CONSTRUCCIÓN DE UN APLICATIVO CLIENTE SERVIDOR PARA LA DETECCIÓN DE VULNERABILIDADES DE RED EN SMARTPHONES ANDROID UTILIZANDO UNA HERRAMIENTA DE ESCANEO

DIRECTOR:
DAVID ANTONIO FRANCO BORRÉ (Msc)

INVESTIGADORES:
ÁLVARO PAYARES GUZMÁN
BRAYAN ORTEGA GARCÍA

UNIVERSIDAD DE CARTAGENA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
CARTAGENA DE INDIAS, 2015.

CONSTRUCCIÓN DE UN APLICATIVO CLIENTE SERVIDOR PARA LA DETECCIÓN DE VULNERABILIDADES DE RED EN SMARTPHONES ANDROID UTILIZANDO UNA HERRAMIENTA DE ESCANEO

PROYECTO DE DESARROLLO

PROYECTO DE GRADO PRESENTADO COMO REQUISITO PARCIAL PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

DIRECTOR:

DAVID ANTONIO FRANCO BORRÉ (Msc)

INVESTIGADORES:

ÁLVARO PAYARES GUZMÁN

BRAYAN ORTEGA GARCÍA

UNIVERSIDAD DE CARTAGENA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

CARTAGENA DE INDIAS, 2015.

Tesis de Grado: CONSTRUCCIÓN DE UN APLICATIVO CLIENTE SERVIDOR PARA LA DETECCIÓN DE VULNERABILIDADES EN SMARTPHONES ANDROID UTILIZANDO UNA HERRAMIENTA DE ESCANEO.

Investigador: ÁLVARO PAYARES GUZMÁN, BRAYAN ORTEGA GARCÍA.

Director: DAVID ANTONIO FRANCO BORRÉ.

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

Cartagena de Indias, Noviembre del 2015

RESUMEN

El presente trabajo de grado se desarrolló para implementar un aplicativo para la detección de vulnerabilidades en dispositivos móviles con sistema operativo Android. El objetivo del trabajo es desarrollar un aplicativo cliente-servidor para la detección de vulnerabilidades en smartphones Android utilizando una herramienta de escaneo. Para lograr lo anterior, se utilizó la metodología RUP (Rational Unified Process), la cual provee un entorno de desarrollo basado en estándares que aseguran la creación de software de alta calidad, con tiempo y presupuesto predecibles. Además, se especificaron los requerimientos funcionales del sistema para elaborar el diseño de la arquitectura del mismo. Por último, se realizaron pruebas del sistema para verificar su funcionamiento en diferentes escenarios, lo que contribuyó a la redacción de un informe que contiene resultados obtenidos que especifican las causas y consecuencias de una vulnerabilidad detectada.

Uno de los resultados importantes de este proyecto es el aplicativo mencionado anteriormente, buscando con ello mostrar que tan vulnerable es un dispositivo móvil Android y las posibles consecuencias que se pueden generar. Otro resultado es el reporte generado por la aplicación cuando detecta vulnerabilidades, pues detalla el origen de la vulnerabilidad con la factible razón de su causa.

Gracias a la investigación y análisis realizados en el presente trabajo de grado, se presenta como conclusión más relevante que la telefonía móvil ha evolucionado rápido estos últimos años y el uso de dispositivos Android dentro de empresas se extiende rápidamente y es importante contar con una herramienta que aporte seguridad a la información, tal como lo hace el aplicativo desarrollado, que no requirió de grandes inversiones para su desarrollo debido a que fue realizado con software libre. Además, desde el punto de vista científico y tecnológico, es viable debido que aporta en el área de seguridad informática minimización de riesgos en smartphones Android, utilizando tecnologías disponibles en el mercado.

Palabras Claves: Vulnerabilidad, Sistema operativo Android, Dispositivos móviles, Aplicativo cliente-servidor, Herramienta de escaneo, Software libre.

ABSTRACT

This degree work was developed to implement an application to detect vulnerabilities on mobile devices with Android operating system. The aim of this work is to develop a client-server application to detect vulnerabilities in Android smartphones using a scan tool. To achieve this, the RUP (Rational Unified Process) was used, which provides a development environment based on standards that ensure the creation of high quality software with predictable time and budget. In addition, the functional requirements of the system is specified to make the design of the processor architecture. Finally, system tests were conducted to verify their performance in different scenarios, which contributed to the drafting of a report containing results that specify the causes and consequences of a detected vulnerability.

One of the important results of this project is the application mentioned above, seeking thereby to show how vulnerable is an Android mobile device and the possible consequences that can arise. Another result is the report generated by the application when it detects vulnerabilities, detailing the origin because of the vulnerability to the likely reason for their cause.

Through research and analysis conducted in this paper grade is presented as the most important conclusion that mobile telephony has rapidly evolved in recent years and the use of Android devices within companies is expanding rapidly and it is important to have a tool that contribution to information security, as does the developed application, which did not require large investments for development because it was done with free software. Moreover, from the point of view of science and technology, it is feasible because it brings in the area of computer security risk minimization in Android smartphones, using available technologies on the market.

Keywords: Vulnerability, Android operating system, mobile devices, client-server software, scanning tool, Free Software.

DEDICATORIA

A mis padres por su apoyo, comprensión, paciencia y sacrificios realizados para ayudarme a cumplir este logro que es tan mío como de ellos.

AGRADECIMIENTOS

Agradecer a Dios por darnos el conocimiento y voluntad para cumplir nuestros objetivos.

El mayor agradecimiento es para nuestros padres, que fueron el motor y gran motivación para cumplir nuestras metas, que juntos con nuestros hermanos han sido el mayor apoyo que hemos contado en todas las etapas de nuestra vida.

Gracias a nuestros amigos y compañeros que en algún momento hicieron parte de esta etapa universitaria, su compañía, motivación e interés en cumplir nuestros logros, fueron y son invaluables.

Sin el asesoramiento del profesor Humberto Caicedo este proyecto no hubiera sido posible, gracias por su tiempo y conocimiento compartido.

Especialmente agradecer al ingeniero David Antonio Franco Borré, por su tutoría, paciencia e iniciativa en el desarrollo de la investigación.

Agradecer al cuerpo docente del programa de Ingeniera de Sistemas por su labor la cual se ve reflejada en cada una de estas páginas.

TABLA DE CONTENIDO

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

INTRODUCCIÓN

Hoy día, los dispositivos móviles son utilizados para acceder a servicios corporativos, tales como ver datos de la empresa, hacer negocios, acceder a redes sociales y otros servicios personales. Sin embargo, muchos de estos dispositivos, por no mencionar la gran mayoría, no están controlados por un administrador con conocimientos de seguridad, lo que implica que datos personales del usuario o datos corporativos de su empresa estén expuestos a diversos problemas de seguridad [4]. Con la evolución de la tecnología de la información y las comunicaciones, las empresas quieren ir a la vanguardia y desean tener la mejor tecnología que se encuentre en el medio, por lo tanto, se han visto en la necesidad de utilizar dispositivos móviles con sistema operativo Android para mejor sus sistemas de comunicación y confidencialidad en la transmisión y recepción de información.

La descarga de software para dispositivos móviles por medio de la red se ha convertido en un hobbie o en una necesidad para los usuarios, sea para bien de una organización o para bien personal. Como consecuencia de las descargas, se hace difícil el control de quiénes crean el software y de quiénes los modifican, promoviendo a que los usuarios no tengan idea de qué tipo de software será instalado, ni qué privilegios puedan ser asignados para realizar cambios en la configuración del dispositivo.

La detección de vulnerabilidades es un tema común en el campo de telecomunicaciones y seguridad de la información. A nivel local, se ha investigado y desarrollado poco en el tema, como antecedente más relevante cabe destacar la herramienta para detectar vulnerabilidades desarrollada por un egresado de Ingeniería de Sistemas de la Universidad de Cartagena, en el cual se diseña un nuevo paradigma para detectar y evaluar vulnerabilidades en una red de datos, utilizando la técnica de identificación de servicios. Sin embargo, dicho trabajo no está enfocado para dispositivos móviles con sistema operativo Android sino a una red de datos en general. A lo anterior se le suma que el software sólo se ejecuta en equipos con sistema operativo Windows, lo que excluye a dispositivos móviles. Razones por las cuales se ha desarrollado este proyecto de investigación y desarrollo.

El objetivo fue crear un aplicativo móvil con una herramienta de escaneo para identificar vulnerabilidades en la red donde se encuentran conectados dispositivos móviles con sistema operativo Android. Además, será un aplicativo cliente-servidor que se podrá descargar desde la tienda virtual, originando un gran impacto a nivel local, regional e internacional, convirtiendo al proyecto en novedoso, al no haber otros proyectos con las mismas características en la ciudad.

Para el cumplimiento del objetivo de este trabajo se realizó el estado del arte sobre los problemas de seguridad en smartphones Android y de herramientas de escaneo de vulnerabilidades, logrando con ello ampliar la literatura existente referente al tema. Además, se identificaron los requerimientos funcionales para diseñar e implementar un componente arquitectónico que permitiera desarrollar el aplicativo de detección de vulnerabilidades, que fue sometido a pruebas de funcionamiento para capturar resultados que contribuyan a la seguridad del dispositivo móvil. Una vez cumplido con el objetivo de este trabajo se pudo dar respuesta a la pregunta ¿Cómo construir un aplicativo cliente servidor para detectar y minimizar vulnerabilidades en smartphones Android?

La importancia del estudio se ve en el aporte al campo de seguridad informática ya que se desarrolló un aplicativo que contribuye a proteger la información en uno de los sistemas operativos más reconocidos mundialmente. Además de agregar características de adaptabilidad y fácil uso, busca promover que profesionales en el tema se interesen por añadir mejoras al producto teniendo en cuenta otros factores que influyen en el proceso de detección de vulnerabilidades.

Es importante resaltar que el proyecto ha sido desarrollado en la ciudad de Cartagena de Indias durante los años 2014 y 2015 en el laboratorio de ingeniería de software de la Universidad de Cartagena.

1. DESCRIPCIÓN DEL PROYECTO

1.1 Planteamiento del problema

La telefonía móvil es una de las tecnologías que ha evolucionado con mayor rapidez durante estos últimos años y esto se debe a que cada día las personas alrededor del mundo han buscado romper las distancias que los separan entre sí, además la portabilidad que poseen estos dispositivos le permiten a un individuo estar siempre conectado con sus intereses sin importar el lugar donde este se encuentre. Las ventajas que ofrece la telefonía móvil han sido aprovechadas por las empresas a nivel mundial, que como lo evidencia [1] el uso de los dispositivos Android dentro de la empresa sigue extendiéndose rápidamente, a la medida que los empleados exigen que los departamentos de tecnologías de la información ofrezcan acceso a los recursos de la empresa tales como correo electrónico, software para la gestión de clientes u otros datos de la empresa indispensables para su buen funcionamiento, ofreciendo de este modo soluciones empresariales para quiénes implementen esta tecnología en su entorno de trabajo.

Ahora bien, al mismo ritmo que crecen las tecnologías de la información y de las comunicaciones, también lo hacen los ataques a los servicios de red y las técnicas de plagio de datos e información, con consecuencias cada vez más graves. Los dispositivos Android no son inmune a estos hechos y pueden llegar a ser presas fáciles de los atacantes, quienes a través de aplicaciones que circulan por la internet, pueden filtrar sus aplicaciones maliciosas en diversos programas que las personas suelen descargar para sus móviles, las cuales en apariencia pueden parecer inofensivas, pero que pueden resultar ser un software espía que por medio de vulnerabilidades, resulten plagiando información importante de una empresa o persona.

Debido a que esta plataforma tecnológica es la de mayor crecimiento a nivel empresarial, según informe de la firma Ipass en 2011 [2], se hace necesario detectar

vulnerabilidades a las que están expuestos las redes donde normalmente funcionan estos dispositivos, los usuarios y empresas.

Teniendo en cuenta lo expuesto anteriormente se requiere desarrollar un aplicativo para dispositivos móviles con sistema operativo Android, con el cual se puedan detectar cualquier tipo de vulnerabilidades de los servicios de red que estén presentando estos dispositivos, y de este modo minimizar los riesgos de ser víctimas de delitos informáticos.

1.2 Formulación del problema

¿Cómo construir un aplicativo cliente servidor para detectar y minimizar vulnerabilidades en smartphones Android, utilizando una herramienta de escaneo para mejorar la seguridad de estos dispositivos?

1.3 Justificación

A consecuencia de que los dispositivos móviles necesitan de software para ser de mayor utilidad, se hace necesario descargar aplicaciones en páginas de poca confianza, ya que existen personas u organizaciones dedicadas a construir software malicioso para el hurto de información basándose en las vulnerabilidades que hay en las redes de estos dispositivos, que cada día son más y difíciles de detectar.

Otro aspecto muy importante que hay que tener en cuenta es que las TICs se encuentran estrechamente ligadas a la cotidianidad de las personas, independientemente del entorno en el que se encuentren y la tendencia de estas tecnologías, se va enfocando o está convergiendo cada vez más a dispositivos con menor tamaño, menor peso, mayor portabilidad y que debido a esto no pierda capacidad relacionada con el procesamiento de la información.

Actualmente la descarga de software para estos dispositivos móviles por medio de la red se ha convertido en hobbies para muchos y en una necesidad para otros, sea para bien de una organización o para bien personal. Como consecuencia de las grandes descargas, se hace difícil el control de quienes crean el software y de quienes los modifican, implicando que usuarios no tengan idea de qué tipo de software será instalado, ni que

privilegios puedan ser asignados para realizar cambios en la configuración del dispositivo.

La importancia de este proyecto, además de la detección de las vulnerabilidades en smartphones Android, radica en que será un aplicativo cliente-servidor que podrá ser descargado desde la tienda virtual Google Play Store, originando un gran impacto a nivel local, regional e internacional, convirtiendo al proyecto en novedoso, al no haber otros proyectos con las mismas características.

El desarrollo de este proyecto es complejo para los autores debido a que deben realizar un estudio de las herramientas de escaneo de vulnerabilidades del mercado para su posterior integración al aplicativo cliente-servidor que se pretende desarrollar, se aplicarán los conocimientos adquiridos y herramientas manejadas en la carrera universitaria como ingenieros de sistemas, vistos en asignaturas como: algoritmos, programación básica, programación orientada a objetos, ingeniería de servicios web, ingeniería de software, base de datos, sistemas operativos, redes, seguridad informática, seguridad en redes, entre otras.

