.** Al realizar un escáner con una de las herramientas automáticas, se evidencia que aparte de GET y POST, el servidor tiene activo los siguientes métodos HTTP: HEAD, OPTIONS, y TRACE, este último significando un riesgo potencial. Cuando este método está activado, el servidor responderá a las solicitudes que utilizan el método TRACE haciendo eco en su respuesta a la petición exacta que se envió. Esto puede llegar ser aprovechado por un atacante para hacer que un usuario haga una petición TRACE al servidor y recuperar la respuesta de éste último, con lo que el atacante estará en capacidad de capturar cualquier dato sensible, como puede ser las cookies del usuario, lo que puede aumentar los impactos de otro tipo de ataque, como XSS.

Figura 114. Identificación de métodos HTTP activos en el servidor.

Fuente: El autor.

Recomendaciones. El método HTTP Trace debe ser deshabilitado en el servidor.

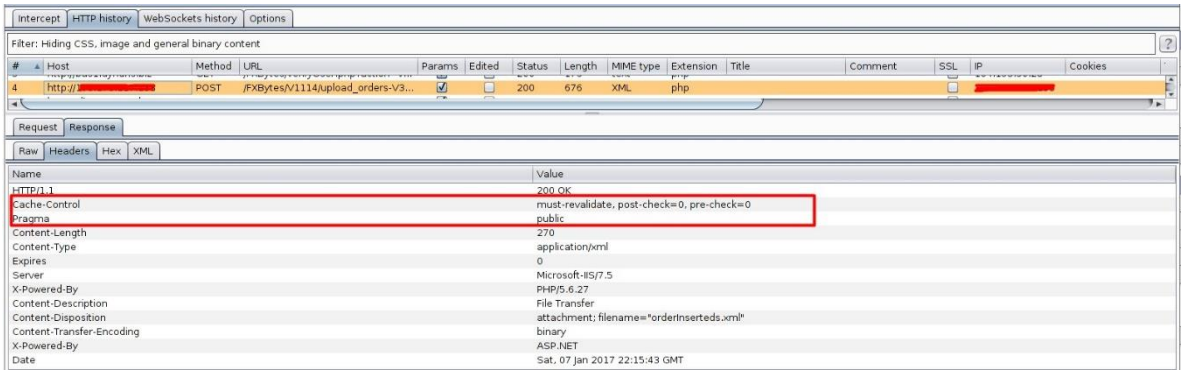
- **MSTG-53. Falsificación de petición en sitios cruzados (Cross Site Request Forgery (CSRF)/SSRF).** Debido a que un ataque del tipo CSRF se basa en el engaño de un usuario que se encuentra correctamente autenticado en un sistema para inducirlo a realizar determinada acción, y que gracias a que en los Web Services no se realiza una adecuada validación del origen de las peticiones y de la identidad del usuario que las hace; se puede decir que no es necesario realizar esta prueba. Un atacante no necesita poner un enlace en un sitio web, correo electrónico, o cualquier otra herramienta de ingeniería social para lograr que se lleve a cabo determinada acción en el servidor, sólo tendrá que hacer el llamado directamente a cualquiera de los Web Services, desde donde quiera, las veces que quiera.

Recomendaciones. Como primera medida, se deberá eliminar la referencia directa que actualmente existe a la funcionalidad expuesta por los diferentes Web Services alojados en el servidor.

Luego, se debe implementar un manejo adecuado de las sesiones, en la que se tenga plena identificación del usuario para el que se realizan las peticiones, preferiblemente con un sistema de tokens de usuarios únicos, lo suficientemente extensos y aleatorios. Esto último sería la opción más recomendada, debido a se trata de un conjunto de Web Services que pueden ser accedidos por las aplicaciones móviles o sistema externos, en lo que otras políticas de protección contra CSRF no pueden ser aplicadas, como la validación del "mismo origen" y tokens CSRF, los cuales son más apropiados para la protección en un sitio web.

- **MSTG-54. Respuestas HTTPS almacenadas en caché.** Se podría decir que esta prueba no se debería aplicar, debido a que todos los servicios web son accedidos vía HTTP. Sin embargo, durante la revisión de las peticiones se notó que las cabeceras **cache-control** y **Pragma** no contienen los valores apropiados en las respuestas de los servicios web:

Figura 115. Cabeceras Cache-control y pragma en respuestas del servidor.



