UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

Pruebas de Penetración a Smartphones Android e iOS

TESIS

PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

ERICK MINERO REYES

DIRECTOR DE TESIS:

ING. ALDO JIMÉNEZ ARTEAGA

MÉXICO, D.F., CIUDAD UNIVERSITARIA, 2015

Dedicatoria y Agradecimiento

El principal agradecimiento es para mis padres, que todo este tiempo han estado conmigo y siempre he recibido su apoyo de más de una forma, sinceramente no tengo con que pagarles todo lo que han hecho por mí, porque no solo me han dado la vida, sino me han enseñado a como tener una buena vida.

A la maestra Jaquelina quiero darle las gracias por todo el apoyo que me ha mostrado, desde el 2012 ha sido una maestra y un apoyo para mí, nunca olvidaré ese apoyo; espero siempre su vida siga llena de bendiciones por la excelente persona y maestra que es usted.

Finalmente a mi director, profesor, maestro y amigo Aldo, me has enseñado que es ser un buen ingeniero y a pensar como tal, siempre hemos tenido de que hablar y que trabajar, espero este no sea el último trabajo en el que pueda colaborar contigo.

Índice

CAPÍTULO I

Auditorías de Seguridad y Arquitectura de los Sistemas Operativos para Móviles

Introducción

Los dispositivos móviles constituyen actualmente un gran y creciente grupo de computadoras que facilitan la realización de tareas al ser humano. Su portabilidad y flexibilidad los ha convertido en herramientas indispensables dentro de la vida de toda la gente; actualmente estos forman parte de prácticamente cualquier actividad humana conocida.

La categoría de dispositivos móviles es muy amplia. En ésta se encuentran los teléfonos inalámbricos, cámaras digitales, tabletas electrónicas, teléfonos inteligentes, entre otros. El presente documento se enfocará en uno de estos grupos: los teléfonos inteligentes o también conocidos como smartphones.

Los smartphones se han convertido en un amplio y popular mercado en el mundo, debido a que fusionan los servicios de telefonía celular con servicios, como son el acceso a internet, reproductores de contenido multimedia, cámaras fotográficas y de video, GPS; lo que les otorga un poder de cómputo mayor que el de un teléfono celular convencional. Una característica esencial de los teléfonos inteligentes es que poseen un sistema operativo (SO) que administra todas las funciones del dispositivo, muy parecido a las computadoras de escritorio y laptops.

En los últimos años una gran cantidad de estos sistemas operativos han aparecido dentro del mercado. Actualmente son dos sistemas operativos los que están presentes en más del 85% de smartphones en el mundo: Android e iOS, que pertenecen a Google y Apple, respectivamente. En la Figura 1 se aprecia cómo se han desarrollado los sistemas operativos para móviles en los últimos años; Android no siempre fue el sistema operativo con la mayor presencia o el más seguro. Esto último se abordará en la sección de Android.

La presencia de los smartphones ha traído una serie de enormes ventajas con las que no se contaba algunos años atrás; éstas han beneficiado muchas facetas de la vida cotidiana del ser humano, porque permiten tener acceso, procesar y compartir información prácticamente en cualquier lugar, creando versatilidad, rapidez y precisión entre otras ventajas al realizar dichas tareas.

Lamentablemente, la presencia de smartphones presenta una problemática muy importante en lo que respecta a su administración, sobre todo dentro del campo laboral. Esto es debido a que su gran flexibilidad y portabilidad hacen de su control y administración una tarea difícil; sin embargo debe ser considerada importante, pues los smartphones pueden poseer información valiosa, que de perderse o ser robada por terceros podría constituir pérdidas financieras dentro del proceso de negocio de una organización. Aunado a esto, deben considerarse todas las fallas de seguridad que presentan estos dispositivos y que

podrían ser usadas por un tercero malintencionado para afectar a una organización o a un individuo.

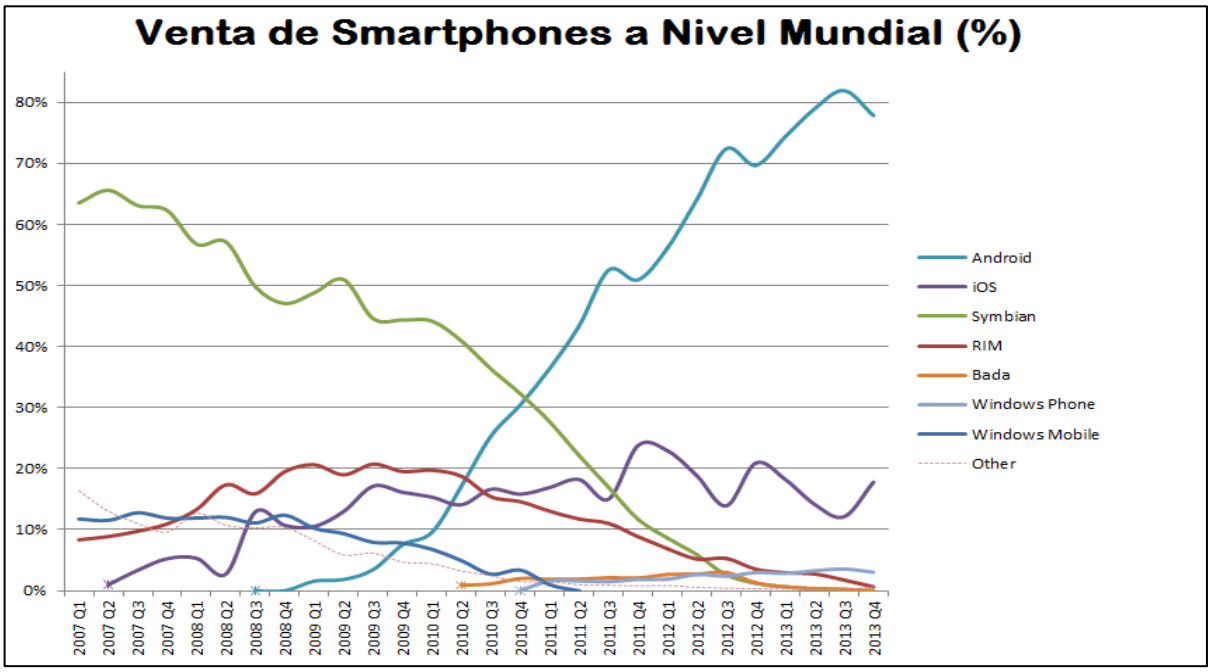


Figura 1. Venta porcentual de smartphones a nivel mundial

Fuente: Gatner. Imagen tomada de Wikipedia en Inglés [1]

Una buena administración lógica resulta indispensable, sobre todo en organizaciones en donde la información es su activo más valioso, como suelen ser los bancos; por ello; constantemente estas organizaciones deben estar revisando la manera en que realizan sus tareas para poder encontrar los puntos débiles y mejorarlos una vez localizados; estos deben abarcar todo aspecto de la tarea en sí misma: logística, tiempo, seguridad, entre otros aspectos. A estas revisiones se les llama auditorias; éstas pueden evaluar uno o varios puntos, pero es muy importante que se apeguen a estándares de calidad, que garanticen el hecho de que las tareas se están realizando de la manera más eficiente posible.

Las auditorías son compuestas por varias fases; la cantidad de fases depende de qué se esté verificando y la complejidad de la misma. Es importante que dentro de estas fases exista una buena comunicación entre el prestador de servicios y el cliente, porque de ella depende que el prestador de servicios realice solo las tareas que el cliente desee de la manera correcta en el tiempo establecido; de otra forma, se pudiesen estar realizando tareas innecesarias, o peor aún, que éstas afecten las actividades normales de la organización.