2. MARCO DE REFERENCIA

2.1 ESTADO DEL ARTE

Con el pasar del tiempo se han evidenciado tendencias hacia el desarrollo de malware para dispositivos móviles. Un ejemplo concreto es Cabir, un gusano que nació como una prueba de concepto durante el año 2004, desarrollado para sistemas Symbian. Dicho gusano era capaz de reproducirse a través de conexiones del tipo Bluetooth y fue el primer malware creado para explotar estas plataformas.

Otros códigos maliciosos diseñados para dispositivos telefónicos de alta gama fueron Skull (2004), que remplazaba la imagen original de los íconos por la imagen de una calavera; CommWarrior (2005), cuya propagación se llevaba a cabo a través de mensajes MMS y tarjetas de memoria; Doomboot (2005), que simula ser el juego Doom 2 pero instala una variante de Cabir y otra de CommWarrior y RedBrowser (2006), que enviaba mensajes de texto a un número telefónico de Rusia.

A mediados de 2009, la aparición de un gusano conocido como SexyView, cuya propagación se llevaba a cabo a través de mensajes del tipo SMS que direccionaban hacia un sitio web malicioso, no hizo más que demostrar que el desarrollo de amenazas informáticas para estas plataformas continúa en aumento [3].

Para el caso de los dispositivos Android, el primer malware detectado fue el conocido como Droid09, apareció en noviembre de 2009 en el Android Market. Pretendía ser una aplicación para la gestión de algunos bancos online, ofreciendo al usuario un listado bastante limitado y solicitando las credenciales posteriormente. Desde la percepción del usuario, no era de extrañar que una aplicación bancaria pidiese tales datos, aunque el destino de sus credenciales no fuera en absoluto legítimo. La aplicación fue borrada inmediatamente de Android Market., tienda virtual de Google hoy llamada Google Play.

Otro tipo de malware detectado a finales de 2010 para esta plataforma es Android.Fakeplayer, que pretendiendo ser una aplicación para reproducir vídeos, era en realidad un código malicioso destinado a enviar mensajes SMS Premium, el vector de

infección más popular para los terminales móviles es el uso de aplicaciones modificadas para realizar alguna acción irregular. De esta forma, los propios usuarios de los terminales móviles infectan sus teléfonos al instalarse dichas aplicaciones desde su mercado de distribución [4].

Android ha sido la plataforma más atacada debido al crecimiento de usuarios que la utilizan. Según informes de Gartner, empresa dedicada a la investigación del mercado de las nuevas Tecnologías de la Información, el sistema operativo es líder de plataformas móviles desde mediados de 2011. Posee más de 400 millones de dispositivos que lo utilizan en todo el mundo y crece en promedio 550 mil dispositivos por día. [5] Además según McAfee, en el sistema operativo móvil Android se convirtió en la plataforma más popular para el nuevo malware según ellos, Android se ha convertido casi en la plataforma exclusiva de todo el malware nuevo para dispositivos móviles [6]. La siguiente ilustración muestra el aumento de malware diseñados para dispositivos móviles Android.

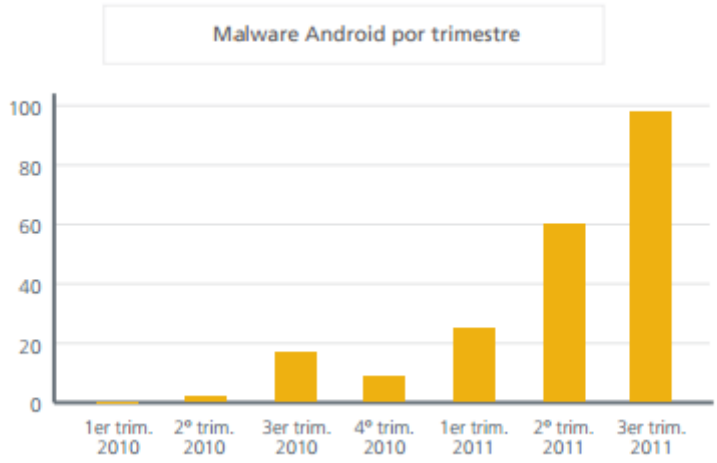


Ilustración 1. Gráfica lineal: Propagación de malware en dispositivos móviles Android.

A raíz de estas vulnerabilidades que se han estado presentando, ha sido indispensable crear soluciones de seguridad para estos dispositivos. Entre las que existen actualmente se pueden mencionar los antivirus, navegadores web seguros y MDM (Mobile Device Management).

Ya existe una serie de escáneres antivirus de primera generación para la plataforma Android. Estos escáneres son efectivos en la detección de amenazas conocidas Android, pero ofrecen poca protección contra amenazas desconocidas. El uso de antivirus como solución al problema de seguridad, sólo resuelve las amenazas en la categoría de malware, así como un subconjunto de malware compuesto por ataques basados en el abuso de recursos y la pérdida de datos. Hay que destacar que el uso de un antivirus generalmente viene de la mano con un gran gasto de recursos, por lo tanto, varias empresas han presentado sus propias aplicaciones de navegadores Web seguros. Cada vez que el usuario visita una URL mediante el navegador seguro, el navegador comprueba la dirección URL en una base de datos o lista negra y bloquea las páginas maliciosas. El único problema con este tipo de navegador seguro es que el usuario no puede utilizar el navegador que viene instalado por defecto en el dispositivo. En su lugar, debe utilizar el navegador Web seguro para realizas toda la navegación. De los seis diferentes tipos de amenazas que afrontan los dispositivos móviles, los navegadores seguros pueden tratar con eficacia los ataques basados en Web y los ataques de ingeniería social. También puede bloquear la introducción de malware descargado a través del navegador.

Las soluciones móviles de gestión de dispositivos (MDM) no protegen de manera específica de las amenazas, sí que pueden ayudar a reducir el riesgo de un ataque. Por ejemplo, si el administrador utiliza la solución MDM para configurar el dispositivo y bloquear la introducción de todas las nuevas aplicaciones, esto puede eliminar la introducción de nuevos programas maliciosos, así como limitar el abuso de los recursos, las amenazas de la integridad, y una posible pérdida de datos [4].

Muchas empresas y desarrolladores particulares han empezado a crear aplicaciones para mitigar el problema de seguridad en los dispositivos móviles Android, es por esto que IBM ha puesto en marcha su plan de acción en contra de estas amenazas expuesto en el Informe IBM X-Force de Tendencias y Riesgos 2011, en dicho informe, Stuart Dross vicepresidente de Ventas y Marketing de Cigital, Inc. Señalo: “La capacidad de escanear aplicaciones móviles nativas e híbridas para detectar vulnerabilidades de seguridad es un importante paso adelante para asegurar los datos sensibles y mitigar los riesgos de seguridad”. IBM resaltó que las aplicaciones móviles representan un nuevo

blanco de amenazas, ya que conllevan un riesgo más alto de ataque, en comparación con las vulnerabilidades de las aplicaciones web [7].

La empresa DuoSecurity también se ha puesto en la tarea de solucionar problemas de seguridad en teléfonos Android y por esta razón ha desarrollado una aplicación llamada X-Ray que se encarga de escanear el dispositivo para determinar si existen vulnerabilidades que permanecen sin parchear. La aplicación tiene una lista de vulnerabilidades que es capaz de identificar y permite comprobar la presencia de cada punto vulnerable en su dispositivo. La ventaja de esta aplicación con respecto a los antivirus es que no consume recursos, ya que estos últimos suelen contener un sistema de escaneo en tiempo real, que la mayoría de las veces no es de gran ayuda [8]. X-Ray no identifica malware, solo agujeros de seguridad que tampoco se encarga de corregirlos sino únicamente se limita a informar y recomendar utilizar una variante de Android más segura.

Las aplicaciones dedicadas a la seguridad en los dispositivos móviles Android son escasas y algunas poco óptimas, es por esta razón que hay que indagar más a fondo en las posibles soluciones a esta problemática que continuará en constante crecimiento sino se le da un tratamiento adecuado

Según [10] entre todo el malware móvil existente, el porcentaje de malware basado en Android es superior a 46% y sigue creciendo rápidamente. También se alerta de que hay un 400% de aumento de malware basado en Android desde el verano de 2010, debido a esto el estudio de estos autores que tuvo como objetivo clasificar el malware existente en esta plataforma. Lograron recoger más de 1.200 muestras de malware que cubren la mayoría de las actuales familias de malware Android, que van desde su debut en agosto de 2010 hasta los más recientes en octubre de 2011. El estudio muestra que entre los ejemplares de malware que se recogieron el 86,0% de ellos son versiones reempaquetadas dentro de aplicaciones legítimas, lo que conlleva a la necesidad de vigilar y detectar las aplicaciones reempaquetadas en los mercados actuales de Android.

Por otro lado, se observa que las más recientes familias de malware Android están adoptando el sistema de actualización para infectar a los usuarios, y ser más sigilosos y

difíciles de detectar. Además, cuando se realiza el análisis respectivo, se muestran una serie de estadísticas alarmantes:

1. Alrededor de un 36,7% de los ejemplares de malware recogidos poseen exploits para comprometer completamente la seguridad Android, lo que representa un alto nivel de las amenazas a la seguridad de los usuarios y su privacidad.
2. Más del 90% controla los teléfonos a través de la red.
3. 45,3% de las muestras se dedican a hacer llamadas telefónicas sin el conocimiento del usuario.

El estudio también revela que estos malware están evolucionando rápidamente para eludir la detección anti-virus para móviles. Los cuales, Basándose en experimentos, muestran que en el mejor de los casos detectan el 79,6% de ellos, mientras que el peor de los casos sólo detecta 20,2% del malware contenido en una base de datos. Estos resultados requieren claramente la necesidad de desarrollar una solución para este problema. A continuación se muestra en la tabla los resultados de este estudio [10].

	Installation				Activation								
	Repackaging	Update	Drive-by Download	Standalone	BOOT	SMS	NET	CALL	USB	PKG	BATT	SYS	MAIN
ADRD	√				√		√	√					
AnserverBot	√	√			√	√	√		√		√	√	
Asroot				√									
BaseBridge	√	√			√	√	√				√	√	
BeanBot	√					√		√					
BgServ	√				√	√							√
CoinPirate	√				√	√							
Crusewin				√	√	√							
DogWars	√												
DroidCoupon	√				√		√	√		√			
DroidDeluxe				√									
DroidDream	√												√
DroidDreamLight	√				√			√					
DroidKungFu1	√				√						√	√	
DroidKungFu2	√				√						√	√	
DroidKungFu3	√				√						√	√	
DroidKungFu4	√				√						√	√	
DroidKungFuSapp	√				√						√	√	
DroidKungFuUpdate	√	√											
Endofday	√				√	√							
FakeNetflix				√									
FakePlayer				√									
GamblerSMS				√	√								
Geinimi	√				√	√							
GGTracker			√	√	√	√					√		
GingerMaster	√				√								
GoldDream	√			√	√	√		√					
Gone60				√									
GPSSMSSpy				√		√							
HippoSMS	√				√	√							√
Jifake	√		√										
jSMSHider	√									√			√
KMin				√	√								
Lovetrap				√	√	√							
NickyBot				√	√	√							
Nickyspy				√	√								
Pjapps	√				√	√						√	
Plankton		√		√									
RogueLemon				√		√							
RogueSPPush				√		√							
SMSReplicator				√		√							
SndApps				√	√								
Spitmo			√	√		√		√					
TapSnake				√	√								
Walkinwat				√									
YZHC				√	√								
zHash				√	√								
Zitmo			√	√		√							
Zsone	√					√							√
number of families	*25*	*4*	*4*	*25*	*29*	*21*	*4*	*6*	*1*	*2*	*8*	*8*	*5*
number of samples	*1083*	*85*	*4*	*177*	*1050*	*398*	*288*	*112*	*187*	*17*	*725*	*782*	*56*

Tabla 1. Resultado de estudios. Tomado de [10]

Según [11] los usuarios Android tienen la capacidad de permitir o negar una solicitud para comunicarse con otras aplicaciones a través de la comunicación entre procesos (CEP). Si el usuario concede este permiso, las aplicaciones de malware pueden ser capaces de comprometer el sistema. Si la aplicación solicita este permiso y el usuario lo niega, la aplicación posiblemente no se instalará. El sistema operativo Android proporciona un rico conjunto de las capacidades de la CEP, pero por desgracia hay muchos problemas de seguridad en el sistema operativo.

Dado que en los teléfonos Android están limitadas la potencia de procesamiento, memoria y velocidad, y hay algunas restricciones adicionales en el funcionamiento de fuertes defensas de seguridad tales como antivirus y programas de firewall. Las

aplicaciones por default poseen los permisos de seguridad del usuario del teléfono y tienen los siguientes tipos de aplicativos: actividades, servicios y, proveedores de contenidos y receptores de radiodifusión. En adición, un servicio en Android se ejecuta en segundo plano y otros componentes de varias aplicaciones diferentes pueden interactuar intercambiando datos y servicios entre sí. Cuando un servicio es instalado, el desarrollador tiene la opción de poner el servicio a disposición de las aplicaciones, Para ello, la instalación del archivo de manifiesto debe incluir una declaración de permiso donde se nombren los servicios compartidos (por ejemplo Súper servicio). Y Si un usuario instala una aplicación, que requiere el uso de Súper servicio, la aplicación, debe solicitar el acceso al Súper servicio a través de su archivo de manifiesto con el que se instaló. Con lo cual se generan tres problemas relacionados con la estrategia de permisos que son:

1) Las opciones de instalación son: lo tomas o lo dejas, así que si un usuario quiere evitar que una aplicación tenga acceso a un solo servicio, el usuario no debe instalar la aplicación.

2) El permiso para utilizar súper servicio solo se comprueba en el tiempo de instalación. Para hacer cualquier cambio de permisos, se requiere que el usuario desinstale completamente y reinstale la aplicación que utiliza súper servicio.

3) Es el usuario final quien debe tomar la decisión si una aplicación recién instalada debe permitir utilizar el súper servicio. Los usuarios generalmente no están en la capacidad de saber qué combinaciones de aplicaciones o servicios son utilizados por el malware. Debido a estos tres inconvenientes se planeta como solución implementar un sistema de depuración de aplicaciones para Android almacenado en la nube. Para evitar que las aplicaciones que se instalan contengan malware, con el cual los hackers pueden realizar el seguimiento de los movimientos y la ubicación del propietario del teléfono.