Fuente: El autor.

Esto implica que aun cuando los servicios se accedieran vía HTTPS, existiría la posibilidad que los datos sensibles pudieran ser accedidos en el cache local de los navegadores web o de la aplicación móvil.

Recomendaciones. Establecer los siguientes valores en cada respuesta del servidor:

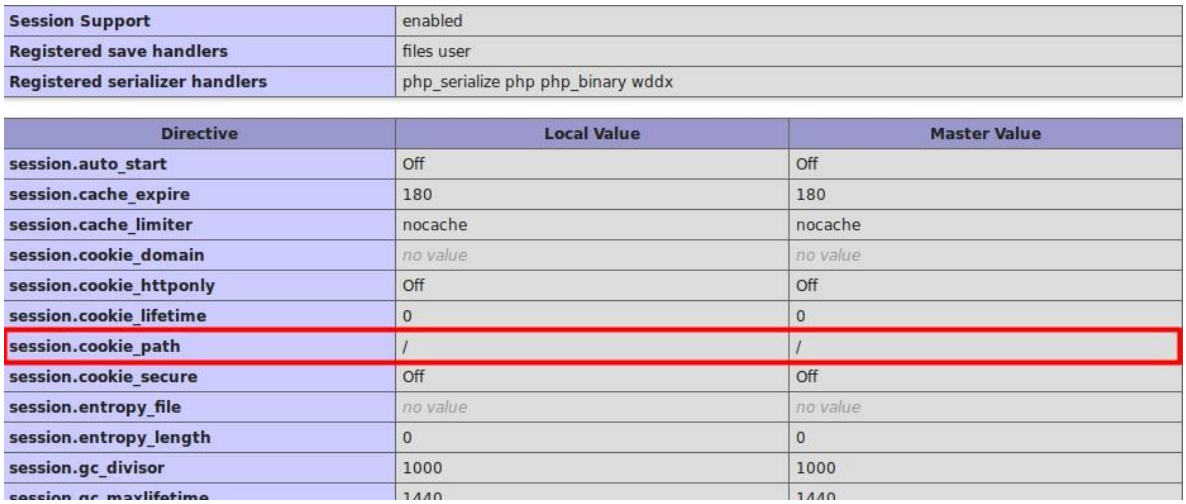
Cache-control: no cache, no-store

Pragma: no-cache

- **MSTG-55. Propiedad o atributo "Path" no establecido en Cookies.** Se encontró que la propiedad o atributo session.cookie.path tiene el valor "/", lo que significa que las cookies creadas en el servidor aplicarán para todos los dominios alojados en el servidor.

Lo anterior implica un problema de seguridad, ya que, si un atacante logra obtener información de las cookies de un dominio alojado en el servidor, podrá trascender su ataque a los demás dominios o sitios.

Figura 116. Identificación de métodos HTTP activos en el servidor.



Session Support	enabled
Registered save handlers	files user
Registered serializer handlers	php_serialize php php_binary wddx

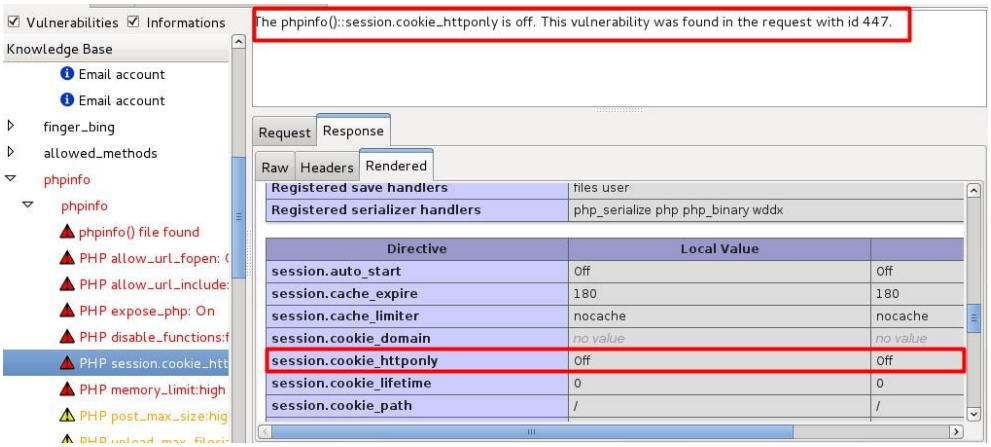
Directive	Local Value	Master Value
session.auto_start	Off	Off
session.cache_expire	180	180
session.cache_limiter	nocache	nocache
session.cookie_domain	*no value*	*no value*
session.cookie_httponly	Off	Off
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.cookie_secure	Off	Off
session.entropy_file	*no value*	*no value*
session.entropy_length	0	0
session.gc_divisor	1000	1000
session.gc_maxlifetime	1440	1440