El presente documento pretende servir como guía para conocer las fallas más comunes de seguridad en los smartphones, sentando las bases teóricas de los procesos más comunes que tienen que ver con la evaluación de la seguridad. Se harán varias demostraciones prácticas, explicadas paso a paso y cómo éstas pueden formar parte del

proceso de auditoría de seguridad. El documento consta de 3 partes: La primera abordará las nociones teóricas básicas de Android, iOS y el proceso que se lleva a cabo durante una auditoría de seguridad. La segunda, se centrará en las prácticas demostrativas de Android; tomando en cuenta la teoría del proceso y las condiciones para su realización. La tercera, se centrará en las prácticas demostrativas de iOS, conservando las mismas características usadas en la segunda sección.

Arquitectura de Android

Android es actualmente un sistema operativo basado en el kernel de Linux desarrollado y mantenido por Google. Este sistema se encuentra principalmente en dispositivos móviles como tabletas electrónicas o smartphones, aunque no son los únicos; versiones de Android son usadas en televisiones, automóviles, relojes y lentes de realidad aumentada.

Android es el sistema operativo mayormente usado en los smartphones. En 2013 este sistema operativo ocupaba el 80% de todo el mercado internacional (Figura 1). En México también existe una importante presencia de este sistema operativo, como podemos ver en la Tabla 1.

Tabla 1. Presencia de los SO móviles en México en el año 2013

Sistema Operativo	Porcentaje %
Android	52.58%
iOS	36.43%
Windows Phone	7.23%
BlackBerry	1.82%

Fuente: GuiaLocal.com.mx [2]

Este sistema operativo tuvo sus inicios en 2005 como un programa para controlar cámaras fotográficas; sin embargo, en sus 10 años de vida, ha sufrido muchos cambios, sobre todo en el ámbito de seguridad. Para entender mejor este último aspecto, se hará un resumen sobre la arquitectura de este sistema operativo.

Arquitectura

Como se comentó en un inicio, Android es un sistema operativo basado en el kernel de Linux, sin embargo no es lo único que lo compone. Android está compuesto por 4 capas o niveles de abstracción. En la Figura 2 se aprecia un esquema de los principales componentes de cada capa de la arquitectura de Android.

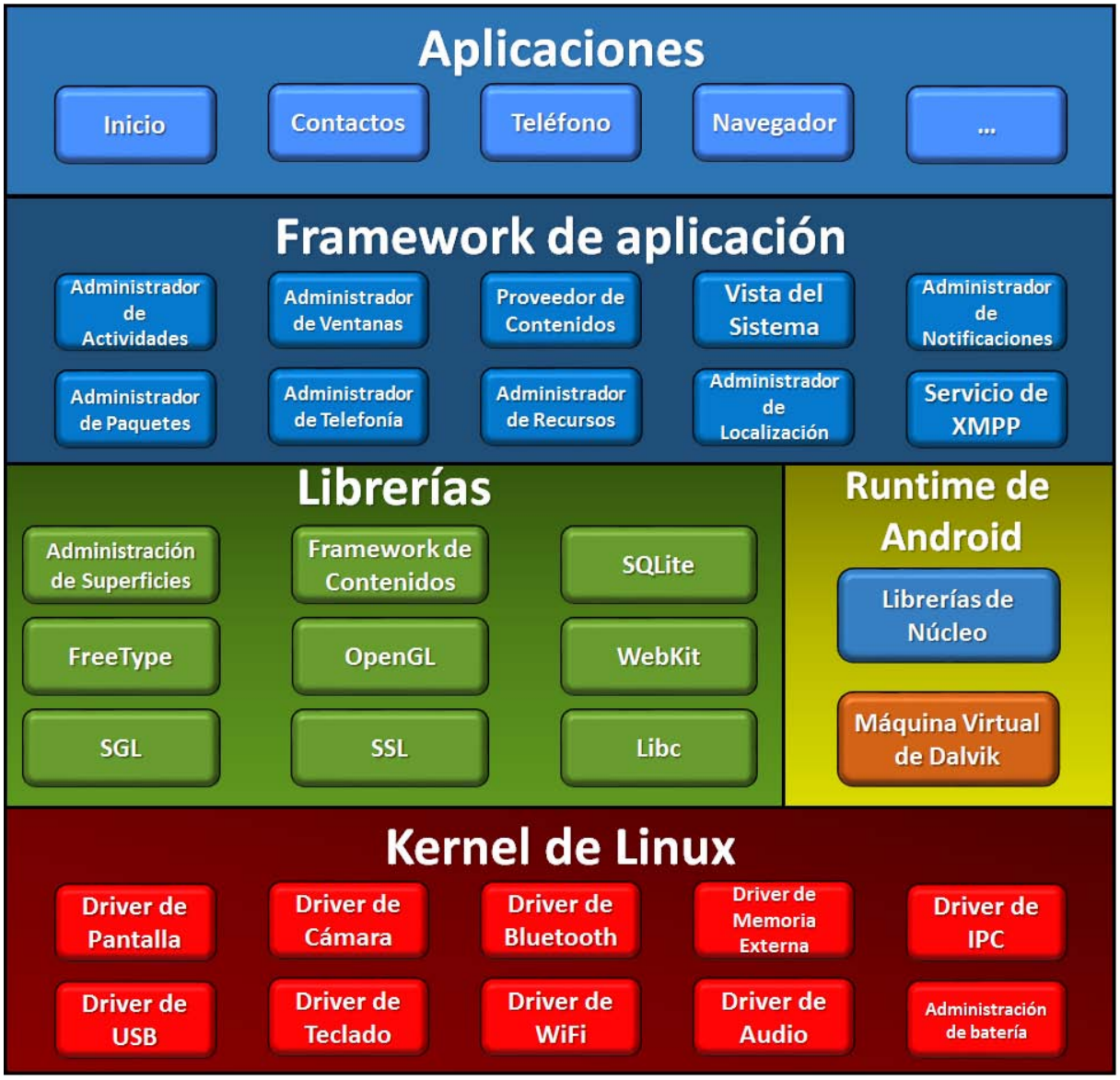


Figura 2. Capas de la arquitectura de Android

Fuente: Android Team. Traducida.

1. **Kernel de Linux:** Es la capa de abstracción más baja de Android. En un sentido estricto no es exactamente igual que el kernel (núcleo) presente en las distribuciones de Linux. El kernel de Android ha sido optimizado para su uso en dispositivos móviles, esto es para evitar librerías que no son usadas en ellos, evitar que el sistema se suspenda completamente y en el manejo de los componentes de hardware, como suele ser la pantalla, cámara, etc. Además se encarga de la gestión de procesos y memoria que es usada por el smartphone. En esta capa se encuentra la parte más robusta de la seguridad en Android. Este concepto resulta importante porque será frecuentemente mencionado en las prácticas posteriores.

   En la Tabla 2 se observan las versiones del kernel que usan las versiones de Android.

Tabla 2. Versiones de Android y sus respectivas versiones del kernel de Linux

Versión de Android	Versión del kernel de Linux
1.5	2.6.27
1.6	2.6.29
2.0-2.1	2.6.29
2.2	2.6.32
2.3.X	2.6.35
3.X	2.6.36
4.X	3.0.1
4.1-4.3	3.0.31
4.4	3.14
5.X	3.14-3.4

Fuente: Learning Pentesting for Android Devices [4]

2. **Librerías:** Éstas librerías de capa 2 son las principales para el sistema operativo, están escritas en los lenguajes C y C++, se encargan de administrar tareas básicas como suelen ser la administración de ventanas, bases de datos, el uso del navegador, la administración de video y, en general en todos los servicios que ofrece el smartphone.

   Las principales librerías que componen esta capa son las siguientes:

   a. **Libc:** Incluye todas las cabeceras y funciones estándar del lenguaje C. Todas las demás librerías usan las cabeceras de libc con la finalidad de dar uniformidad al código.

   b. **Administración de Superficies (Surface Manager):** Compone los diferentes elementos de navegación de la pantalla. También administra las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

   c. **OpenGL y SGL:** Son las librerías gráficas; administran la capacidad gráfica de Android. OpenGL maneja gráficos en 3D y permite utilizar el hardware encargado de proporcionar gráficos en 3D. SGL proporciona gráficos en 2D, por lo que es la librería más utilizada por la mayoría de las aplicaciones. Ambas librerías pueden ser usadas en una misma aplicación de Android.

   d. **Framework de Contenidos (Media Framework):** Son los códec necesarios para el contenido multimedia en Android (audio, video, entre otros).

   e. **FreeType:** Es una librería que permite el uso y administración de fuentes en Android.

f. **SSL:** Permite el uso del protocolo SSL para establecer comunicaciones seguras entre los distintos servicios de Android.

g. **SQLite:** Librería simplificada de SQL que permite la creación y administración de bases de datos.

h. **WebKit:** Es un motor para las aplicaciones del navegador web de Android.