Los riesgos para Android van en aumento, sobre todo porque es un sistema operativo de código abierto que funciona en un entorno móvil heterogéneo. Por lo tanto, hackers pueden acceder fácilmente para manipular y explotar el código del sistema operativo. Cada vez que un atacante descubre un error o vulnerabilidad en un componente central, él puede ser capaz de ejecutar código malicioso en modo privilegiado e incluso tomar el

control completo sobre el dispositivo. Esta amenaza se amplifica debido a que algunos procesos del sistema se ejecutan con privilegios de root y no existe un mecanismo de control para los procesos del sistema Android. Uno de los problemas comunes es el mecanismo de permisos, el cual no está suficientemente protegido, y la instalación de una aplicación maliciosa que utiliza permisos por parte de un usuario desprevenido es un escenario probable. Además, el mecanismo de identificación de usuarios compartidos otorga permisos sin conocimiento del usuario.

Otra vulnerabilidad es la relacionada con inyección de código malicioso a través de un navegador Web. Algunos ataques recientes incluyen desbordamiento de búfer en una biblioteca nativa y en sitios explícitos con scripting (XSS). Ambos ataques permiten que un atacante con habilidades ejecute código malicioso en el dispositivo por medio de los privilegios asignados al navegador. Otra amenaza es la que se produce cuando una aplicación se aprovecha de una vulnerabilidad en el kernel de Linux o el sistema de bibliotecas, lo que compromete la disponibilidad, confidencialidad e integridad. A pesar de esta multitud de amenazas, la probabilidad de ocurrencia es baja, sin embargo, como amenaza podría causar graves daños.

A continuación se mostrara una clasificación de efectos que se producen cuando se explota una vulnerabilidad:

1. El abuso de costosos servicios y funciones (por ejemplo, el servicio de envío de mensajes cortos / servicio de mensajería multimedia (SMS / MMS), hacer llamadas telefónicas, o redirigirlas) por la explotación de forma remota en un componente central que se expone en Internet.

2. La actividad maliciosa en contra de un dispositivo de red o de red (por ejemplo, el envío de spam, infectar a otros dispositivos, escáner de red) de forma remota: la explotación de una vulnerabilidad en el núcleo componente que está expuesta en Internet.

3. El abuso de costosos servicios y funciones (como el envío de SMS / Mensajes MMS, hacer llamadas telefónicas, o redirigirlas) por una aplicación explotando una vulnerabilidad en un componente central.

4. La actividad maliciosa en contra de un dispositivo de red o de red (por ejemplo, el envío de spam, infectar a otros dispositivos, Sniffing) por una aplicación explotando una vulnerabilidad en un componente de núcleo.

5. El abuso de costosos servicios y funciones (como el envío de SMS / Mensajes MMS, hacer llamadas telefónicas, o redirigirlas) con los permisos otorgados por el propietario en la instalación.

6. La actividad maliciosa en contra de un dispositivo de red o de red (por ejemplo, el envío de spam, infectar a otros dispositivos, Sniffing) por aplicaciones maliciosas con los permisos otorgados por el propietario del teléfono durante la instalación.

7. Desactivación de aplicaciones o del dispositivo de forma remota: la explotación de una vulnerabilidad en un componente central que se expone en Internet

8. Desactivación de aplicaciones o del dispositivo por una aplicación explotando una vulnerabilidad en un componente central.

9. Desactivación de aplicaciones o del dispositivo de forma maliciosa utilizando los permisos concedidos por el propietario de un teléfono durante la instalación de una aplicación.
10. Corromper o modificar el contenido privado, modificar o escuchar la comunicación del dispositivo red (por ejemplo, llamadas telefónicas, comunicación por Internet, correos electrónicos o mensajes SMS / MMS) de forma remota por la explotación de una vulnerabilidad en un componente central que está expuesto en Internet.

11. Corromper o modificar el contenido privado, modificar o escuchar la comunicación del dispositivo red mediante una aplicación explotando una vulnerabilidad en el núcleo componente.

12. Corromper o modificar el contenido privado, modificar o escuchar la comunicación del dispositivo red con los permisos otorgados por el propietario del teléfono durante la instalación de una aplicación.

13. Obtener, corromper o modificar el contenido privado cuando se navega por un sitio web malintencionado.

14. El bloqueo, modificación o espionaje en la red de comunicación del dispositivo cuando está conectado a una red no fiable.

15. Recepción de correo no deseado, mensajes SMS / MMS o correos electrónicos.

16. La pérdida de los componentes de hardware.

17. Provocar un fallo en los componentes de hardware

Debido a lo anterior, en [12] proponen revisar y evaluar los mecanismos de seguridad incorporados en Google Android y si son adecuados a la luz de las reiteradas amenazas para teléfonos inteligentes.

En [13] se refieren al hecho de que cuando se utilizan aplicaciones descargadas de internet, los usuarios de teléfonos inteligentes a menudo deben decidir si deben permitir que las aplicaciones accedan a su información personal, y muchas veces no entienden los detalles que implican dar acceso a una aplicación y el comportamiento de ésta en la red.

Debido a la popularidad de sitios web y aplicaciones para teléfonos inteligentes, se desconoce que muchas de estas acceden y obtienen información personal cuando los usuarios no tienen la notificación de forma personal. Sandbox es una herramienta común utilizada en Smartphone. Dos métodos se utilizan en esta arquitectura, en el primer método un ID de usuario diferente se pueden asignar a una sola aplicación. En el segundo método el archivo de manifiesto de la aplicación se utiliza para la concesión de permisos a la información personal. Este marco puede garantizar la privacidad y seguridad para un teléfono inteligente OS. Por lo tanto, se propone un mecanismo para la plataforma Android basada en los resultados de los Sandbox, donde se extraen las

características de las aplicaciones y se analizan utilizando componentes independientes para determinar el nombre de dominio del Android malware. El mecanismo propuesto puede identificar malware de forma automática.

Para [14], los autores de malware fácilmente extienden su código malicioso mediante descargas de aplicaciones limpias, inyectando código malicioso que luego es transferido a los sitios donde se encuentran alojadas y ser consumidas por usuarios desprevenidos. En la literatura de investigación de malware en ordenadores de escritorio y portátiles, un enfoque importante ha sido el de identificar patrones de comportamiento que luego ayudan a clasificar el malware, con la esperanza de una mejor comprensión de sus objetivos y por lo tanto diseñar contramedidas rápidamente. Este estudio tuvo por objetivo identificar patrones de comportamiento en aplicaciones maliciosas que permitan distinguir una aplicación de otra. Para esto se utilizó DroidBox, que es una herramienta de análisis dinámico de aplicaciones para Android cuya funcionalidad es la de detectar escapes de información y monitorear llamadas invocadas por una aplicación. En DoridBox, se corrieron aplicaciones maliciosas durante un período de tiempo fijo de exactamente 60 segundos para que poder comparar la salida de muestras con mayor facilidad. En ese estudio se comprobó que las aplicaciones podrían presentar salidas de datos y así ser propensos al mal uso por parte de creadores de malware. También comprobaron que Droidbox puede ser muy útil tanto en la clasificación de aplicaciones maliciosas en Android como para determinar las deficiencias en aplicaciones apacibles de Android pero con algunas limitaciones como la no supervisión del código nativo que podría tener fugas de datos que pasan desapercibidos.

Otra de las soluciones que se han planteado para la detección de malware es en [15], que se dirige específicamente al problema de las aplicaciones de la tienda Google Play, que si bien poseen un sistema diseñado para rechazar la subida de aplicaciones de malware, no siempre es posible identificar qué tipo de aplicaciones contienen malware incrustados, ya que por lo general se identifican como malware hasta después de que se ejecuta en los usuarios. También se hace referencia al hecho de que este malware puede conseguir acceso a internet valiéndose de aplicaciones que no son malware, pero que contienen huecos de seguridad en cuanto a su programación y si a esto se le añade que los usuarios para hacer la instalación de sus aplicaciones deben conceder todos los permisos a éstas, para que puedan ser instaladas y los mencionados permisos no se

pueden volver a modificar en tiempo de ejecución, y es exactamente en esta vulnerabilidad donde los diferentes tipos de malware se aprovechan para obtener los mismos permisos y hacer el trabajo para el cual fueron diseñados. La solución que se propone es evaluar el modelo de seguridad y mejorar la arquitectura de Android, también proponen WallDroid, que es un cortafuegos virtualizado. Dicha solución es considerada como una aplicación de Firewall pero con funcionalidades extras. Esta se basa en la nube haciendo el seguimiento de millones de aplicaciones y su reputación con ítems de bueno, malo, o desconocido, y la comparación de los flujos de tráfico de aplicaciones con una lista de conocidos servidores con IP maliciosas.

2.2 MARCO TEÓRICO

2.2.1 Plataforma móvil

Unas de las plataformas móviles más reconocida en el mundo son Blackberry y Android, esta última es un sistema operativo creado inicialmente por ANDROID INC. Nace de la unión de Linux y una plataforma basada en java llamada Dalvik, adquirida años después por Google en el 2005. Este sistema operativo está basado en un kernel de Linux, a diferencia de otras plataformas como IOs o Window Phone.

Android se desarrolla en una licencia abierta, lo que permite acceder con facilidad al código, por ser código abierto. Otro punto a favor del sistema operativo Android es que los fabricantes le han tomado absoluta confianza gracias a que la oferta de los teléfonos con este sistema operativo es mayor en comparación a otros. Los aspectos negativos de los dispositivos móviles Android radican en que este sistema operativo maneja la fragmentación debido a las múltiples versiones y la gran cantidad de hardware.

Android cuenta con un potente sistema de aislamiento para asegurar que las aplicaciones sólo accedan a los recursos del sistema que han sido aprobados. Este sistema de aislamiento no sólo aísla cada aplicación de las otras aplicaciones en el sistema, sino que también impide el acceso de las aplicaciones al kernel del sistema operativo, lo que garantiza que una aplicación maliciosa no puede tener nivel de administrador en el dispositivo. La política de aislamiento por omisión prohíbe el

acceso a prácticamente todos los subsistemas del dispositivo, con las excepciones notables siguientes:

- Las aplicaciones pueden obtener la lista de aplicaciones instaladas en el dispositivo y examinar el código de cada aplicación (pero no sus datos privados).
- Las aplicaciones pueden leer (pero no escribir) el contenido de la tarjeta flash SD del usuario, que normalmente tiene la música del usuario, archivos de vídeo, los programas instalados y, posiblemente documentos o archivos adjuntos guardados.
- Las aplicaciones pueden leer todos los datos de la tarjeta SD sin restricciones (sin importar qué aplicación creó un dato concreto, todas las aplicaciones pueden leer los datos).
- Las aplicaciones pueden lanzar otras aplicaciones en el sistema, como el navegador, la aplicación de mapas, etc. [4].

2.2.2 Características de los dispositivos móviles Android

Los dispositivos móviles Android se caracterizan por:

- Diseño: la plataforma Android es adaptable a diferentes tamaños de pantallas. Emplea Biblioteca de gráficos 2D y biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0.

- Almacenamiento de los datos: realiza mediante SQLite la conectividad de los dispositivos móviles. Android soporta las siguientes tecnologías de conectividad: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE y WiMAX.

- En su mensajería utiliza SMS y MMS como formas de mensajería, incluyendo mensajería de texto y ahora la Android Cloud to Device Messaging Framework (C2DM) es parte del servicio de Push Messaging de Android.

- El navegador web incluido en Android está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript V8 de Google Chrome. El navegador obtiene una puntuación de 93/100 en el test Acid3.

- Ejecuta un manejador java que, aunque las aplicaciones son escritas en este lenguaje, no hay una máquina virtual de Java en la plataforma. El código Java se compila en el ejecutable Dalvik y corre en la máquina virtual Dalvik, siendo este último una máquina virtual especializada y diseñada específicamente para Android, además, es optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados. El soporte para J2ME puede ser agregado mediante aplicaciones de terceros como el J2ME MIDP Runner [4].

2.2.3 Vulnerabilidades en dispositivos móviles

2.2.3.1 Vulnerabilidad.

Una vulnerabilidad es un elemento de un sistema informático que puede ser aprovechado por un atacante para violar la seguridad, así mismo puede causar daños sin tratarse de un ataque intencionado.

A las vulnerabilidades se les consideran un elemento interno del sistema, por lo que es tarea de los administradores y usuarios detectarlas, valorarlas y reducirlas. Además, son el resultado de errores de programación (bugs), fallos en el diseño del sistema, incluso las limitaciones tecnológicas pueden ser aprovechadas por los atacantes.

Globalmente clasificamos las vulnerabilidades en:

2.2.3.1.1 Vulnerabilidades de desbordamiento de búffer.

Se produce cuando un programa no controla la cantidad de datos que se copian en búffer, de tal manera que si esa cantidad es superior a la capacidad del búffer los bytes sobrantes, se almacenan en zonas de memoria adyacentes sobrescribiendo su contenido

original. Se puede aprovechar para ejecutar código que otorgue privilegios de administrador.

2.2.3.1.2 Vulnerabilidades de condición de carrera (*race condition*).

La condición de carrera se da principalmente cuando varios procesos acceden al mismo tiempo a un recurso compartido, por ejemplo una variable, cambiando su estado y obteniendo de esta forma un valor no esperado de la misma.

2.2.3.1.3 Vulnerabilidades de error de formato de cadena (*format string bugs*).

La principal causa de los errores de cadena de formato es aceptar sin validar la entrada de datos proporcionada por el usuario. Es un error de programación y el lenguaje más afectado es C/C++. Un ataque puede conducir de manera inmediata a la ejecución de código arbitrario y revelación de información.

2.2.3.1.4 Vulnerabilidades de Cross Site Scripting (XSS).

Abarcan cualquier ataque que permitiera ejecutar scripts como VBScript o JavaScript en el contexto de otro sitio web. Estos errores se pueden encontrar en cualquier aplicación que tenga como objetivo final presentar la información en un navegador web. Un uso de esta vulnerabilidad es hacer phishing, donde la víctima ve en la barra de direcciones un sitio pero realmente está en otro, ésta introduce su contraseña y se la envía al atacante.

2.2.3.1.5 Vulnerabilidades de Inyección SQL.

Una inyección SQL se produce cuando, de alguna manera, se inserta o "inyecta" código SQL invasor dentro del código SQL programado, a fin de alterar el funcionamiento normal del programa y lograr así que se ejecute la porción de código invasor incrustado en la base de datos.

2.2.3.1.6 Vulnerabilidades de denegación del servicio.

La denegación de servicio provoca que un servicio o recurso sea inaccesible a los usuarios. Normalmente provoca la pérdida de la conectividad de la red por el consumo del ancho de banda de la red de la víctima o sobrecarga de los recursos informáticos del sistema de la víctima.