Fuente: El autor.

Recomendaciones. El valor de esta propiedad debería contener un valor diferente a "/". Para cada dominio alojado en el servidor se debe especificar la ruta de dicho dominio.

- **MSTG-56. Cookie Sin HttpOnly habilitado.** Las pruebas también reflejaron que la propiedad HttpOnly se encuentra en off. Esto indica que ciertos tipos de ataques del lado del cliente, como XSS, podrían capturar y/o modificar de forma trivial los valores de las cookies utilizando lenguajes del lenguaje del cliente, como JavaScript.

Figura 117. Cookie sin HttpOnly habilitado.

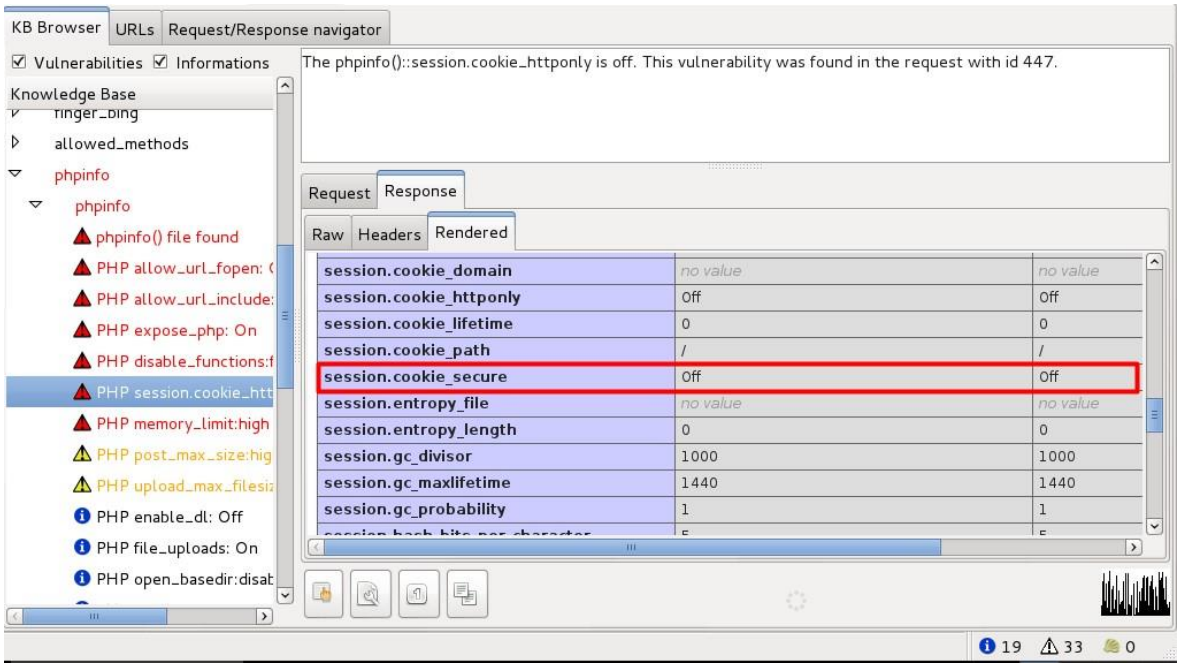


Fuente: El autor.

Recomendaciones. Normalmente no existe una buena razón para no usar la bandera HttpOnly en todas las cookies. A menos que se requiera específicamente que scripts del lado del cliente puedan acceder a las cookies de los usuarios, siempre se debería incluir la cabecera HttpOnly en las cookies.