3. **Runtime de Android:** El runtime de Android funciona mediante el uso de una máquina virtual llamada Dalvik. Esta máquina virtual es la encargada de correr los programas ejecutados en Android de la manera más óptima posible. También se encuentran todas las librerías del núcleo programadas en Java para darle soporte a la máquina virtual y el desarrollo de aplicaciones dándole, portabilidad y consistencia al sistema.

4. **Framework de aplicación:** En esta capa se encuentran presentes todas las herramientas de desarrollo en Android. Esta capa se encarga de dar uniformidad a las herramientas disponibles sin importar si es Google, un desarrollador de terceros o el propio usuario; todas las aplicaciones desarrolladas en Android usan el mismo conjunto de APIs de capa 3.

   Las APIs más importantes de esta capa son:

   a. **Administrador de Actividades (Activity Manager):** API que administra las aplicaciones de Android durante todas sus etapas.

   b. **Administrador de Ventanas (Window Manager):** Utiliza la librería de *Administración de Superficies (Capa 2)* para controlar la forma en que las aplicaciones de Android utilizan la pantalla.

   c. **Administrador de Telefonía (Telephone Manager):** Es un conjunto de APIs que controlan las funciones básicas del teléfono como son llamadas, mensajes SMS, etc.

   d. **Proveedor de Contenidos (Content Provider):** Esta API está encargada de compartir datos con el resto de aplicaciones. Estos datos pueden ser acceso a la agenda, mensajes, estado de algún controlador, entre otros.

   e. **Vista del Sistema (View System):** Es un conjunto de APIs que permite crear interfaces de usuario (GUI), incluye tanto vistas estándar usadas

frecuentemente, como herramientas para construir propias (botones, mosaicos, acceso al teclado, etc.).

f. **Administrador de Localización (Location Manager):** API encargada del uso del Sistema de Localización y Posicionamiento Global (GPS, por sus siglas en inglés).

g. **Administrador de Notificaciones (Notification Manager):** API encargada de unificar el uso de notificaciones del usuario, es usada por las aplicaciones para acciones como: llamada entrante, mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc.

h. **Servicio de XMPP (XMPP Service):** API que permite utilizar el protocolo XMPP para el intercambio de mensajes en XML.

5. **Aplicaciones:** En esta capa se encuentran todas las aplicaciones que son utilizadas por los usuarios, también se encuentras las aplicaciones de los proveedores.

Máquina Virtual de Dalvik

Todas las aplicaciones de Android corren bajo esta máquina virtual. La DVM (Dalvik Virtual Machine) se encuentra en todas las versiones de Android hasta el momento, sin embargo; a partir de la versión 4.4 (Kit-Kat) se introduce una nueva máquina virtual llamada Android Runtime (ART), la cual en un futuro reemplazara a la DVM (se encuentra por default en la versión 5.0). Muy parecido a la máquina virtual de java, la DVM provee un entorno que le da portabilidad y consistencia a los programas. En la Figura 3 se pueden ver los pasos esenciales de ambos procesos.

El proceso que sigue la compilación de la DVM es la siguiente:

1. Se genera el código fuente en el lenguaje Java (extensión .java).
2. Posteriormente se compila y se genera el bitcode de Java (extensión .class).
3. Se vuelve a compilar el bitcode con el compilador de Dalvik que en este caso utiliza la extensión .dex (Dalvik EXecutable) o .odex (Optimized Dalvik Executable).

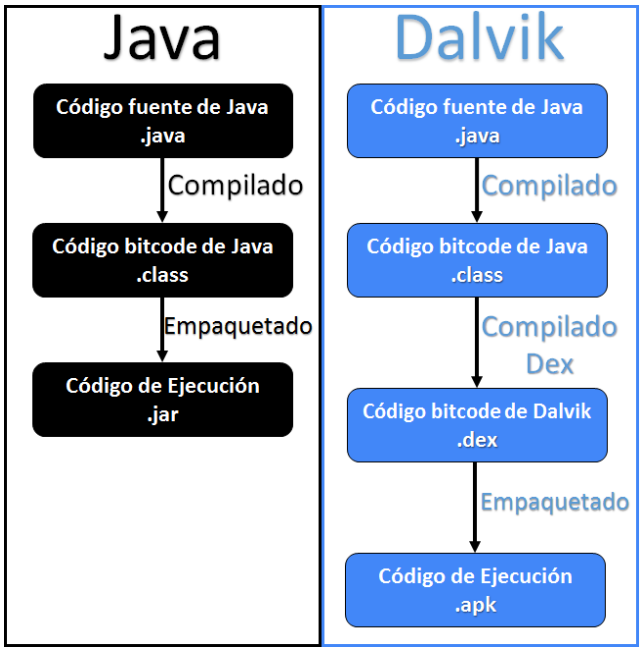


Figura 3.Diferencias entre la compilación de Java y Android.

4. Finalmente se crea la aplicación con extensión .apk.

Para cada ejecución de una aplicación, se requiere una instancia de la máquina virtual; el dispositivo soporta la ejecución de varios programas en la máquina virtual y utiliza un intérprete que optimiza el uso del CPU y la memoria mediante un registro de procesos y un set de instrucciones optimizado. A pesar de todas estas tareas que realiza la máquina virtual, su función principal no es proveer seguridad al sistema, más bien se centra en cuestiones como son la portabilidad y la eficiencia de los recursos de Android.

Seguridad

La seguridad que ofrece Android ha crecido de manera gradual; dependiendo de la versión, se han desarrollado diversas medidas de seguridad. Se mencionarán las medidas de seguridad de Android en función de su capa de abstracción.

1. **Seguridad en el Kernel (Capa 1**): El kernel de Linux como bien es sabido, es código abierto; en otras palabras, cualquiera puede mejorarlo y optimizarlo. La ventaja de esto, consiste en que cualquier vulnerabilidad del kernel es prontamente corregida, por lo que el núcleo de Android adquiere esa misma ventaja. En la actualidad, existen pocos fallos que aprovechan una vulnerabilidad relacionada con el kernel de Android, por lo que limita mucho la cantidad de dispositivos afectados por un problema de seguridad. Como medida añadida a la seguridad, existe la fragmentación de Android.

   La fragmentación consiste en que cada versión de Android, está basada en una versión particular del kernel de Linux, esto afecta completamente la forma en que trabaja el smartphone entre una versión y otra; es una ventaja porque una vulnerabilidad que afecte a una versión en particular, no afectará a otra. Son por estas razones que resulta importante saber la distribución de las versiones de Android cuando se realiza una auditoría de seguridad u otro proceso en el que se requiera conocer este hecho, porque cada versión tendrá que ser procesada de manera distinta. En la figura 4 se observa los porcentajes de presencia de las

Versión	Nombre	API	Distribución
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	4.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	4.1%
4.1.x	Jelly Bean	16	13.0%
4.2.x		17	15.9%
4.3		18	4.7%
4.4	KitKat	19	39.3%
5.0	Lollipop	21	15.5%
5.1		22	2.6%

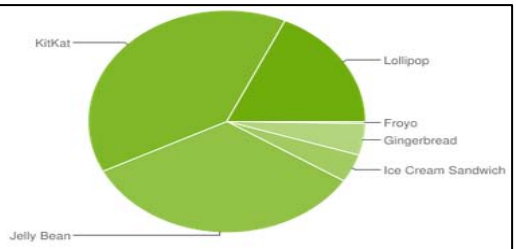


Figura 4. Distribución de las versiones de Android en julio del 2015.
Fuente: Android Developer [3]

versiones de Android más recientes.