2.2.3.1.7 Vulnerabilidades de ventanas engañosas (Window Spoofing).

Las ventanas engañosas son aquellas que dicen que eres el ganador de tal o cual cosa, lo cual es mentira y lo único que quieren es que des información. Hay otro tipo de ventanas que si las sigues, obtienen datos del ordenador para luego realizar un ataque.

2.2.3.2 Escáner de Vulnerabilidades.

Conocido también como análisis de puertos o analizador de red. Es una aplicación que permite realizar una verificación de seguridad en una red mediante el análisis de los puertos abiertos en uno de los equipos o en toda la red. El proceso de análisis utiliza solicitudes que permiten determinar los servicios que se están ejecutando en un host remoto para identificar los riesgos de seguridad. En general, con este tipo de herramienta es posible efectuar un análisis en una serie o lista de direcciones IP a fin de realizar una verificación completa de una red.

El escáner de vulnerabilidades permite identificar los puertos que están abiertos en un sistema al enviar solicitudes sucesivas a diversos puertos, además de analizar las respuestas para determinar cuáles están activos. Mediante un análisis exhaustivo de la estructura de los paquetes TCP/IP recibidos, los escáneres de seguridad avanzados pueden identificar qué sistema operativo está utilizando el equipo remoto, así como las versiones de las aplicaciones asociadas con los puertos y recomendar actualizaciones.

Por lo general se usan dos métodos:

1. Adquisición activa de información: Consiste en enviar una gran cantidad de paquetes con encabezados característicos que normalmente no cumplen con las recomendaciones y analizar las respuestas para identificar la versión de la aplicación utilizada. Como todas ellas utilizan protocolos ligeramente diferentes, esto posibilita su diferenciación.
2. Adquisición pasiva de informaciones (también denominado *análisis pasivo* o *análisis no agresivo*): Es un método mucho menos invasivo que reduce la probabilidad de ser detectado por un sistema detector de intrusiones. Funciona de modo similar, efectuando un análisis de los campos de datagramas IP que circulan en una red utilizando un rastreador de puertos. Dada su naturaleza pasiva, la versión analiza los cambios en los valores de campo dividiéndolos en una serie de fragmentos, lo que requiere un tiempo de análisis mucho más prolongado. Por ello,

este tipo de análisis es muy difícil e incluso imposible de detectar en determinadas ocasiones.

Los escáneres de seguridad son herramientas sumamente útiles para los administradores de sistemas y redes, ya que les permite supervisar la seguridad de todos los equipos que están a su cargo, sin embargo, esta herramienta puede ser utilizada por los piratas informáticos para identificar las vulnerabilidades del sistema.

2.2.3.2.1 Nmap (mapeador de redes)

Es una herramienta de código abierto para exploración de red y auditoría de seguridad. Se diseñó para analizar rápidamente grandes redes, aunque funciona muy bien contra equipos individuales. Nmap utiliza paquetes IP en formas originales para determinar qué equipos se encuentran disponibles en una red, qué servicios (nombre y versión de la aplicación) ofrecen, qué sistemas operativos (y sus versiones) ejecutan, qué tipo de filtros de paquetes o cortafuegos se están utilizando así como otras características. Aunque generalmente se utiliza para auditorías de seguridad, muchos administradores de redes y sistemas lo encuentran útil para realizar tareas rutinarias, como puede ser el inventariado de la red, la planificación de actualización de servicios y la monitorización del tiempo que los equipos o servicios se mantiene activos.

La salida de Nmap es un listado de objetivos analizados con información, la cual, es primordial en la tabla de puertos interesantes. Dicha tabla lista el número de puerto y protocolo, el nombre más común del servicio, y su estado. Dichos estados son:

- Abierto (open): significa que la aplicación en la máquina destino se encuentra esperando conexiones o paquetes en ese puerto.
- Filtrado (filtered): indica que cortafuegos, filtro u otro obstáculo en la red están bloqueando el acceso a ese puerto, por lo que Nmap no puede saber si se encuentra abierto o cerrado.
- Cerrados (closed): no tienen ninguna aplicación escuchando en los mismos, aunque podrían abrirse en cualquier momento.
- No filtrados (unfiltered): son aquellos que responden a los sondeos de Nmap pero que no se pueden determinar si se encuentran abiertos o cerrados. Nmap

informa de las combinaciones de estado open|filtered yclosed|filtered cuando no puede determinar en cuál de los dos estados está un puerto.

La tabla de puertos también puede incluir detalles de la versión de la aplicación cuando se ha solicitado detección de versiones. Nmap ofrece información de los protocolos IP soportados en vez de puertos abiertos cuando se solicita un análisis de protocolo IP con la opción (-sO). Además de la tabla de puertos interesantes, Nmap puede dar información adicional sobre los objetivos, incluyendo el nombre de DNS según la resolución inversa de la IP, un listado de sistemas operativos posibles, los tipos de dispositivo y direcciones MAC.

2.2.4 Tipos de escaneo

Existen 13 tipos de escaneo, algunos de ellos son:

2.2.4.1 Escaneo TCP SYN

Es el predeterminado y probablemente el más utilizado. También es el más rápido y discreto, ya que no llega a completar una conexión TCP. Su funcionamiento consiste en enviar un paquete SYN y esperar una respuesta por parte del servidor, si se recibe un SYN/ACK el puerto está abierto (si lo recibe sin el flag ACK también lo considera así), si se recibe un RST (reset) está cerrado y si no se recibe respuesta tras varios intentos se considera filtrado.

2.2.4.2 Escaneo TCP connect

El predeterminado cuando no tenemos acceso root y por tanto no podemos enviar paquetes. Nmap solicita al sistema que establezca una conexión con la máquina objetivo a través del puerto elegido mediante una llamada de tipo connect. Se trata de una opción menos eficiente que TCP SYN, ya que requiere más tiempo y paquetes para obtener la misma información.

2.2.4.3 Escaneo UDP

Dado que el escaneo de puertos UDP es más lento y dificultoso que el de TCP, muchas veces se deja de lado su auditoría. Es un error, ya que tanto DNS (puerto 53), como SNMP (puertos 161/162) y DHCP (puertos 67/68) corren sobre éste. El escaneo UDP funciona mediante el envío de un paquete UDP a los puertos seleccionados, de tal forma

que si se devuelve un error “ICMP unreachable” el puerto se considera cerrado o filtrado (en función del código de error) mientras que si hay respuesta mediante un paquete UDP se considera abierto.

2.2.4.4 Escaneo SCTP INIT

Este tipo de escaneo se establece como alternativa a los TCP y UDP y sería el equivalente a un escaneo TCP SYN en el ámbito SCTP. Se trata de un escaneo rápido que distingue bien entre los estados abierto, cerrado y filtrado. Además, es muy poco intrusivo ya que no completa la asociación STCP sino que envía un paquete INIT como si se pretendiera abrir una conexión y espera la respuesta, si recibe un INIT-ACK, el puerto está abierto, mientras que si recibe un ABORT el puerto está cerrado. En caso de no recibir respuesta tras varios intentos, el puerto se marca como filtrado.

2.2.4.5 Escaneo TCP personalizado

Esta modalidad tiene como finalidad permitir que el usuario defina su análisis a la carta, especificando las TCP flags a utilizar (URG, ACK, PSH, RST, SYN, FIN) y el tipo de escaneo TCP (-sF, -sA).

Teniendo en cuenta que ningún sistema operativo es vulnerable a cualquier ataque, investigadores de seguridad han descubierto hasta 18 vulnerabilidades en las diferentes versiones del sistema operativo Android. De éstas, la mayoría eran de poca gravedad y sólo permitían a un atacante tomar el control de un solo proceso (por ejemplo, el proceso del navegador Web), pero no permitían tomar el control a nivel de administrador del equipo. Las vulnerabilidades restantes son más peligrosas y cuando se explotan, pueden permitir que el atacante tome el control del dispositivo a nivel de súper usuario, permitiendo el acceso a todos los datos en el dispositivo.

Hasta la fecha, todas, salvo cuatro de estas vulnerabilidades han sido parcheadas por Google. De las cuatro vulnerabilidades sin resolver, una es más grave. Esta vulnerabilidad ha sido solucionada en la versión 2.3 de Android, pero no ha sido resuelta para las versiones anteriores del sistema operativo. Dado que la mayoría de las compañías no han actualizado los teléfonos de sus clientes de Android 2.2 a 2.3, esto significa que prácticamente todos los actuales teléfonos Android (en el momento de escribir este documento) están abiertos a posibles ataques de esta vulnerabilidad. Esta

vulnerabilidad puede ser explotada por cualquier aplicación de terceros y no requiere que el atacante tenga acceso físico al dispositivo. A modo de ejemplo, el reciente malware Android.Rootcager y Android.Bgserv aprovechan esta vulnerabilidad para obtener el nivel de administrador en los dispositivos [4].

En la web muestran posibles soluciones a las vulnerabilidades que los dispositivos móviles están mostrando a la hora de su creación.

2.2.5 Antivirus

Ya existe una serie de escáneres antivirus de primera generación para la plataforma Android. Estos escáneres son efectivos en la detección de amenazas conocidas pero ofrecen poca protección contra amenazas desconocidas. Cuatro de los antivirus más reconocidos para dispositivos móviles Android son los mencionados a continuación.

2.2.5.1 AVG Antivirus Free

Este es uno de los antivirus más comunes que permite mantener su dispositivo protegido con una avanzada solución de seguridad que además incluye: análisis de aplicaciones, configuraciones, archivos y multimedia en tiempo real, además, permite encontrar su teléfono extraviado por medio de Google Maps, realizar copias de seguridad y restaurar todas las aplicaciones y datos más valiosos, bloquear y limpiar el dispositivo para proteger su privacidad, detener tareas o procesos que ralentizan el funcionamiento del teléfono [9].

2.2.5.2 Lookout Mobile Security

Lookout Mobile Security incluye antivirus, copias de seguridad y localización de su smartphone. Bloquea los virus, malware, spyware, descargar de aplicaciones de manera segura, analiza cada aplicación que se descarga y se ejecuta en segundo plano. Copia de seguridad y recuperación de Datos, copia de contactos y fotos, permite acceder a sus datos a través de: http://myLookout.com, restaurar los datos a un teléfono nuevo o existente. Permite detectar cuando el teléfono se encuentra en el mapa, activa una alarma fuerte para encontrar el teléfono, borrando de forma remota los datos para siempre [9].

2.2.5.3 Súper Security Standard

Proporciona protección en la nube basado en un motor antivirus para ayudar a deshacerse de malware. Ofrece caja fuerte para ocultar datos confidenciales. Sus principales características son: búsquedas de nuevos dispositivos, task Manager, anti malware, software Manager, caja fuerte, comunitarios de motor antivirus, live update, tiempo real monitor, soporte de copia de seguridad de sms, la alarma por la falta de dispositivos [9].

2.3 ANTECEDENTES

En Cartagena de indias se realizó una investigación sobre un **Herramienta para la Detección de Vulnerabilidades basada en la Identificación de Servicios** [16], justificada por la necesidad de diseñar un nuevo enfoque para la detección y evaluación de vulnerabilidades en equipos de red mediante la técnica de identificación de servicios. Una de las conclusiones del trabajo fue que la implementación del enfoque propuesto ayuda a los administradores de las tecnologías de la información y la comunicación en la mejor comprensión de los riesgos reales a los que están expuestos, facilitando la mitigación de vulnerabilidades de seguridad.

3. OBJETIVOS

3.1 Objetivo general:

Desarrollar un aplicativo cliente-servidor para la detección de vulnerabilidades de red en smartphones Android utilizando una herramienta de escaneo.

3.2 Objetivos específicos:

- Realizar una revisión del estado del arte sobre los problemas de seguridad de los smartphones Android y de las herramientas de escaneo de vulnerabilidades.

- Identificar las especificaciones y requerimientos de un componente arquitectónico software que permita el desarrollo de un aplicativo cliente servidor para la detección de vulnerabilidades en smartphones Android utilizando una herramienta de escaneo.

- Diseñar un componente arquitectónico software que permita el desarrollo de un aplicativo cliente servidor para la detección de vulnerabilidades de red en smartphones Android utilizando una herramienta de escaneo.

- Implementar el software en base al componente arquitectónico software propuesto que responda a las necesidades de desarrollar un aplicativo cliente servidor para la detección de vulnerabilidades en smartphones Android utilizando una herramienta de escaneo.

- Evaluar la efectividad del software implementado.

- Documentar la investigación en las diferentes fases del proyecto, teniendo en cuenta lo descrito en el cronograma.

3.3 ALCANCE

La idea principal del proyecto es desarrollar un aplicativo cliente-servidor para la detección de vulnerabilidades de red en smartphones Android, dicha detección se realizará mediante la conexión existente entre una aplicación y un servidor que contendrá la herramienta de escaneo que se encargará de verificar las vulnerabilidades que existan en la red a la que se encuentre conectado el Smartphone. Los resultados devueltos por la herramienta se basarán en los registros de la National Vulnerability Database, la cual es una fuente oficial de información sobre los fallos de seguridad que se detectan en diferentes sistemas operativos y software. En primera instancia, este proyecto se enfocará únicamente en la detección de vulnerabilidades, pero se dejará la puerta abierta a que en futuras investigaciones se pueda ofrecer una solución a los fallos que se encuentren durante el escaneo de una red. Para iniciar este proyecto, se realizó la revisión del estado del arte sobre los problemas de seguridad en los smartphones y se investigaron las herramientas de escaneo de vulnerabilidades existentes en la actualidad, lo anterior, conllevó a identificar las especificaciones y requerimientos necesarios para el desarrollo de un aplicativo que fuese capaz de responder a la necesidad planteada. La efectividad del software que se diseñó e implementó fue evaluada y su constancia queda documentada en las diferentes fases del proyecto de investigación.

4. METODOLOGÍA

La presente investigación fue de tipo aplicada ya que se desarrolló un aplicativo cliente-servidor para la detección de vulnerabilidades de red en smartphones Android utilizando una herramienta de escaneo.

El diseño utilizado en la presente investigación fue de carácter exploratorio y experimental, pues se realizó una revisión de literaturas sobre la temática específica que resaltaron aspectos significativos para la investigación. Además, se tuvo un control y manipulación sobre las variables implicadas, siendo transcendental para evaluaciones técnicas y captura de datos importantes en la investigación.

Cabe resaltar que el proyecto ha sido desarrollado en la ciudad de Cartagena de Indias (Bolívar) durante los años 2014 y 2015.