- **MSTG-57. Propiedad o atributo "Secure" no establecido en Cookies.** El atributo Secure le dice a un navegador web o cliente HTTP que sólo enviará cookies a través de una conexión segura, utilizando HTTPS. Pero si se encuentra deshabilitado, las cookies se enviarán en texto claro a través de HTTP, lo cual implica un riesgo de seguridad. En la siguiente figura se evidencia que en el servidor esta propiedad está deshabilitada, lo que implica una vulnerabilidad, que puede ser aprovechada por un atacante para obtener información de las cookies de un usuario autenticado y hacerse pasar por este último.

Figura 118. Propiedad o atributo “Secure” no habilitado en cookies.

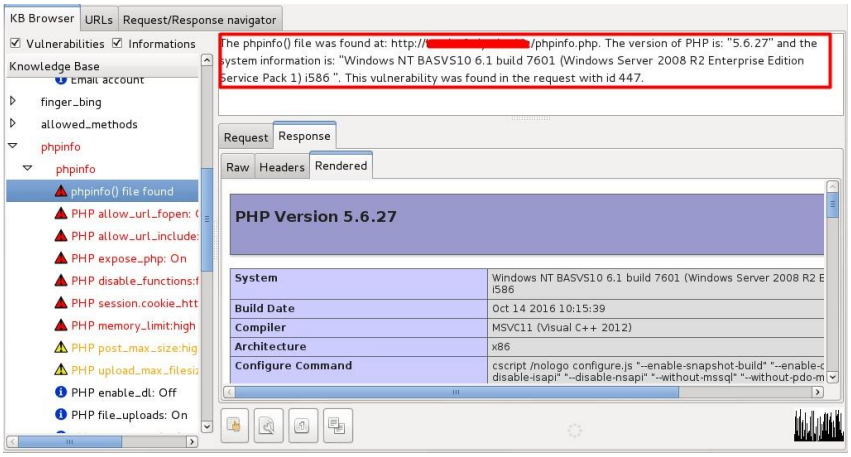


Fuente: El autor.

Recomendaciones. Luego de implementar un certificado SSL en el servidor web, el atributo “Secure” debe ser habilitado en el servidor. De lo contrario, un navegador web u otro cliente HTTP, podría pasar tokens de sesión por un canal de comunicaciones no seguro, usando el protocolo HTTP.

- **MSTG-58. Es posible hacer “fingerprinting” del Servidor y/o el Sistema Operativo.** Finalmente, se pudo evidenciar también que es posible obtener información de la configuración del servidor, aprovechando el hecho de que en el servidor web se encuentran varios archivos que contienen la función phpinfo() de PHP.

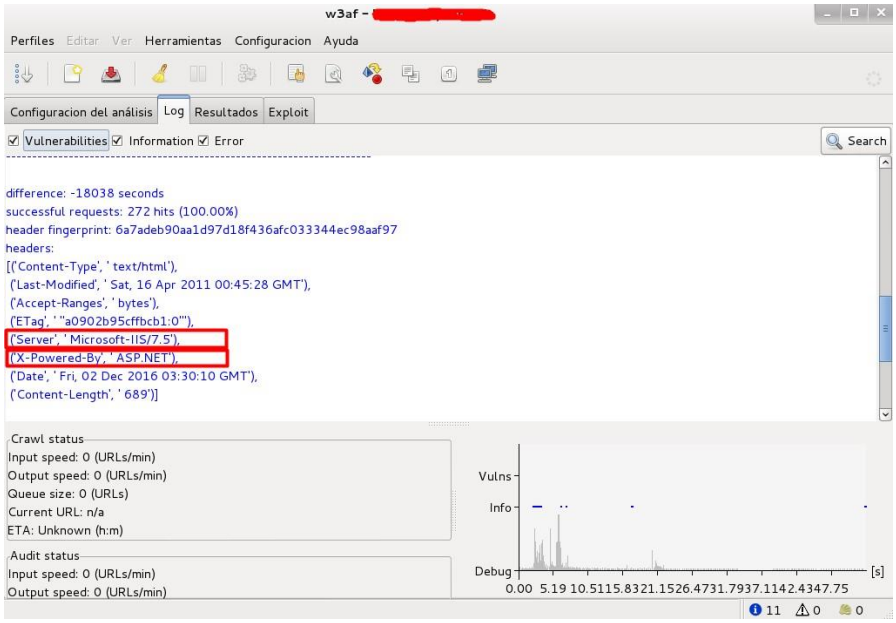
Figura 119. Fingerprinting del servidor mediante phpinfo.