2. **Seguridad en Librerías y el Runtime (Capa 2**): Esta es la Sandbox (caja de arena). Esta herramienta prohíbe que una aplicación (Capa 4) pueda obtener recursos o permisos que no fueron previamente otorgados durante la instalación, limitando hasta cierto punto, las capacidades, alcance y el daño que esta pudiera causar. Un problema con esta medida de seguridad, es la falta de información sobre las implicaciones que tienen los permisos que son aceptados por el usuario; en otras palabras, durante la instalación de una aplicación, no se dan detalles de exactamente qué recursos utiliza la misma. Sin embargo, aunque los permisos se mostraran de manera explícita, es poco probable que el usuario común comprenda a cabalidad los mismos, terminaría por aceptar los permisos y esto causaría que su smartphone fuera vulnerable por una aplicación. Esto reduce en gran medida su efectividad, por lo que no es propiamente una medida de seguridad óptima. En la Figura 5 se aprecia una ventana típica con los permisos que una aplicación tiene, una vez ya instalada.

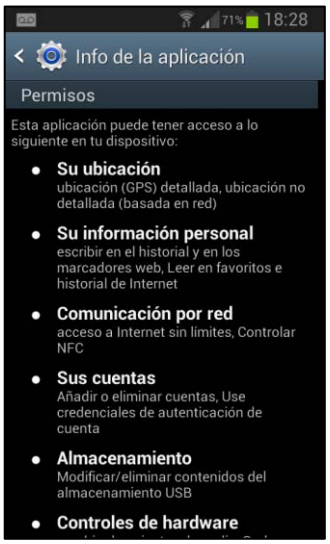


Figura 5. Los permisos que solicita un App pudieran no ser los correctos.

3. **Seguridad en el Framework de Aplicación (Capa 3**): Esta capa existe para unificar las herramientas de desarrollo de aplicaciones. Esta unificación permite que las APIs no contengan funciones, o combinación de ellas; que causen un comportamiento dañino al sistema en una aplicación determinada. Esto limita parcialmente el desarrollo de aplicaciones maliciosas en Android. Esta medida puede ser evadida, si se programa sin el uso de las APIs del Framework de Aplicación.

4. **Seguridad en las Aplicaciones (Capa 4):** Estas pueden ser propias de Google o desarrolladas por terceros. Por parte de Google, por default no se permite la instalación de aplicaciones ajenas a la Play Store, no tiene habilitada la depuración por USB y ha desarrollado aplicaciones para verificar el comportamiento de otras. Una de estas se llama “Android Verify Apps” (Versiones 4.1-4.3), verifica y detecta los procesos que corren por background (no ejecutadas por el usuario), en busca de comportamientos sospechosos.

En aplicaciones desarrolladas por terceros se pueden encontrar: antivirus, protectores de información en la nube, localizadores GPS, analizadores de procesos, entre otros. Aunque muchas de estas son eficientes, solo cubren parcialmente la seguridad del teléfono, sin mencionar que no pueden cubrir fallas de seguridad presentes en otras aplicaciones que el dispositivo tenga instalado; estas pudieran ser

permisos excesivos, puertas traseras, susceptibilidad a desbordamientos, etc., sin mencionar que consumen recursos de sistema, lo que afecta su desempeño con el usuario final.

Arquitectura de iOS

iOS es el sistema operativo basado en una versión optimizada de OS X, sistema operativo de las computadoras iMac, ambos desarrollados y mantenidos por Apple. Éste sistema operativo tiene la particularidad que solo funciona en dispositivos propietarios de Apple, y no permite su instalación en dispositivos patentados por terceros. Actualmente se encuentra en toda la familia de dispositivos móviles Apple: iPod Touch, iPad, iPad Mini, Apple TV e iPad Air. Este es el segundo sistema operativo mayormente usado en el mundo; tenía una presencia cerca del 20% del mercado a finales del 2013. Durante el mismo año en México, iOS representaba el 36.43% del mercado.

Este sistema operativo tuvo sus inicios en 2007 bajo el nombre iPhone OS, que originalmente fue diseñado para la primera versión del iPhone, este lanzamiento de Apple tuvo tanto éxito que provocó la aparición de nuevos dispositivos de la corriente móvil: iPod Touch, iPad, Apple TV y las versiones posteriores de los iPhone, todas ellas con el mismo sistema operativo.

iPhone OS cambió su nombre a iOS en junio del 2010, junto con la aparición del iPad. Sin embargo, esto constituyó un problema para Apple, porque IOS (Internetnetwork Operating System) es el nombre del sistema operativo presente en los Routers y Switches Cisco; esto llevó a Apple a pagar los derechos de uso comercial de este nombre a Cisco a partir de ese año, en adelante.

OS X e iOS

OS X es el sistema operativo de Apple para sus computadoras de sobremesa, este sistema se encuentra basado en Darwin (sistema operativo basado en la familia de los BSD); de OS X Apple desarrolló una versión para sus dispositivos móviles: iOS.

Al igual que en el caso de Linux, Darwin es un sistema operativo de código abierto, lo que permite que pueda ser mejorado independientemente del desarrollador. En el caso de OS X, utiliza el mismo núcleo que Darwin, por lo tanto el kernel de OS X es código abierto, pero el resto del sistema operativo es código cerrado. Este hecho se vuelve importante cuando se requiere buscar información técnica sobre el sistema operativo.

En la Figura 6 se observa un esquema simplificado de la historia del desarrollo de los BSD y su relación que tienen estos con Darwin, OS X e iOS.

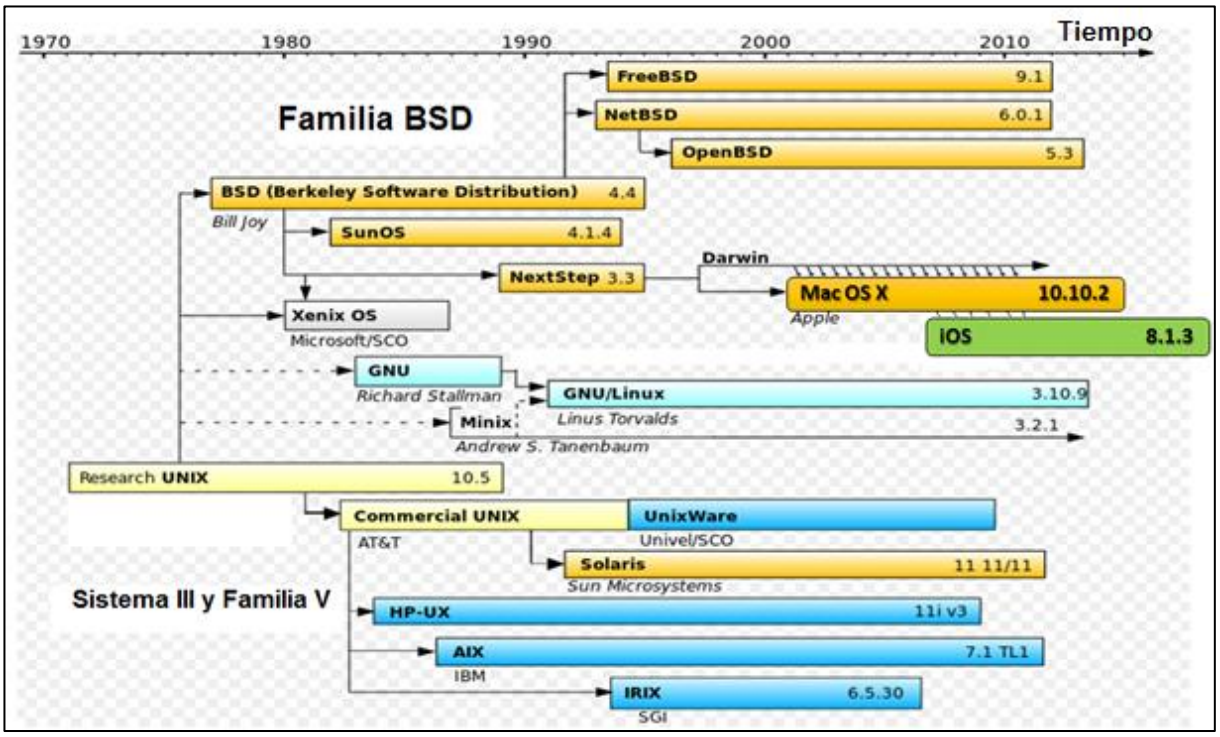


Figura 6. Historia de los SO BSD y su relación con OS X e iOS.