La metodología utilizada en este trabajo de grado abarcó actividades de investigación, diseño, implementación, configuración, evaluación y análisis, para contribuir a la implementación de un software en un contexto específico. Para lograr lo anterior, el trabajo se dividió en actividades particulares que corresponden a la realización de los objetivos específicos del proyecto, plasmados en la sección de objetivos. Primeramente se realizó una revisión del estado del arte sobre los problemas de seguridad de los smartphones Android y de las herramientas de escaneo de vulnerabilidades. Esta investigación se dejó plasmada dentro del estado del arte, **esto corresponde al primer objetivo específico**.

Posteriormente, se identificaron las especificaciones y requerimientos de un componente software que permitiera el desarrollo de un aplicativo cliente-servidor para la detección de vulnerabilidades en smartphones Android utilizando una herramienta de escaneo, teniendo en cuenta estudios previos que garantizaran el desarrollo del aplicativo para atender los requerimientos identificados, **esto corresponde al segundo objetivo específico.**

Con los resultados de la etapa anterior finalizados, la siguiente fase se encamina hacia el desarrollo técnico del proyecto, por lo tanto, se realizó el modelo de diseño a partir de la elaboración de los diferentes diagramas (diagramas estructurales, de comportamiento, de interacción) propuestos por el lenguaje de modelado unificado UML[1], los cuales hacen una descripción detallada del aplicativo cliente-servidor para escaneo de vulnerabilidades en sistemas Android. Como resultado de la elaboración de estos diagramas se obtuvo el diseño de la arquitectura del sistema, **esto corresponde al tercer objetivo específico.**

Con las especificaciones, requerimientos y componentes arquitectónicos ya definidos, se pasó a la construcción del aplicativo cliente-servidor, que tiene como función detectar vulnerabilidades. Con el propósito de satisfacer las necesidades de un cliente, se empleará un equipo o servidor de prueba para la puesta en marcha del prototipo software propuesto. Para su distribución, el aplicativo será ofrecido por medio de un servidor para cualquier usuario que posea un Smartphone Android, **esto corresponde al cuarto objetivo específico**.

El siguiente paso para la etapa final del proyecto, es evaluar la efectividad del software a través de pruebas periódicas que muestren resultados puntuales que determinen si hay una vulnerabilidad presente en el dispositivo. Se analizó el sistema en un ambiente ideal con el fin de garantizar que cumple con el propósito esperado, **esto corresponde al quinto objetivo específico**.

Finalmente, se documenta la investigación en las diferentes fases del proyecto ejecutadas para tener una organización de la información y que sea útil para futuras generaciones que se interesen en investigar y desarrollar sobre esta temática, además, se redactó el informe final donde se mostraron los resultados obtenidos durante todo el trabajo de investigación y desarrollo, se resaltaron las conclusiones acerca de las contribuciones alcanzadas en los diferentes procesos y los posibles escenarios factibles para su continuación, **esto corresponde al sexto objetivo específico**.

UML, por sus siglas en inglés, *Unified Modeling Language.* Para mayor información visite la página oficial **http://www.uml.org/**

Teniendo en cuenta cada una de las etapas anteriores, se llega al final de la metodología utilizada en el proceso de esta investigación. Este último paso es de suma importancia para el gremio de consumidores de teléfonos Android ya que gracias a los resultados obtenidos se puede hablar de un análisis económico y eficiente a la hora de detectar vulnerabilidades. Los valores que se reflejan y que se mostrarán más adelante hacen referencia a cada una de las necesidades identificadas y que son necesarias para lograr un óptimo rendimiento del software. Además, se logró cumplir con los diferentes atributos de calidad entregándole al usuario final un aplicativo con características de facilidad de uso y que a su vez cumple con los requerimientos solicitados.

5. RESULTADOS Y DISCUSIÓN

La especificación de requerimientos se realiza para cumplir con el objetivo específico 2, el cual abarca identificar las especificaciones y requerimientos de un componente arquitectónico software que permita el desarrollo de un aplicativo cliente servidor para la detección de vulnerabilidades de red en smartphones Android utilizando una herramienta de escaneo.

5.1 Especificación de requerimientos

La especificación de requerimientos son los que describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento, los requerimientos son aquellas propiedades o restricciones determinadas de forma precisa que deben satisfacerse.

Es importante resaltar que este proceso se hizo con base en la información obtenida de fuentes primarias y el estado de arte.

5.1.1 Requerimientos funcionales

Expresa la naturaleza del funcionamiento del sistema, como interactúa el sistema con su entorno y cuáles van ser su estado y funcionamiento. Este capítulo contiene los requerimientos que representa el comportamiento funcional y característico que el sistema debe soportar. En la Tabla se presenta los requisitos funcionales que luego se describen en forma detallada teniendo en cuenta que su cumplimiento es verificado en el subcapítulo de pruebas.

Código	Requerimientos Funcionales	Prioridad
Rq1	La aplicación escanea vulnerabilidades de red en dispositivos móviles Android en su totalidad	**Alta**
Rq2	La aplicación muestra cada una de las vulnerabilidades.	**Alta**
Rq3	La aplicación asocia las vulnerabilidades a un nombre extrayéndolo de una base de datos	**Alta**
Rq4	El aplicativo ofrece informes detallados de cada una de los puertos y vulnerabilidades de un dispositivo Android.	**Alta**

Tabla 2. Requerimientos Funcionales.

5.1.1.1 La aplicación Escanea vulnerabilidades de red en dispositivos móviles Android.

El aplicativo de detención de vulnerabilidades tiene como función principal escanear la red en su totalidad a través de un dispositivo móvil, arrojando como resultado cada una de los puertos abiertos o sus vulnerabilidades, haciendo referencia a toda la información necesaria que permita corregir las falencias.

5.1.1.2 La aplicación muestra cada una de las vulnerabilidades.

El sistema tiene como característica o función, mostrar datos específicos sobre las vulnerabilidades encontradas después de un escaneo al dispositivo móvil. La aplicación mostrara una ventana adicional indicando vulnerabilidades y en qué sector se están dando.

5.1.1.3 La aplicación asocia las vulnerabilidades a un nombre extrayéndolo de una base de datos

La aplicación se conectara a una base datos hecha en MySQL donde se exportará el nombre de cada una de las vulnerabilidades. La base de datos se alimentará del estado de arte, donde se especifican y se detallan las vulnerabilidades con el nombre que se le ha dado para identificarlo, y así sea más fácil buscar posibles soluciones.

5.1.1.4 El aplicativo ofrece informes detallados de cada una de los puertos y vulnerabilidades de un dispositivo Android.

El aplicativo después de haber realizado el escaneo y haber asociado las vulnerabilidades a los nombres dados en la base de datos, posee la capacidad de realizar un informe dónde establece y muestra cada uno de las falencias o puertos abiertos del dispositivo móvil Android.

5.1.2 Requerimientos no funcionales

Contiene todas aquellas restricciones y niveles de desempeño que el sistema debe cumplir. Estos requisitos se presentan como reglas explícitas que se implementaron en el proyecto, que se ejecutan durante la puesta en marcha del programa y controlan el procesamiento de la información y transacciones. En la Tabla 3 se presentan los requisitos no funcionales seguidos de una descripción detallada de cada uno de ellos.

Código	Requerimientos No Funcionales	Prioridad
Rq1	Facilidad de uso	**Alta**
Rq2	Interfaz gráfica	**Alta**
Rq3	Seguridad	**Alta**
Rq4	Facilidad de mantenimiento	**Alta**

Tabla 3. Requerimientos No Funcionales.

5.1.2.1 Facilidad de Uso

El sistema debe permitir ser usado con eficacia y eficiencia por personas con conocimientos básicos de informática e internet.

5.1.2.2 Interfaz Gráfica

El sistema debe ser agradable a la vista para los usuarios, debe contar con una interfaz atractiva que maneje los colores institucionales

5.1.2.3 Seguridad

El aplicativo se instalará en cada dispositivo y para realizar el escaneo se conectará a una base de datos que estará instalada en un servidor que arrojará la información.

5.1.2.4 Facilidad de Mantenimiento

La aplicación está construida sobre la base de un desarrollo evolutivo e incremental. De manera que nuevas funcionalidades y requerimientos eventuales relacionados puedan ser incorporados afectando en lo más mínimo el código existente.

5.2 Casos de uso del mundo real

Se procedió a la elaboración de la vista de casos de uso del mundo real a través de la cual se realizó una definición del alcance del software en cada uno de los subsistemas que lo constituyen. Teniendo en cuenta que la metodología utilizada se enfoca en los casos de uso, es de vital importancia identificarlos de forma adecuada y precisa. Al momento de hacer la recolección de información se detectaron los anteriores casos de uso del mundo real, permitiendo entender de manera global el comportamiento de cada una de las partes.

5.3 Diseño del sistema

El diseño del sistema se realiza para cumplir con el objetivo específico 3, el cual abarca el diseño un componente software que permita el desarrollo de un aplicativo cliente servidor para la detección de vulnerabilidades en smartphones Android utilizando una herramienta de escaneo.

En este punto se especifica y se describe la manera cómo se elaboró el sistema, es decir, lo que se definió como la mejor solución informática a partir de los requerimientos previamente establecidos, obteniendo una arquitectura de software adecuada, previa a las actividades de implementación.

5.3.1 Modelo de dominio

Un modelo de dominio es un artefacto de la disciplina de análisis, construido durante la fase de concepción y que no contiene conceptos propios de un sistema de software sino de la realidad física.

Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento en una área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo, similares a los mapas mentales utilizados en el aprendizaje, además, es utilizado por el analista como un medio para comprender el sector al cual el sistema va a servir.

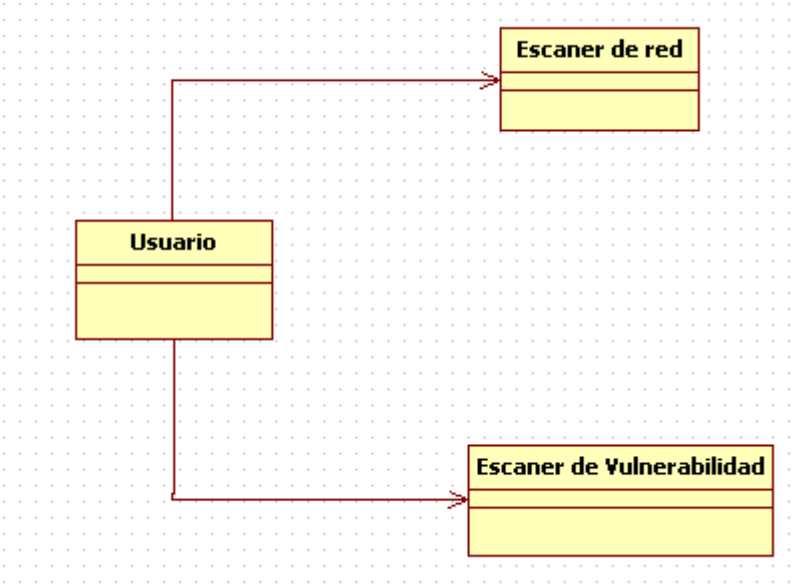


Ilustración 2. Modelo de dominio del proyecto.

5.3.2 Vista de casos de uso

Se procedió a la elaboración de la vista de casos de uso a través de la cual se realizó una definición del alcance funcional del software en cada uno de los subsistemas que lo componen. De acuerdo a lo mostrado anteriormente, este sistema se encuentra organizado al más alto nivel en tres subsistemas funcionales. Teniendo en cuenta que la metodología utilizada se guía en los casos de uso, es de vital importancia identificarlos de forma adecuada y precisa. A continuación se muestra el caso de uso general y algunos de los casos de uso más importantes utilizados en el desarrollo del sistema.

5.3.2.1 Caso de uso general

Caso de uso que describe las funcionalidades de la aplicación.

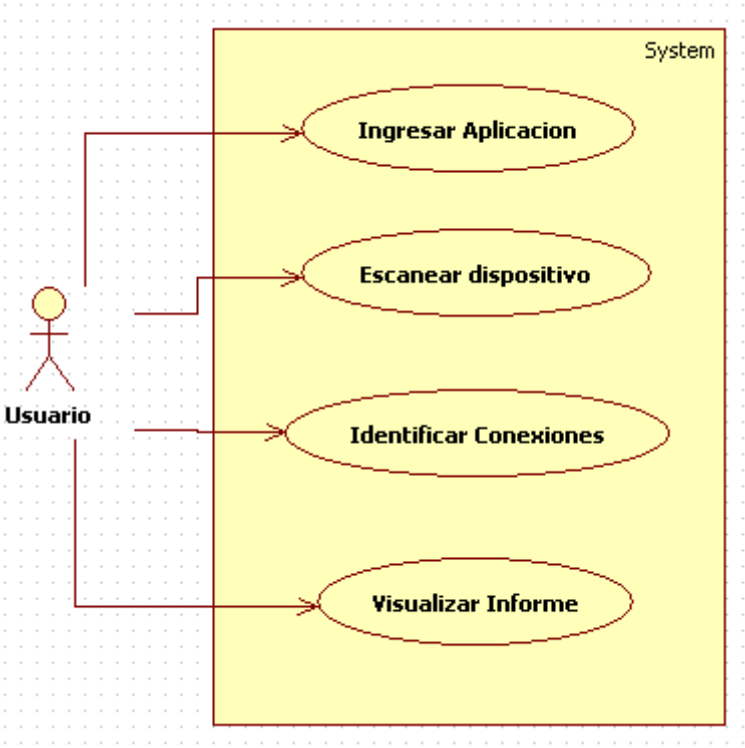


Ilustración 3. Caso de uso general.