Fuente: El autor.

Mediante un escaneo automático, también se logra obtener información del servidor:

Figura 120. Fingerprinting del servidor mediante escaneo automático.



Fuente: El autor.

Una manera mucho más simple de detectar el tipo de servidor web fue acceder a la raíz del dominio, en el que se muestra el mensaje de bienvenida establecida por defecto:

Figura 121. Fingerprinting del servidor por mensaje de bienvenida.

Fuente: El autor.

Recomendaciones. Cuando un atacante planea llevar a cabo un atacante contra un servidor web, lo primero que hace es tratar de recopilar la mayor cantidad de información de su objetivo, con el fin de poder encaminar su ataque, identificar las posibles vulnerabilidades que se podrían encontrar en el servidor, y seleccionar las herramientas que considere necesarias. Algunas de las acciones que pueden dificultar esta labor son las siguientes:

- Ofuscar o remover las cabeceras en las que se brinde información del servidor y su configuración:
    - Server
    - Powered-by
    - X-AspNet-Version,
    - etc.

- Cambiar las páginas de bienvenida, de errores, y/o para los diferentes códigos de respuesta (500, 404, 413, etc.) proporcionadas por defecto por el servidor web IIS.

- Ubicar el servidor web detrás de sistemas de protección de red, como:

    - Firewall
    - Servidor proxy inverso
    - IPS: Sistema de prevención de intrusiones
    - IDS: Sistema de detección de intrusiones.

Finalmente, se recomienda eliminar cualquier archivo PHP que contenga la función phpinfo().

Anexo B. Carta de aprobación para realizar análisis de seguridad.

15 de Noviembre de 2016

Autorización Análisis de Seguridad de Aplicación Prism For Android y el sistema de Backend relacionado.

Por medio de la presente la empresa **Barcode Aps** autoriza al señor **DAYCITH JOSÉ GONZÁLEZ RODRÍGUEZ** la ejecución de pruebas de seguridad informática tipo Ethical hacking, sobre la aplicación Prism For Android y el sistema de Backend asociado, bajo las siguientes condiciones:

1. **BARCODE APPS** manifiesta su conocimiento y aprobación de las pruebas de penetración de tipo Ethical Hacking a la aplicación PRISM For Android y el Sistema de Backend asociado a dicha aplicación. Dichas pruebas se realizarán durante el periodo comprendido entre el **15 de Noviembre de 2016 y 30 de Abril de 2017.**

2. **BARCODE APPS** aprueba en su totalidad el plan de trabajo presentado por **DAYCITH JOSÉ GONZÁLEZ RODRÍGUEZ**, el cual se adjunta a este documento.

3. **BARCODE APPS** faculta a **DAYCITH JOSÉ GONZÁLEZ RODRÍGUEZ** para explotar vulnerabilidades que puedan permitir el acceso a los sistemas de información. De la misma forma, se reconoce que **DAYCITH JOSÉ GONZÁLEZ RODRÍGUEZ** para la realización de las pruebas, podría crear o adquirir software considerado malicioso o espía.

4. **BARCODE APPS** reconoce que la ejecución de las pruebas de seguridad por parte de **DAYCITH JOSÉ GONZÁLEZ RODRÍGUEZ** tienen completa autorización y por lo tanto, no se está incumpliendo ninguna normatividad o ley de delitos informáticos vigentes.

Anexo B. (Continuación)

5. **DAYCITH JOSÉ GONZÁLEZ RODRÍGUEZ** realizará las pruebas de seguridad mediante técnicas de mínimo impacto sobre la operación de la plataforma tecnológica, sin afectar la integridad, confidencialidad o disponibilidad de la información que no sea de prueba.

6. **BARCODE APPS** habilitará un servidor de pruebas con características similares a los de producción, y así mismo creará usuarios y datos de pruebas con el fin de ser utilizados y manipulados en las pruebas realizadas.

Cordialmente,

Mauricio Franco.
Vicepresidente