Fuente: Wikipedia en Inglés [10].Traducida.

iOS y OS X guardan muchas similitudes en cuanto a su diseño y el uso de tecnologías de desarrollo, pero tienen varias diferencias importantes en cuanto a arquitectura se refiere. Las más importantes son:

- OS X está diseñado para arquitecturas Intel, mientras que iOS para arquitecturas del tipo ARM (RISC). Esta es una diferencia importante, porque el hardware es utilizado de manera distinta entre OS X e iOS, pues los recursos de los dispositivos móviles son menores a las computadoras de sobremesa, por eso utiliza la arquitectura ARM. Es una cuestión que debe ser considerada al momento de desarrollar aplicaciones en este sistema operativo.

- iOS es código cerrado completamente, mientras que OS X parcialmente lo es. A pesar de que iOS es completamente código cerrado, es posible encontrar información relacionada con iOS sobre la estructura del sistema, abordando OS X o Darwin, sistemas en los cuales se basa.

- El kernel es compilado de manera diferente, sobre todo en la cuestión de funciones integradas (administración del hardware); esto es por el uso de arquitecturas diferentes entre los sistemas operativos y el uso de frameworks como base.

- En cuestión de línea de comandos (shell) iOS puede ser accedido de manera remota mediante SSH.

- La administración de memoria es más estricto en iOS. Este hecho debe ser considerado por los desarrolladores, el uso eficiente de la memoria es importante en iOS porque esta es menor en comparativa con OS X.

- iOS es un sistema más complejo en la administración de usuarios y permisos. Solo permite la instalación de Apps oficiales de la App Store de Apple, prohíbe privilegios de administrador (root) y todas las Apps trabajan bajo el uso de una caja de arena estricta.

- Algunas de las carpetas de sistema tienen un nombre distinto al de OS X, también cuando al dispositivo se le realiza Jailbreak (capítulo 3), más carpetas de sistema son agregadas.

Arquitectura

Al igual que Android, la arquitectura de iOS está divida por capas y cada una de estas se encarga de un conjunto de tareas en específico. En la Figura 7 se aprecia la arquitectura de iOS:

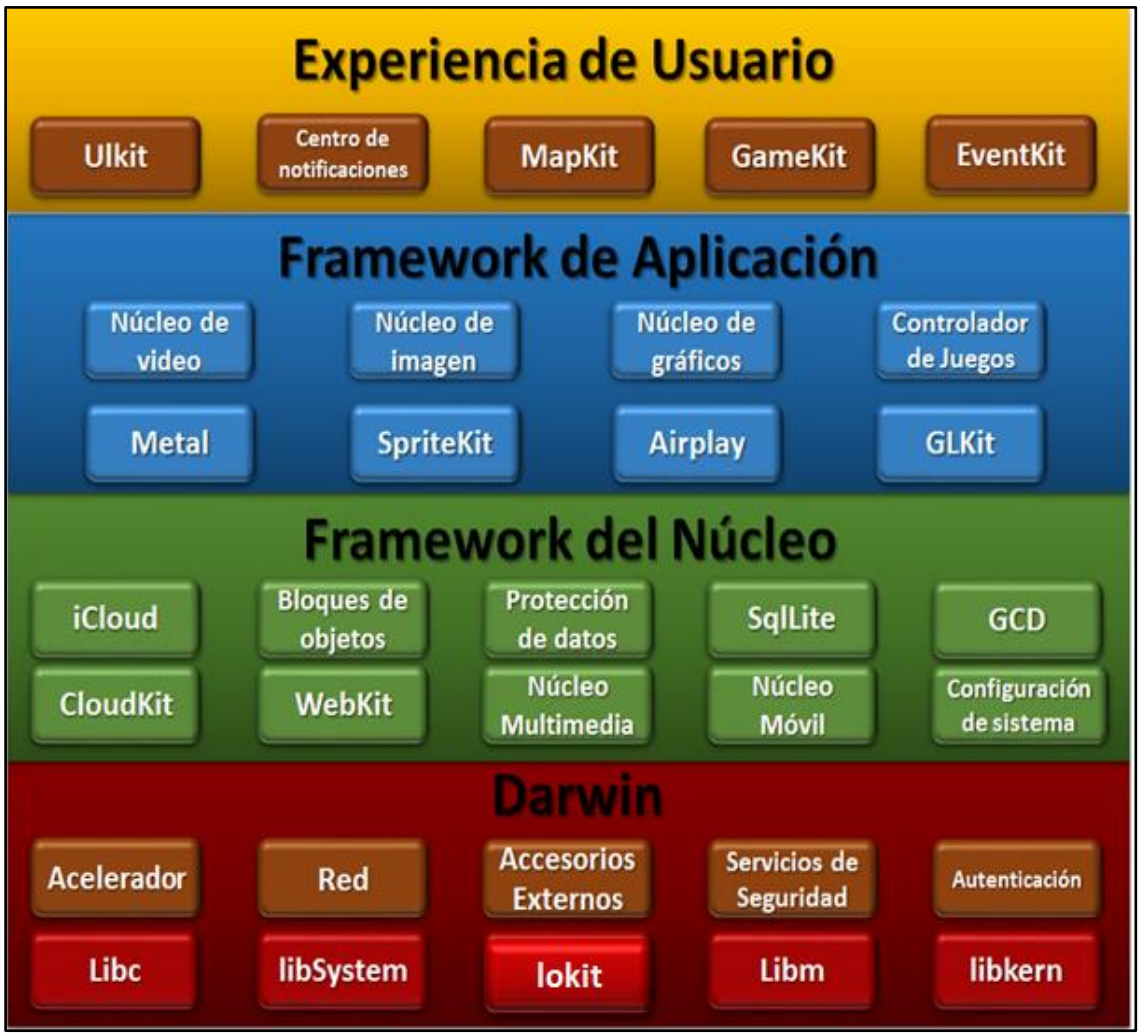


Figura 7. Arquitectura de iOS

Fuente: Apple Developer [11]. Traducida

1. **Darwin:** También conocida como capa de núcleo del sistema; es el kernel derivado de los BSD (UNIX). Esta capa contiene todos los códigos en bajo nivel de los frameworks de capa superior; además, en esta capa se encuentran los drivers de todo el hardware del dispositivo. En Darwin se encuentran dos grandes grupos de código: las librerías del núcleo (figura 9 en rojo) y las librerías o frameworks de servicios (figura 9 en naranja).

   Las librerías principales del núcleo son:

   a. **Libc**: Cumple la misma función que en Android, proveer todas las cabeceras y funciones estándar del lenguaje C, para dar uniformidad al código.

   b. **libSystem:** Es una librería que provee un acceso de bajo nivel a todas las interfaces y servicios superiores de iOS. Estos servicios incluyen el manejo de recursos y su concurrencia (POSIX), el manejo de sockets (servicios de red), el acceso a archivos de sistema, el uso de salida y entrada estándar (Hardware), uso de la memoria, entre otras.

   c. **IoKit:** Es una librería programada en C++, que autoriza y maneja recursos que son utilizados por los frameworks, librerías y otras herramientas presentes en iOS. Una de ellas es la configuración automática del hardware (por ejemplo cuando conectas el dispositivo con un cable USB). Además permite crear nuevos drivers para el manejo del hardware, característica que facilita la programación de Apps.

   d. **Libm:** Es una librería que maneja y procesa operaciones matemáticas básicas que son utilizadas por el compilador C y el CPU. La finalidad de agrupar éstas en una librería, es crear rutinas y optimizarlas de acuerdo a la arquitectura del dispositivo.

   e. **Libkern:** Esta librería se encarga de configurar el uso de las instrucciones que se ejecutan en el procesador, además se encarga de proveer las clases bases que son usadas en C++. En esta clase se guardan todas las configuraciones que son creadas y utilizadas por IoKit.