5.3.2.2 Caso de uso: Escanear dispositivo

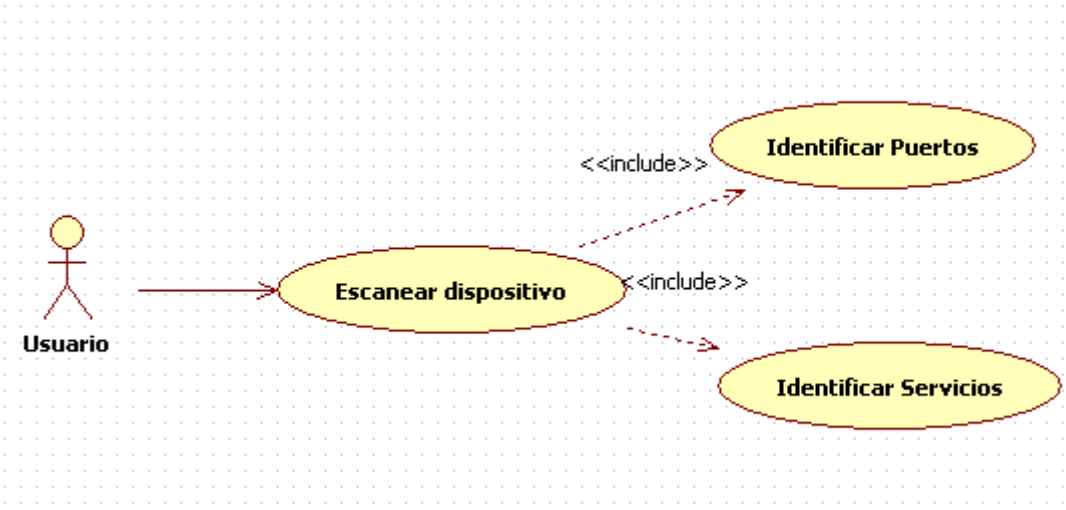


Ilustración 4. Caso de uso: Escanear dispositivo.

Actor Principal: El Usuario
Personal Involucrado e Intereses:
Usuario: El usuario requiere Identificar puertos abiertos
Precondiciones: El usuario debe tener un dispositivo Android
Garantías de éxito (Postcondiciones): El usuario identifica los puertos abiertos de su dispositivo Android.
Escenario principal de éxito (o flujo básico):
1 .El usuario abre la aplicación.
2 .El Sistema inicia el proceso de escaneo de los puertos.
3. El sistema muestra el estado de los puertos del dispositivo.
4. El usuario reconoce los puertos de su dispositivo.
Extensiones (o Flujos alternativos):
a. En cualquier momento el sistema falla.
Requisitos especiales:
Tiempo de ejecución de la acción debe ser menor de 15 segundos.
Interfaz de usuario con buen diseño gráfico.
Lista de Tecnologías y variaciones de datos:
1. El sistema operativo del dispositivo debe ser Android.
Frecuencia:
Cada vez que el usuario requiera escanear los puertos del dispositivo.

5.3.2.3 Caso de uso: Identificar conexiones

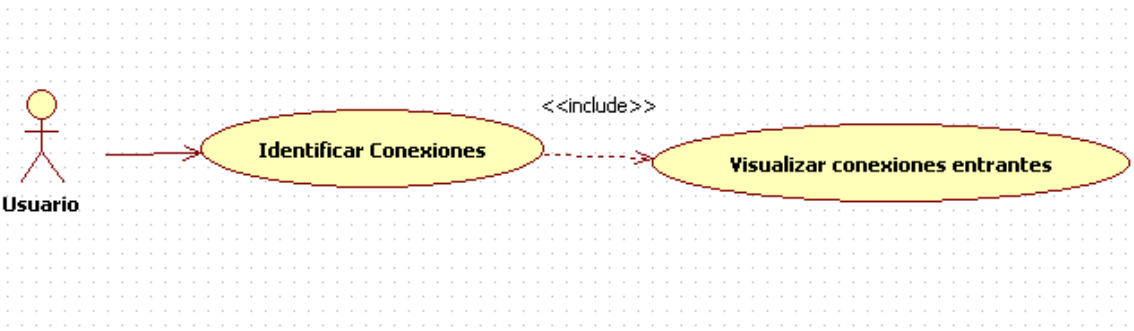


Ilustración 5. Caso de uso: Identificar conexiones.

Actor Principal: El Usuario
Personal Involucrado e Intereses:
Usuario: Desea identificar las direcciones IP conectadas al dispositivo.

Precondiciones: El usuario debe tener un dispositivo móvil Android con acceso a internet.
Garantías de éxito (Postcondiciones): El usuario escanea su dispositivo e identifica las direcciones IP conectadas a él.
Escenario principal de éxito (o flujo básico):

1. El usuario abre la aplicación.
2. El sistema escanea los puertos del dispositivo.
3. El sistema retorna la lista de los puertos con conexiones entrantes.
4. El usuario selecciona cualquiera de los puertos con conexiones entrantes.
5. El sistema retorna la dirección IP que tiene conexión hacia el puerto seleccionado.
6. El usuario identifica la dirección IP.

Extensiones (o Flujos alternativos):
a. El sistema no retorna el listado de puertos con conexiones entrantes.
2. El usuario inicia nuevamente el escaneo de los puertos del dispositivo.
b. El sistema no retorna la dirección IP que tiene conexión hacia el puerto seleccionado.
4. El usuario nuevamente selecciona uno de los puertos con conexiones entrantes.
Requisitos especiales:
Tiempo de ejecución de la acción debe ser menor de 15 seg.
Interfaz de usuario con buen diseño gráfico.
Lista de Tecnologías y variaciones de datos:
1. El sistema debe tener conexión a Internet.
Frecuencia:
Cada vez que el usuario desee identificar las direcciones IP conectadas a su dispositivo

5.3.2.4 Caso de uso: Escanear vulnerabilidades

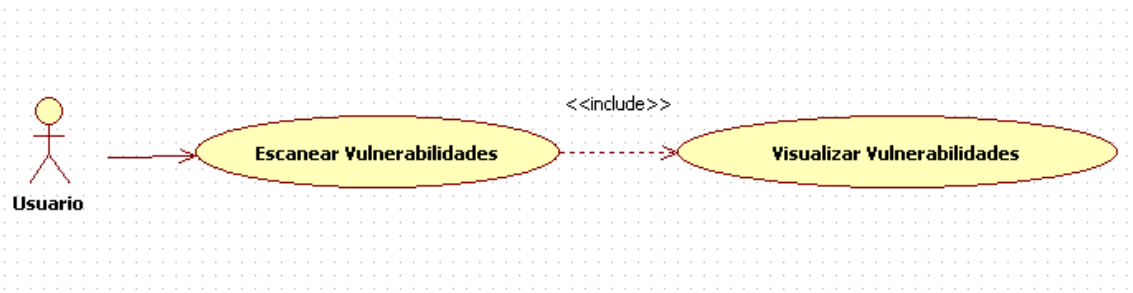


Ilustración 6. Caso de uso: Escanear vulnerabilidades.

Actor Principal: El Usuario
Personal Involucrado e Intereses:
Usuario: Requiere identificar las vulnerabilidades de la red a la que pertenece su dispositivo.
Precondiciones: El usuario debe tener un móvil Android, con acceso a internet.

Garantías de éxito (Postcondiciones): el usuario detecta las vulnerabilidades de la red a la que pertenece su dispositivo.
Escenario principal de éxito (o flujo básico):
1 .El usuario accede a la aplicación.
2. El usuario solicita el escaneo de vulnerabilidades de red.
3. El sistema se conecta al servidor.
4. El servidor inicia el escaneo de las vulnerabilidades.
5. El servidor retorna los resultados de la operación.
6. El sistema muestra las vulnerabilidades detectadas.
Extensiones (o Flujos alternativos):
a. No es posible conectarse al servidor.
1. el usuario nuevamente inicia el proceso de escaneo de vulnerabilidades.
b. El servidor no retorna los resultados de la operación.
1. el usuario nuevamente inicia el proceso de escaneo de vulnerabilidades.
Requisitos especiales:
Tiempo de ejecución de la acción debe ser menor de 60 segundos.
Interfaz de usuario con buen diseño gráfico.
Lista de Tecnología y variaciones de datos:
1. El sistema debe tener conexión internet.
Frecuencia:
Cada vez que el usuario quiera detectar las vulnerabilidades de red

5.4 Vista de diseño

La arquitectura hardware y software se definieron en esta fase, describiendo los subsistemas y las interrelaciones entre cada uno de ellos. Por medio de esta etapa se pudieron identificar los elementos más importantes del sistema así como sus relaciones, obteniendo una visión macro del mismo, la cual se representa por medio de las vistas contempladas en la metodología RUP.

A continuación se muestra la funcionalidad del diseño dentro del sistema en términos de la estructura estática y el comportamiento dinámico del sistema. En ésta fase fueron construidos los diagramas de clases y entidad relación que se muestran en las ilustraciones a continuación.

5.4.1 Diagrama de clases

Es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y relaciones entre ellos. Son utilizados durante el proceso de análisis y

diseño de sistemas, donde se crea el diseño conceptual de la información que se manejará además de los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

Los diagramas de clase son importantes no sólo para la visualización, especificación y documentación del modelo estructural, sino también para la construcción de sistemas ejecutables. Las siguientes ilustraciones muestran los diagramas de clases correspondientes al sistema adoptado, los cuales presentan las diferentes interacciones entre las entidades para así conocer su comportamiento.

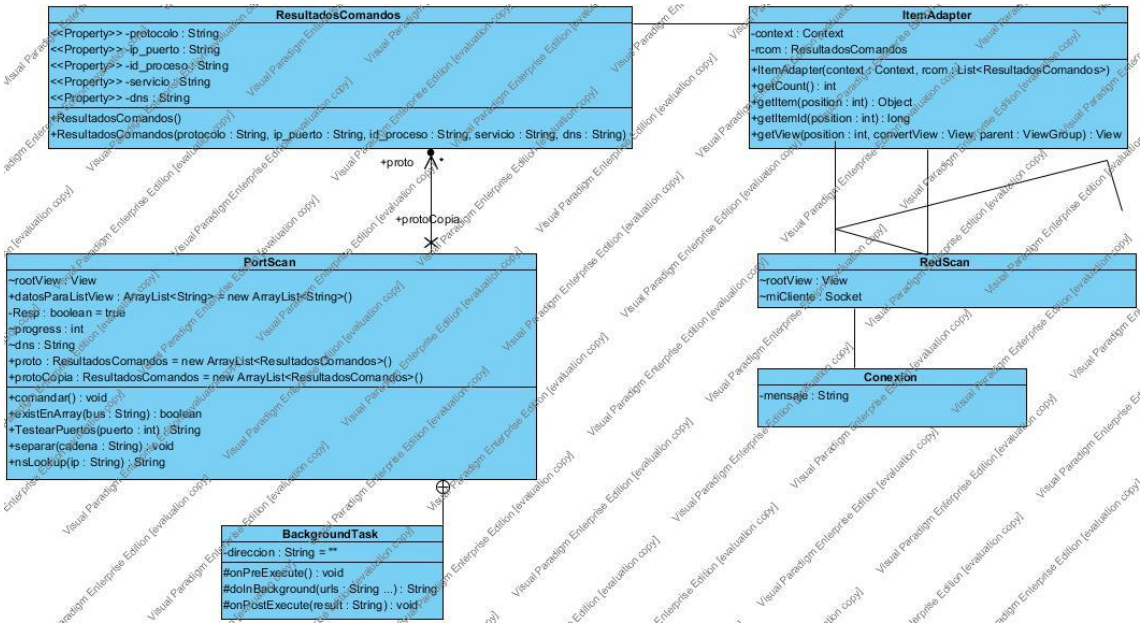


Ilustración 7. Diagrama de clases general.

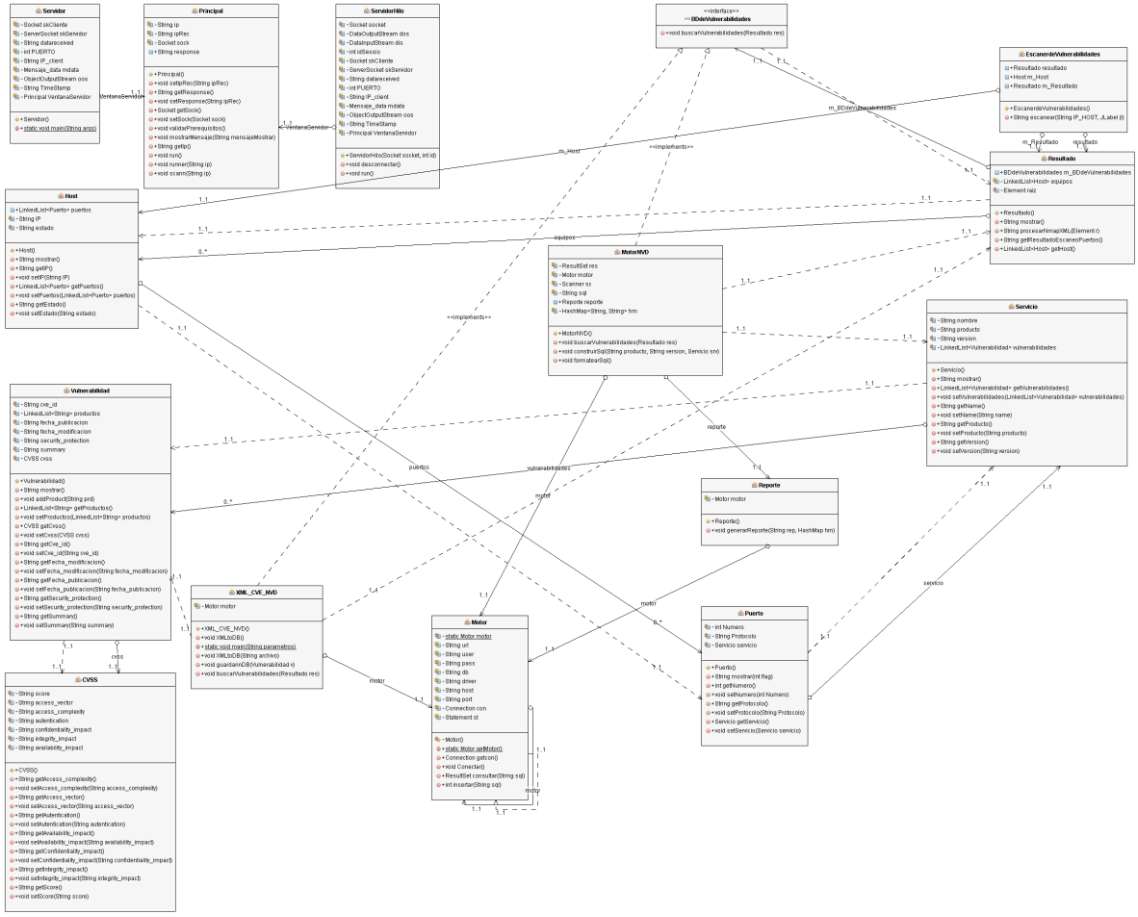


Ilustración 8. Diagrama de clases servidor.

5.5 Implementación y desarrollo del sistema

La implementación y desarrollo del sistema se realiza para cumplir con el objetivo específico 4, que se refiere a la implementación del software en base al componente arquitectónico software propuesto que responda a las necesidades de desarrollar un aplicativo cliente servidor para la detección de vulnerabilidades en smartphones Android utilizando una herramienta de escaneo.