      Las librerías principales de servicios son:

   a. **Red (Network Extension Framework):** Se encarga de las configuraciones de red como son las VPN. Específicamente se encarga de este servicio, porque es la forma en que establece comunicación con otros dispositivos. Esta librería se encarga de su configuración y administración.

b. **Acelerador (Accelerate Framework):** Este framework se encarga del desempeño que existe en el manejo de señales digitales, álgebra lineal y todo cálculo relacionado con el procesamiento de imágenes, la ventaja de utilizar este framework, es que en cuanto se actualiza el mismo, todas las Apps que lo utilizan son actualizadas también al utilizar funciones más eficientes.

c. **Accesorios Externos (External Accesory Framework:):** Su función es darle soporte a la comunicación de los accesorios que son conectados con cable USB (ambas interfaces) o si es conectado con Bluetooth. Aquí se encuentra toda la información técnica para iniciar sesiones con los dispositivos externos.

d. **Servicios de Seguridad (Security Framework, Generic Security Services Framework, Local Authentication Framework):** Se trata de un conjunto de frameworks que tiene la finalidad de proveer seguridad al dispositivo. En caso del framework de servicios genéricos de seguridad, se apoya en los RFC 2743 y 4401 en lo que respecta a la autenticación y el uso de protocolos al conectarse con servidores en internet permitidos en el desarrollo de aplicaciones; además provee plantillas para ser usadas por los desarrolladores. El framework de seguridad está encargado de la protección de los datos mediante el cifrado, el uso de certificados, llaves públicas y privadas, políticas de uso y otras medidas que deberán ser utilizadas en las Apps. El framework de autenticación local se encarga de mecanismos de seguridad como son el bloqueo de pantalla, la huella dactilar y otras medidas para el uso común del equipo.

2. **Framework del Núcleo:** También conocido como Capa de Servicios del Núcleo, en esta capa se encuentra los servicios fundamentales del sistema que son utilizados por las Apps; todos son administrados con frameworks y tecnologías individuales con la finalidad de administrar de manera correcta estos recursos. Al igual que en la capa anterior, se encuentran frameworks de alto nivel y de bajo nivel.

   Los principales frameworks son:

   a. **iCloud:** Abarca toda función o información relacionada con los servidores centralizados para los dispositivos Apple, en este caso iCloud. Maneja toda la información del usuario como son compras, datos personales, protección y uso de dispositivos Apple, sincronización de información, etc. La finalidad es proveer una medida que optimice el uso de la información y su vigencia. Existen varias funciones al respecto, una de ellas es Cloud Kit Storage (almacenamiento), que se encarga de administrar el contenido que uno desea hacer público o compartirlo de manera pública.

b. **Bloques de Objetos (Block Objects):** Son funciones anónimas que pueden ser utilizadas tanto en C como en Objective C que permite crear versatilidad en lo que respecta al uso de información sin importar en qué lenguaje se esté programando, esto incluye el uso de métodos, funciones con retorno, tareas asíncronas, etc.

c. **Protección de Datos (Data Protection):** Es un conjunto de funciones que permiten a las Apps utilizar la información del usuario como sensible, lo que permite utilizar servicios de Darwin como es el cifrado, sin embargo, esto puede ser afectado con el Jailbreak.

d. **Gran Centro de Envíos (Grand Central Dispatch):** GCD en inglés, es una tecnología propia de los BSD en la que se administra la ejecución de tareas en la App. Posee tanto un modelo asíncrono de programación y un modelo optimizado de ejecución de tareas. También maneja el uso de descriptores de atributos de archivos, el uso de contadores, manejo de señales y eventos.

e. **SQLite:** Es una librera de tamaño pequeño para la administración de bases de datos mediante SQL, permite conectarse a una base de datos remota que pertenezca a un servidor o crear dentro del mismo dispositivo archivos y tablas utilizadas por la App, la librería es de propósito general, es decir; puede ser utilizado por cualquier App.

f. **Núcleo Multimedia (Core Media Framework):** Es una librería de bajo nivel que permite el control preciso de datos y procesos en el desarrollo de contenido de audio y video.

g. **Núcleo Móvil (Mobile Core Services Framework):** Provee la interfaces para crear identificadores del tipo uniforme de bajo nivel que son utilizados por otras Apps.

h. **Configuración del Sistema (System Configuration Framework):** Este framework administra y configura la información sobre el uso y estado de conexión de la banda celular y otras tecnologías como el Wi-Fi.

i. **WebKit (WebKit Framework):** Es un administrador y manejador de contenido HTML presente en servicios web, este framework permite extraer códigos fuentes, enlaces a otras páginas presentes en las páginas.

3. **Framework de Aplicación:** También conocida como la Capa de Medios contiene todos los frameworks para la creación de imágenes, audio y video en el desarrollo de

Apps. Cada sección multimedia contiene varios frameworks relacionados, lo que permite tener una variedad de opciones al desarrollar una App.

Algunos frameworks de esta capa son:

a. **Núcleo de Gráficos (Core Graphics Framework):** Conocido con el nombre de Quartz, es un motor de gráficos en 2D mediante renderizado, este framework es utilizado como base para todas las Apps desarrolladas; es menos veloz que OpenGL.

b. **Núcleo de Imagen (Core Image):** Es un framework utilizado en la administración de video e imagen mediante filtros avanzados, forma parte de muchas Apps que utilizan funciones como la detección de código QR, efectos en fotografías, etc. Trabaja en un ambiente no destructivo, es decir; que crea una copia de la original y trabaja con la copia para no afectar el archivo original.

c. **Núcleo de Video (Core Video):** Proporciona buffers adicionales en el procesamiento de imágenes y video.

d. **Metal:** Es un framework que permite el uso del GPU del dispositivo a bajo nivel, permite a desarrolladores avanzados realizar Apps con el máximo uso de los recursos del dispositivo.

e. **GLKit y OpenGL:** OpenGL es un framework que permite el procesamiento avanzado de gráficos en 2D y 3D, proporciona un control completo durante el mismo y es el framework más utilizado en este apartado; este trabaja en C. En caso de GLKit es un conjunto de clases en Objective C para el uso de OpenGL en un entorno de programación orientado a objetos.

f. **Núcleo de Audio (Core Audio):** Es una familia de frameworks que administra todo el procesamiento de audio, esto incluye: grabaciones, mezclas, filtros, reproducción, borrado, etc. Los frameworks que lo componen son: Core Audio, Audio Toolbox, Audio Unit, Core MIDI, Media Toolbox.

g. **OpenAL:** Es un framework de la familia de OpenGL enfocada en el procesamiento avanzado de audio en el desarrollo de una App.

4. **Experiencia de Usuario:** También conocida como Capa de Cocoa Touch es la capa donde se encuentra la interfaz de usuario y todo lo que su administración conlleva. En esta capa se encuentran los frameworks básicos en el desarrollo de Apps, cuando

un desarrollador comienza a crear una App en iOS lo recomendable es que utilice los frameworks de esta capa.

Algunos frameworks de esta capa son:

a. **Ulkit Framework:** Provee la infraestructura principal en el desarrollo de una App, entre las acciones que controla son: el arranque de la App, control de la interfaz de usuario, el control de objetos, administración de la pantalla touch, soporte multitarea, etc.

b. **MapKit Framework:** Este framework se encarga de la administración de mapas mediante GPS o cualquier sistema de posicionamiento y su uso en una App.

c. **EventKit Framework:** Este framework se encarga de la administración del uso del calendario y agendas de tareas tanto propio de iOS como el usado en una App.

d. **GameKit Framework:** Implementa y administra el soporte que tiene el dispositivo con el servidor de Game Center, el servidor principal de videojuegos de Apple, esto incluye la administración de cuentas, torneos, amigos, etc.

e. **Centro de Notificaciones (Notification Center Framework):** Administra todos los mensajes emergentes que son utilizados por las Apps y que aparecen en la pantalla principal del dispositivo.