En este apartado se presentan los programas y lenguajes utilizados para el desarrollo del software, los cuales fueron elegidos en su momento por su facilidad de uso, estabilidad, seguridad y mejor rendimiento, lo que permitió un buen desarrollo del sistema. Entre los cuales se puede apreciar:

- Udcescaner.
- Nmap.

- Lenguaje de Programación: JAVA.

5.5.1 Diagrama de despliegue

Básicamente los diagramas de despliegue muestran el hardware del sistema, el software instalado en él y el middleware utilizado para conectar unos nodos con otros. Generalmente el diagrama de despliegue es utilizado para aplicaciones que se tendrán en diferentes máquinas, aunque también se crean para tener claro el funcionamiento de sistema donde se quiera apreciar la forma en que el software y el hardware trabajan juntos. Lo anterior se puede apreciar en la siguiente ilustración.

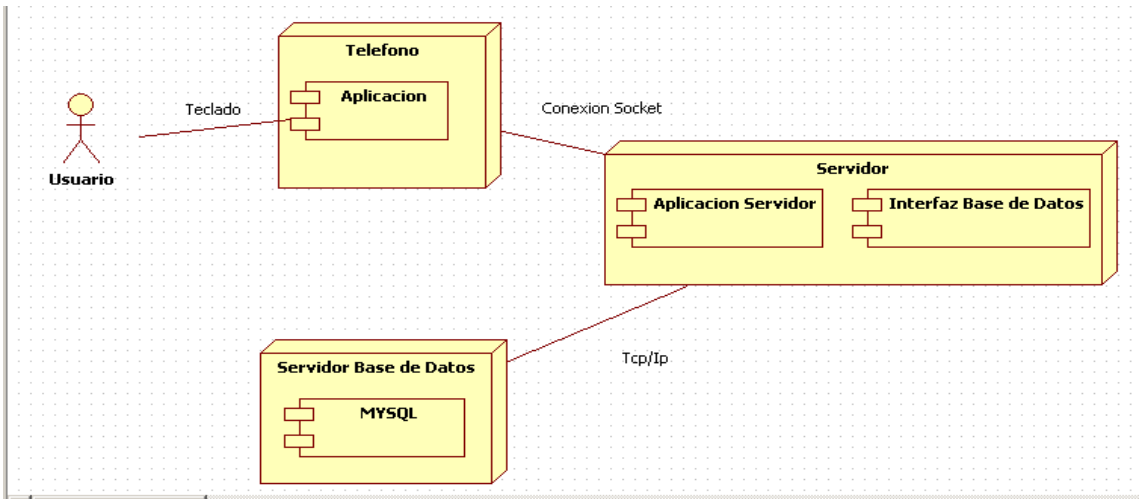


Ilustración 9. Diagrama de despliegue.

5.5.2 Diagrama de componentes

Debido a la utilización del patrón arquitectónico Modelo Vista Controlador se realiza la separación del sistema en tres capas principales, con lo cual se separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue mayor seguridad y un mantenimiento más sencillo de la aplicación, como se puede apreciar en la siguiente ilustración.

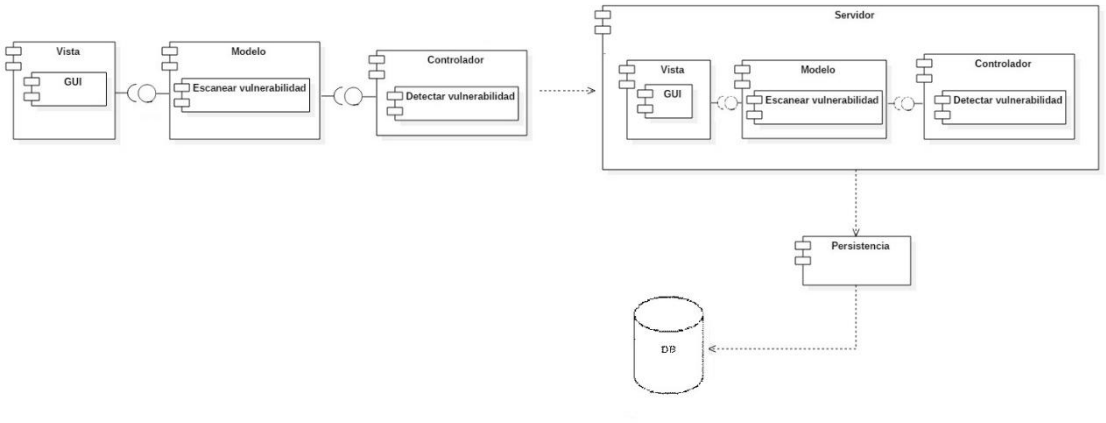


Ilustración 10. Diagrama de componentes del sistema.

5.6 Pruebas realizadas

La fase de pruebas se realizó para cumplir con el objetivo específico 5 que corresponde a la evaluación de la efectividad del software implementado.

Las pruebas del sistema se realizaron con el objetivo de buscar incompatibilidades entre el programa y sus objetivos, enfocándose en los errores hechos durante la transición del proceso al diseñar la especificación funcional. Esto hace a las pruebas del sistema un proceso vital, ya que en términos del producto, número de errores hechos, y severidad de estos mismos, es un paso en el ciclo de desarrollo generalmente propenso a la mayoría de las fallas. Las pruebas se basaron en dos puntos fundamentales, las pruebas de sistema y las pruebas de funcionalidad.

5.6.1 Pruebas del sistema

En este tipo de pruebas no se enfocó la atención a cómo se generaron las respuestas del sistema sino fundamentalmente a la perspectiva del análisis de los datos de entrada y los de salida.

5.6.1.1 Instalación

El aplicativo no requiere demasiados permisos para poder ser instalado. A diferencias de otras aplicaciones, los permisos son numerosos e insistentes que provoca en el usuario final duda o inseguridad que contribuye a la no instalación del aplicativo.

La aplicación es instalada de forma externa con un .apk activando la opción de aceptar la instalación de orígenes desconocidos.

5.6.1.2 Almacenamiento

La aplicación requiere aproximadamente 10 MB de espacio de almacenamiento, lo que es un tamaño considerable para ser una aplicación instalada en un Smartphone Android.

5.6.1.3 Memoria RAM

El consumo de memoria RAM generado por la aplicación es bajo es aproximadamente de 5MB en segundo plano y de 7MB en ejecución. Es un consumo relativamente considerable a diferencia de otros consumos efectuado por otras aplicaciones.

Es importante resaltar que los ítems mencionados anteriormente pueden variar según la gama del Smartphone Android, por lo tanto se tomó una muestra de 6 dispositivos de varias marcas para verificar el funcionamiento del aplicativo. La siguiente tabla muestra el resultado obtenido durante la instalación, espacio de almacenamiento y consumo de RAM.

Referencia	Instalación	RAM	Almacenamiento
Huawei G510	√	X	√
Motorola moto G	√	√	√
Samsung Galaxy S6	√	√	√
LG G3	√	√	√
Alcatel One Touch Idol 3	√	X	√

Tabla 4. Resultado de pruebas del sistema.

Como se puede observar en la tabla anterior, el comportamiento de la aplicación varía según las características del smartphone que se utiliza. Por ejemplo, el consumo de RAM en un Huawei G510 es alto a diferencia de un Samsung Galaxy 6.

5.6.2 Pruebas de funcionamiento

Estas pruebas fueron realizadas para encontrar incompatibilidades entre la aplicación y los requerimientos funcionales del sistema.

Esta prueba fue aplicada para determinar que el sistema funciona correctamente y que puede ofrecer un rendimiento óptimo aún en situaciones de constantes peticiones. En

otras palabras, el sistema debe ser capaz de responder a varios usuarios que soliciten los mismos recursos durante el mismo período de tiempo. Esto se ve evidenciado, cuando el sistema se comporta correctamente.

Para realizar esta prueba, se tomó una muestra de 10 dispositivos, se instaló la aplicación y se ejecutó para determinar los puertos en escucha reconocidos por la aplicación y así detectar cuales eran vulnerables. Se tomó al azar un puerto de cada dispositivo para visualizar su estado y bajo que protocolo estaba trabajando. La siguiente tabla muestra el resultado de la prueba para los 10 dispositivos:

Dispositivo	Protocolo	Puerto	Abierto
1	UDP	43635	Escuchando
2	TCP	50896	Escuchando
3	TCP	39253	Escuchando
4	TCP	50896	Escuchando
5	TCP	58944	Escuchando
6	TCP	39885	Escuchando
7	TCP	46358	Escuchando
8	TCP	39297	Escuchando
9	TCP	44683	Escuchando
10	TCP	53212	Escuchando

Tabla 5. Resultado de escáner de puertos.

Como se puede observar en la tabla anterior, los puertos en escucha varían según la situación pero en determinadas ocasiones los puertos en escucha son iguales sin importar el dispositivo, como es el caso de los dispositivos 2 y 4, ambos tienen en escucha el puerto 5896.

Después de finalizar el escaneo de puertos, la aplicación escanea las vulnerabilidades y genera un reporte en PDF con la información detallada. La siguiente ilustración muestra el reporte generado por la aplicación en caso de encontrar una o varias vulnerabilidades.

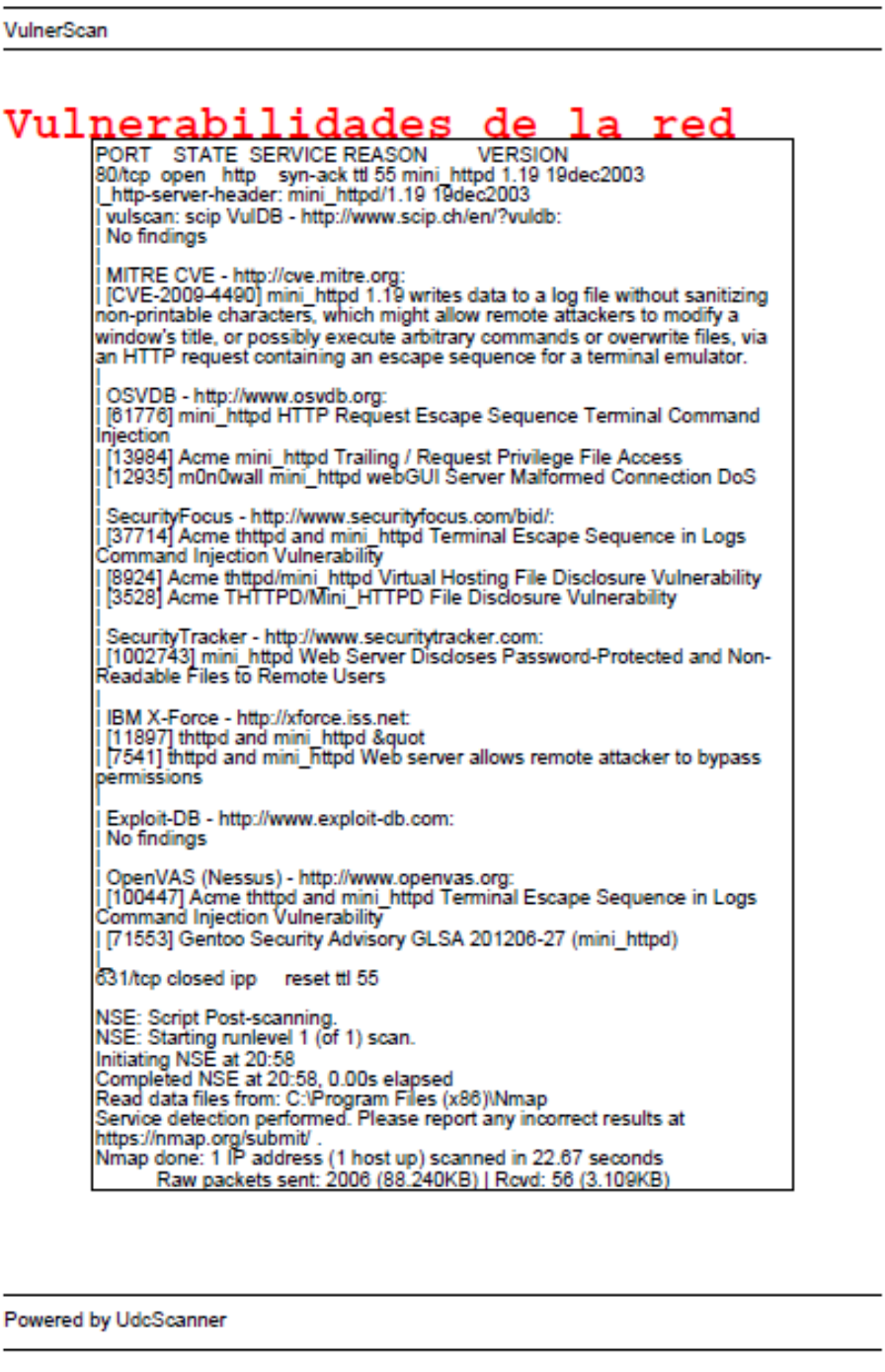


Ilustración 11. Reporte de vulnerabilidades encontradas en un smartphone.

De acuerdo a la ilustración anterior, se puede observar que el puerto tcp 80 está abierto para este dispositivo, lo que genera una vulnerabilidad grave porque las sesiones que se inicien pueden ser interceptadas por terceras personas. Cabe destacar que el idioma del reporte es en ingles técnico, fácil de interpretar para la resolución de problemas.

Con este último procedimiento se finaliza la parte de pruebas del sistema y de funcionamiento. Cada dispositivo utilizado en este apartado de pruebas posee un fabricante distinto, sin embargo, el dispositivo no presentó inconvenientes a la hora de ser instalado y ejecutado gracias a que el sistema operativo ejecutado es Android sin importar la interfaz implementada por el fabricante.

Es importante resaltar que hoy día, la probabilidad de encontrar un sistema vulnerable tiende a ser baja pero no es nula, por lo tanto, una herramienta útil y eficiente es indispensable para detectar qué vulnerabilidad tiene un sistema. Posiblemente en dispositivos móviles las vulnerabilidades son detectadas y eliminadas con cada actualización de la versión de Android por parte del fabricante, pero no está de más contar con una herramienta que nos indique que tan vulnerables somos y nos genere un reporte con dicha información.

6. CONCLUSIONES

Se desarrolló como objetivo principal de esta investigación un aplicativo cliente-servidor para la detección de vulnerabilidades en smartphones Android utilizando una herramienta de escaneo.