Seguridad

A diferencia de Android, cuya seguridad ha crecido gradualmente, iOS desde su diseño, se ha procurado que tenga medidas de seguridad en toda su infraestructura y en la mayor cantidad de procesos que realizan estos dispositivos. Apple también ha trabajado en actualizar y optimizar dichas medidas conforme pasa el tiempo. En esta sección se han visto varias de estas medidas presentes en tecnologías, procedimientos y estándares. A continuación veremos la recopilación en función de su capa de abstracción.

1. **Seguridad en Darwin (Capa 1):** En esta capa se encuentran todos los frameworks principales que proveen seguridad, entre ellos está el de Red que provee la administración de VPNs y la forma en la que establece conexiones seguras con otros dispositivos; los servicios genéricos de seguridad controlan protocolos de seguridad al navegar por internet y la parte de autenticación con otros servidores;

adicionalmente existe otro framework de autenticación local que controla el acceso al dispositivo: la pantalla de bloqueo, la activación y toda contraseña que pueda implementar el usuario.

En cuestión del kernel, el sistema de archivos iOS maneja HFSX, una versión de administración de archivos sensible, que no permite más de un archivo con el mismo nombre, el mismo sistema operativo controla la compresión de archivos. La principal medida seguridad que posee en el núcleo es la caja de arena de la cual se hablará en breve.

2. **Seguridad en Frameworks del Núcleo (Capa 2):** En esta capa se limita el uso que tienen todos los desarrolladores al uso de frameworks para tener acceso al hardware y carpetas del sistema, esto impide que un desarrollador pueda crear procesos maliciosos en segundo plano o el mal uso de archivos personales, porque todos estos se encuentran cifrados, como sucede en los bloques de objetos o el uso que le dan al conectarse con el servidor de iCloud.

3. **Seguridad en Frameworks de Aplicación (Capa 3):** De manera similar que en capa 2 los frameworks de aplicación limitan el uso de funciones o procesos maliciosos cuando se programa en alto nivel el desarrollo de una App en lo que respecta el uso o administración del hardware.

4. **Seguridad en Experiencia de Usuario (Capa 4):** Cualquier función o proceso que interactúe directamente con el usuario está controlado por el framework que haya sido utilizado para programar la App, eso permite que los datos sensibles del usuario sean cifrados, adicionalmente; antes de colocar una App en la Appstore para disposición general, es sometida, bajo un riguroso proceso de validación por parte de Apple (no presente en Google), en caso de no ser aprobada la App no es publicada.

La Caja de Arena

Esta caja de arena es completamente diferente a la que se encuentra presente en Android, ésta trabaja como agente principal para establecer un principio de seguridad llamado “compartimentación” (compartmentalization), muy parecido al principio de seguridad por oscuridad, ambos trabajan en ocultar toda información importante y accesos a agentes (usuarios y procesos) que no requieren tener acceso a tal información o privilegios. Para ello, crea entornos seguros en los cuales los procesos de cualquier App son ejecutados en dicha caja de arena, adicionalmente de la existencia de frameworks que regulan el código utilizado en la misma.

Por estas funciones que realiza la caja de arena, coloquialmente es conocida como "Jail" (prisión) debido a las restricciones que esta coloca sobre los usuarios y desarrolladores, entre las principales funciones que realiza la caja de arena en iOS son:

- Restringir el acceso al sistema de archivos y privilegios de root a cualquier App instalada en el dispositivo. Todas las Apps se encuentran instaladas en una misma carpeta y solo tienen acceso al contenido de ella (/var/mobile/Applications/NombreApp).

- Restringir el acceso de cualquier App a los recursos del sistema u otra App. Esto se consigue mediante el identificador de procesos (UID).

- Restringir el uso del hardware, sino es mediante el uso de algún framework que provee Apple. A pesar que pudiera programarse sin la necesidad de ellos, la restricción aplica cuando se encuentra en revisión por parte de la Appstore, está firmada digitalmente la App para su uso y si no tiene la firma no es utilizada.

- Restringir la generación de código dinámico por cualquier otro proceso.

- Restringir el incremento de recursos o privilegios una vez que el proceso esté en ejecución.