Cada objetivo creaba una incógnita que debía ser solucionada, por lo tanto, cada inconveniente que surgía durante el desarrollo del proyecto se estudiaba, se analizaba y se buscaba una propuesta que solucionara el inconveniente generado. En la búsqueda de resultados se dieron altibajos que no favorecían el cumplimiento del objetivo principal del proyecto, sin embargo, se obtenían otros que se mostraban positivos para la posible implementación del aplicativo. Se encontraron dificultades para conectar el aplicativo basado en Android con el aplicativo de detección de vulnerabilidades basado en Java y eso desalentaba la realización del proyecto; sin embargo, la asesoría de expertos en el tema incentivó a la continuación del proceso de investigación ya que mostraron interés por una posible implementación de nuevos aplicativos para escanear vulnerabilidades, donde cada uno pensaba en las facilidades que le aportarían dichos servicios al desarrollo de actividades que se realizan dentro de los ámbitos de trabajo que conciernen a una organización como tal.

Al realizar el proceso anterior se cumplió a plenitud todos los objetivos específicos propuestos dando como resultado: un aplicativo cliente-servidor para detectar vulnerabilidades en smartphones Android utilizando una herramienta de escaneo, con las respectivas pruebas del aplicativo.

En otras investigaciones se describieron procesos similares al detallado en este trabajo, las semejanzas se ven en los pasos de conexión y configuración de aplicativos para detectar vulnerabilidades, clasificación de herramientas de escaneo para el análisis de puertos TCP y UDP en escucha. Dichas investigaciones hicieron estos pasos para proponer la implementación de aplicativos computacionales que garanticen la seguridad de sistemas empresariales, lo cual contrasta con la presente investigación que realizó un proceso parecido para implementar un aplicativo para detectar vulnerabilidades pero basado en Android.

El desarrollo de este proyecto se hizo importante y necesario en la medida de ser Android uno de los sistemas operativos móviles más usados en el mundo. Android es utilizado por empresarios para realizar sus actividades diarias sin necesidad de utilizar un computador físico, y es importante que el dispositivo utilizado por dicha persona no sea vulnerable, debido a que maneja información confidencial que puede ser pan caliente para piratas informáticos. La búsqueda de información, conocimiento y nuevos retos es uno de los aspectos que hace al ser humano una gran creación de Dios, por lo tanto, la propuesta está hecha y la invitación que se deja es seguir con el proceso de investigación con respecto a aplicativos que detecten vulnerabilidades en sistemas Android, pues es un sistema operativo muy utilizado y este debe ser el primer paso para mostrar a futuras generaciones la importancia de investigar sobre tecnologías y servicios que contribuyen a la continuidad y progreso de naciones.

De los resultados inesperados de la investigación se concluyó que hay pocas evidencias para determinar que las estrategias usadas durante el proyecto fueron eficientes para equilibrar la detección de vulnerabilidades en Android. Aunque los diseños fueron aprobados por expertos en el tema, los materiales relacionados con las estrategias fueron creados por los investigadores de este proyecto afectando en parte la calidad de éstas.

7. RECOMENDACIONES

Una de las limitaciones de la investigación fue la no existencia de un aplicativo que conectara una herramienta de escaneo con sistemas Android. Otra limitación fue la falta de una guía sobre cómo realizar una conexión entre 2 sistemas basados en 2 lenguajes de programación diferentes para trabajar en conjunto.

Una estrategia para solucionar las limitaciones mencionadas anteriormente fue revisar minuciosamente el código de la herramienta de detección de vulnerabilidades para amoldarlo a las necesidades del aplicativo en Android, de tal manera que existiera una conexión entre ellos para lograr con éxito el objetivo. La anterior recomendación sería una mejora para la realización del cuarto objetivo específico.

Para trabajos futuros se recomienda agregar funcionalidades al aplicativo teniendo en cuenta otras variables que determinen un resultado con mayor información sobre vulnerabilidades. Otra recomendación es agregar la descripción en multiidioma para facilitar la interpretación de resultados en diferentes contextos a nivel mundial.

8. REFERENCIAS BIBLIOGRÁFICAS

[1]. Sybase Business Intelligence Solutions. *Sybase Enterprise Solutions for Android* [En línea] [fecha de consulta: 18 de agosto de 2012]. Disponible en: <http://www.sybase.com.mx/mobilize/mobile-device-management/android>

[2].Ipass Inc, *The iPass Global Mobile Workforce Report* [En línea]. 16 de noviembre de 2011 [fecha de consulta: 19 de agosto de 2012], There are Apples and Androids in the Enterprise. Disponible en:< http://www.wballiance.com/wba/wp-content/uploads/downloads/2012/07/ipass_mobileworkforcereport_q4_2011.pdf>

[3]. Eset Latinoamérica, *Seguridad en dispositivos móviles* [En línea]. 20 de octubre de 2009 [fecha de consulta: 19 de agosto de 2012]. Disponible en: <http://www.psicofxp.com/forums/seguridad-informatica.47/971041-seguridad-en-dispositivos-moviles.html>

[4].Vicente, M. (2011). *Malware en dispositivos móviles Android.* Universidad de Almería, España.

[5]. Hernández, J. Portillo, M. *Android, más vulnerable a ser atacada* [En línea]. 4 de junio de 2012. [Fecha de consulta: 19 de agosto de 2012].Diario el Mundo de El salvador. Disponible en:< http://elmundo.com.sv/android-mas-vulnerable-a-ser-atacada>

[6]. Laboratorios McAfee® Labs™.*Informe de McAfee sobre amenazas: tercer trimestre de 2011* [Fecha de consulta: 19 de agosto de 2012]. Estadísticas de malware para dispositivos móviles. Disponible en: < http://www.mcafee.com/es/resources/reports/rp-quarterly-threat-q3-2011.pdf?cid=WBB042>

[7]. Excélsior, *IBM anunció nuevo software para fortalecer seguridad de dispositivos* [En línea]. 10 de junio de 2012. [Fecha de consulta: 19 de agosto de 2012].Diario Excélsior de México. Disponible en: http://www.excelsior.com.mx/index.php?m=nota&seccion=&cat=380&id_nota=840347

[8]. DuoSecurity, *Comprueba las vulnerabilidades de tu Android con X-Ray* [En línea]. 29 de julio de 2012. [Fecha de consulta: 19 de agosto de 2012]. Disponible en: http://planetamoviles.com/comprueba-las-vulnerabilidades-de-tu-android-con-x-ray/

[9]. Tecnovirus, antivirus para dispositivos móviles Android [fecha de consulta:22 de agosto de 2012]. Disponible en: http://www.tecnovirus.com/blog/4-de-los-mejores-antivirus-para-android/

[10].Y.Zhou, X.Jiang (2012) *Dissecting Android Malware: Characterization and Evolution* (pp. 95-109) . Disponible en: IEEE Symposium on Security and Privacy.

[11]. W. Berhane Tesfay,T. Booth,K. Andersson (2012). *Reputation Based Security Model for Android Applications* (pp. 896-901). Disponible en: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)

[12]. A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, C. Glezer (2010). *Google Android: A Comprehensive Security Assessment* (pp. 35-44). Disponible en: IEEE Security and Privacy

[13]. T. Wei,C. Mao,A.B. Jeng,H. Lee,H. Wang,D. Wu (2012). *Android Malware Detection via a Latent Network Behavior Analysis* (pp. 1251-1258). Disponible en: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom).

[14]. M. Alazab,V. Monsamy,L. Batten,P. Lantz,R. Tian (2012). *Analysis of Malicious and Benign Android Applications* (pp. 608-616). Disponible en: 2012 32nd International Conference on Distributed Computing Systems Workshops (ICDCS Workshops)

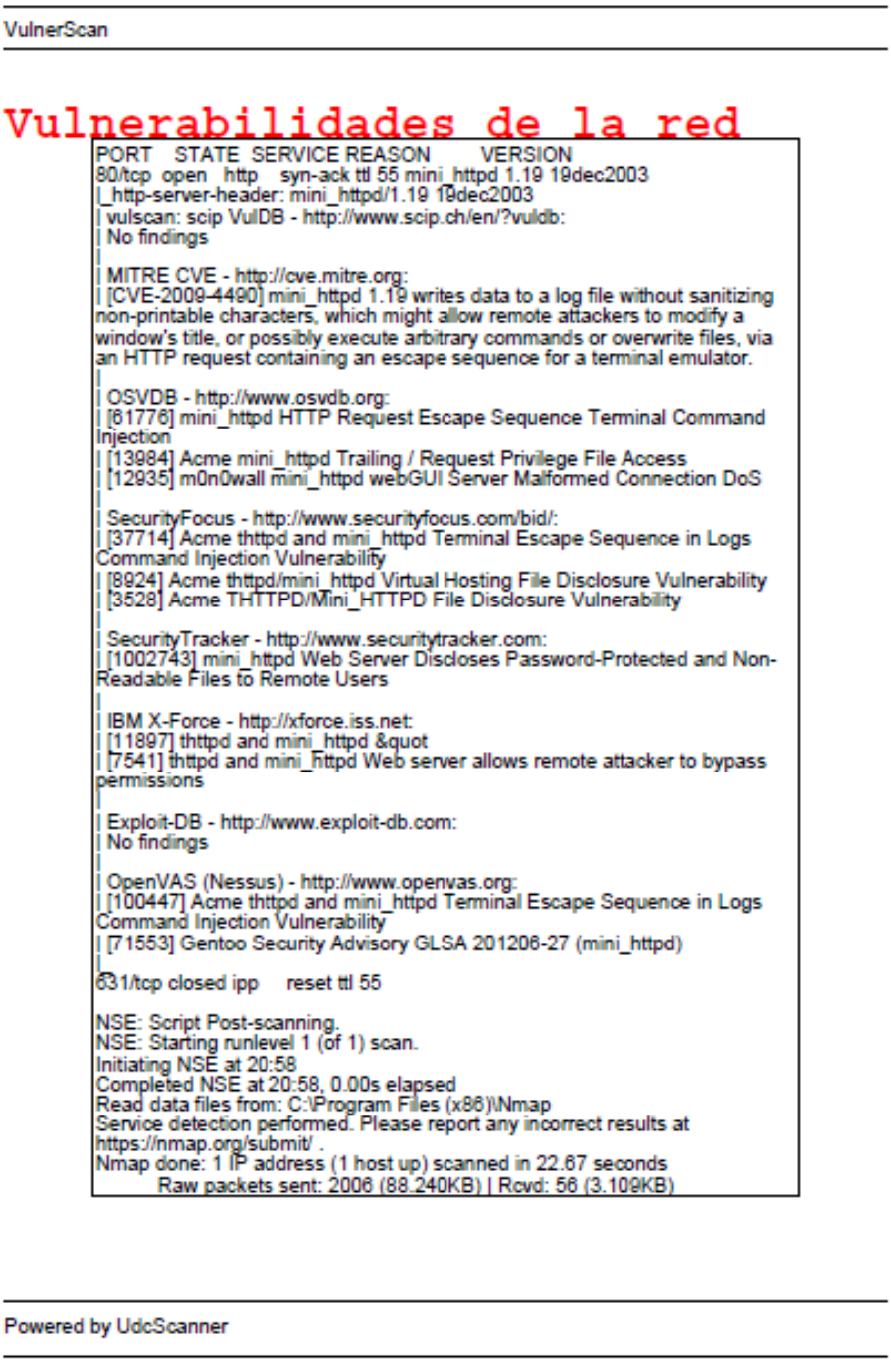
[15]. C. Kilinc,T. Booth,K. Andersson (2012). *WallDroid: Cloud Assisted Virtualized Application Specific Firewalls for the Android OS* (pp. 877-883). Disponible en: 2012

IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), disponible en: http://redyseguridad.fi-p.unam.mx/proyectos/tsi/capi/Cap2.html

[16] Franco, D. A., Perea, J. L., & Tovar, L. C. (2013). Herramienta para la Detección de Vulnerabilidades basada en la Identificación de Servicios.*Información tecnológica*, *24*(5), 13-22.Dispomible en: http://www.scielo.cl/scielo.php?pid=S0718-07642013000500003&script=sci_arttext

9. ANEXOS

Anexo 1: Reporte de pruebas.



VulnerScan

Vulnerabilidades de la red

```
PORT    STATE  SERVICE REASON       VERSION
80/tcp  open   http    syn-ack ttl 55 mini_httpd 1.19 19dec2003
|_http-server-header: mini_httpd/1.19 19dec2003
| vulscan: scip VulDB - http://www.scip.ch/en/?vuldb:
| No findings
|
| MITRE CVE - http://cve.mitre.org:
| [CVE-2009-4490] mini_httpd 1.19 writes data to a log file without sanitizing
non-printable characters, which might allow remote attackers to modify a
window's title, or possibly execute arbitrary commands or overwrite files, via
an HTTP request containing an escape sequence for a terminal emulator.
|
| OSVDB - http://www.osvdb.org:
| [61776] mini_httpd HTTP Request Escape Sequence Terminal Command
Injection
| [13984] Acme mini_httpd Trailing / Request Privilege File Access
| [12935] m0n0wall mini_httpd webGUI Server Malformed Connection DoS
|
| SecurityFocus - http://www.securityfocus.com/bid/:
| [37714] Acme thttpd and mini_httpd Terminal Escape Sequence in Logs
Command Injection Vulnerability
| [8924] Acme thttpd/mini_httpd Virtual Hosting File Disclosure Vulnerability
| [3528] Acme THTTPD/Mini_HTTPD File Disclosure Vulnerability
|
| SecurityTracker - http://www.securitytracker.com:
| [1002743] mini_httpd Web Server Discloses Password-Protected and Non-
Readable Files to Remote Users
|
| IBM X-Force - http://xforce.iss.net:
| [11897] thttpd and mini_httpd &quot
| [7541] thttpd and mini_httpd Web server allows remote attacker to bypass
permissions
|
| Exploit-DB - http://www.exploit-db.com:
| No findings
|
| OpenVAS (Nessus) - http://www.openvas.org:
| [100447] Acme thttpd and mini_httpd Terminal Escape Sequence in Logs
Command Injection Vulnerability
| [71553] Gentoo Security Advisory GLSA 201206-27 (mini_httpd)
|_
631/tcp closed ipp     reset ttl 55

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 20:58
Completed NSE at 20:58, 0.00s elapsed
Read data files from: C:\Program Files (x86)\Nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.67 seconds
        Raw packets sent: 2006 (88.240KB) | Rcvd: 56 (3.109KB)
```

Powered by UdcScanner

Ilustración 12. Reporte de vulnerabilidades.