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
      <key>com.apple.springboard.debugapplications</key>
      <true/>
      <key>get-task-allow</key>
      <true/>
      <key>task_for_pid-allow</key>
      <true/>
      <key>run-unsigned-code</key>
      <true/>
</dict>
</plist>
```

Figura 8. Directiva utilizada por la caja de arena de iOS

Fuente: Mac OS and iOS Internals [8].

En la Figura 8 se observa un documento en formato XML sobre una directiva utilizada por un servidor de corrección de errores (Debug Server) en el documento, en la sección <dict>, podemos observar las acciones que tuviera o no permitidas realizar; para que la caja de arena pueda trabajar con esta directiva es necesario que Apple la firme digitalmente, de esta manera se encuentra aprobada, en caso contrario; no procede.

La caja de arena corre como un proceso demonio en todo momento en el dispositivo (/usr/libexec/sandboxd), adicionalmente; tiene un verificador de integridad, tanto de los archivos como de las directivas utilizadas al momento de ejecutar algún proceso.

La desventaja que se otorgue todo el control a un solo proceso centralizado, es el hecho de que una vez que pueda ser vulnerado, este otorga el control completo del dispositivo como lo veremos en la práctica del Jailbreak.

Auditorías de Seguridad

Con el avance de la tecnología en los últimos años, se ha conseguido un considerable y benéfico avance dentro las actividades humanas, el hecho de que casi todas éstas se realicen empleando el uso de las redes y el internet permite que puedan realizarse desde cualquier computadora, en cualquier lugar y casi en cualquier tiempo; realizándose de una manera más rápida y eficiente que como anteriormente se hacía.

Todas estas ventajas han servido para mejorar la calidad de vida de la gente, sin embargo; no han sido ventajas del todo, porque adquirir ventajas como: la portabilidad, la disponibilidad, la compatibilidad han creado en contra parte; un aumento en el tiempo, esfuerzo y dinero de otras tareas para conservar las ventajas, a saber; la administración de los sistemas en los que van implicadas dichas actividades.

Dentro de la administración de una red, se ven envueltas tareas que pueden ser clasificadas en dos vertientes: los mantenimientos preventivo y correctivo; esto aplica a las estructuras físicas y lógicas de todo el sistema con el que funciona una organización. Los recursos que se requieren invertir para el buen funcionamiento de los sistemas, dependerá de varios factores como suelen ser: el tamaño, el tiempo disponible, la complejidad del sistema, el personal envuelto, entre otras cosas. Como se tratan de tareas complejas y a gran escala, se requiere tener una metodología bien estructurada y definida que pueda garantizar el buen funcionamiento de las mismas una vez realizada; el uso y apego a los estándares relacionados garantizan que las tareas se están realizando de la mejor manera posible.

El uso de estándares permite que las organizaciones adquieran certificaciones o reconocimientos de calidad, tenerlas representa en una ventaja, por ejemplo; dan confianza a las inversiones y a los inversionistas, facilitan alianzas o contrataciones con otras organizaciones, pero sobre todo, protegen sus propios activos de la manera más eficiente posible.

Tomando todo lo anterior en cuenta, nos enfocaremos en un estándar reciente en cuestión de las auditorias de seguridad, El Penetration Testing Execution Standard (Estándar

de Ejecución de Pruebas de Penetración), no es el único que existe, sin embargo; se escogió este por el grado de practicidad que posee en la realización de las tareas necesarias para cada una de sus fases.

Penetration Testing Execution Standard

Es un standard que tiene como fin, definir una guía confiable del proceso a seguir para realizar una auditoría de seguridad (pentest) de manera efectiva. Establece diversas directrices con el objetivo de identificar las necesidades y focos de atención durante el proceso mismo, además proporciona sugerencias y ejemplos que permiten su correcta comprensión. También proporciona un catálogo bastante amplio de herramientas, tanto gratuitas como de paga que podrían ser usadas durante alguna de sus fases.

Esta metodología se divide en 7 partes fundamentales:

- Negociación Previa (Pre-engagement Interactions)
- Recopilación de la Información (Intelligence Gathering)
- Modelado de Amenazas (Threat Modeling)
- Análisis de Vulnerabilidades (Vulnerability Analysis)
- Explotación (Exploitation)
- Post- Explotación (Post Exploitation)
- Reportes (Reporting)

La forma de abordar esta metodología será mediante una referencia general de las actividades a realizar y los implicados, estos serán identificados como el “Cliente” que es la persona u organización que solicita los servicios de la auditoria y el “Prestador de Servicios” que es la persona o grupo de personas que se encargaran de ejecutar la misma.

Negociación Previa

En esta fase se definen todas las actividades y el alcance que tendrán durante el proceso de Pentesting, la profundidad, los ámbitos, el enfoque y las metas que dicho proceso tendrán, se hace mediante un documento formal que se firma entre el prestador de servicios como el cliente que lo solicita. La intención es evitar problemas de comunicación y tener un plan bien trazado de lo que se quiere llegar con dicha prueba, para enfocarse en las necesidades que el cliente solicita, sin desviarse en ningún momento de dicho objetivo.

Es necesario que durante esta fase se determinen todas las métricas necesarias, éstas abarcan aspectos como el tiempo que se le dedicará al proyecto, los medios de comunicación entre los implicados, definir a los responsables, formas de pago, consideración de los aspectos legales, la forma en que afectará a la organización y la metodología a seguir. De acuerdo al estándar, el tiempo invertido en esta fase debe ser de al menos el 20% del

total que se planea invertir durante todo el proyecto. Adicionalmente, se recomienda que exista un 20% del tiempo como reserva por cualquier atraso o inconveniente que exista; lo que deja el 60% para la realización de las 6 fases restantes.

Las acciones esenciales durante esta fase se resumen en la Tabla 3:

Tabla 3. Resumen de actividades durante la fase de Negociación Previa

Acción	Autor	Descripción
Reunión previa de negociación	Cliente y Prestador del Servicio	Tiene como fin el determinar entre el cliente y el prestador del servicio, los siguientes aspectos que definirán el proceso que se llevará durante la prueba: 1.Alcances -Duración del proyecto (Fecha de Inicio y Término, Extra) -Actividades a realizar y la profundidad de las mismas -Actividades no previstas -Activos del cliente implicados 2.Metas 3.Contacto entre el cliente y el prestador de servicios 4.Reglas de negocio 5.Términos de pago La elaboración de cuestionarios específicos puede ser de utilidad en alcanzar y fijar las metas que el cliente requiere. Responsabilidad del Prestador del Servicio: -Solicitar toda la información ya sea con entrevistas con el cliente, información escrita, visita presencial a las instalaciones del cliente, etc. Tomando como referencia los puntos anteriores. -Llevar un informe detallado de la información que se obtenga con el cliente. Responsabilidad del Cliente: -Proporcionar la información solicitada por el prestador de servicios.

		-Proporcionar información que considere necesaria que pueda ayudar a la mejor realización de la prueba. En el anexo 1A se detalla más información al respecto.
Negociación con terceros	Prestador del Servicio ,Cliente ,Terceros	Si el cliente disfruta de servicios provisto por terceros, el prestador del servicio debe acordar con los mismos, los términos y condiciones en las que el prestador podrá realizar la prueba de pentest a los recursos y activos de los terceros que tengan relación con el cliente, de ser necesario; puede solicitarle el cliente información necesaria para llegar a dichos acuerdos. Entre los principales terceros se encuentran servicios ISP, Web Hosting, Cloud, MSSP, Leyes internacionales o extranjeras que pudieran verse implicadas, etc.
Elaboración del contrato escrito	Prestador del Servicio	Elaboración de un documento formal donde quede por escrito todo lo acordado en las reuniones previas, tanto el cliente con el prestador del servicio, como el prestador del servicio y los terceros con los que tiene relación el cliente. Este documento servirá como guía tanto el cliente como al prestador de servicios.
Lectura y firma del contrato	Cliente	El cliente lee el contrato y al firmarlo aprueba la forma en que se llevará a cabo la prueba. De no estar acuerdo, debe corregir los puntos que no apruebe con el prestador del servicio, hasta quedar conforme.
Preparación previa a la prueba	Prestador del Servicio	El prestador de servicio con la información ya obtenida y su experiencia, comenzará la preparación necesaria tanto de su equipo, como de las herramientas necesarias que usará durante la prueba.

Recopilación de la Información

Esta fase consiste en la recolección de información importante que será útil durante las fases de planeación, análisis y explotación; en esta fase el prestador del servicio debe formarse una idea clara sobre la organización y su forma de operar de la misma; con el fin de que

pueda realizar una planeación exitosa durante la siguiente fase y puedan alcanzarse las metas que se fijaron en el contrato.

Este proceso incluye identificar a todos los activos implicados, sus características, sus escenarios en los que se desenvuelven, su trabajo dentro de la organización, sus puntos fuertes y débiles, en esencia; cualquier dato que pueda ser útil para el prestador de servicio, para que las pruebas que se realizaran en la organización, sean lo más completas y útiles, para alcanzar las metas fijadas.

Las acciones esenciales durante esta fase se resumen en la Tabla 4:

Tabla 4. Resumen de actividades durante la fase de Recopilación de la Información

Acción	Autor	Descripción
Selección de objetivos	Prestador del Servicio	Se deben identificar y nombrar todos los objetivos que estarán bajo la prueba, se debe tomar en cuenta lo siguiente al momento de identificarlos: • Recordar las reglas de negocio y los límites de la prueba. • Considerar el tiempo de la prueba. • Considerar la meta final de la prueba. • Establecer que se espera obtener durante la fase de recopilación. • Planear como se espera conseguirlo.
Búsqueda de información en vías públicas	Prestador del Servicio	El prestador de servicio busca información pública de los objetivos, es decir; información que no requiere de comprometer a un sistema, se recomienda usar la técnica OSINT. Este establece el siguiente diagrama: +Corporativo -Física -Lógica -Financiera -Electrónica -Organización y Relaciones -Infraestructura +Individual -Historia -Redes sociales -Servicios de red que utiliza -Localización física

		-Datos de contacto -Uso de dispositivos móviles En el Anexo 1B se detalla más información sobre OSINT.
Búsqueda de información de manera encubierta	Prestador del Servicio	Obtener información del cliente con una interacción directa de manera encubierta, en caso de nivel personal; puede utilizar el HUMINT si es aplicable u otro método de ingeniería social.
Footprinting externo	Prestador del Servicio	Obtener información útil mediante técnicas que tengan interacción con el equipo. Está parte se enfoca en obtener información fuera de la red interna; se debe determinar los equipos, servidores, número de sistemas autónomos, escaneo de puertos, captura de banners, barridos SNMP, zonas de transferencia, rebote de SMTP, descubrimiento de DNS, fuerza bruta a DNS y ataques en reversa, escaneo de aplicaciones web, identificar servicios virtualizados, infraestructura, presencia de dispositivos móviles, etc.
Análisis de resultados	Prestador del Servicio	Realizar un análisis de la información recopilada para elaborar una guía, con toda la información útil para el desarrollo de las fases posteriores.
Footprinting interno	Prestador del Servicio	Obtener información útil del cliente desde la red interna, solo si tiene acceso a ella, esto comprende las mismas actividades que el Footprinting externo, más todos los datos que solamente se puede obtener dentro de la red interna; como es el localizar servidores DHCP, rangos de IP, dispositivos móviles, infraestructura de VoIP, servicios de intranet, directorios, distribución de los diferentes departamentos, etc.
Análisis de resultados	Prestador del Servicio	Realizar un análisis de la información recopilada para elaborar una guía, con toda la información útil para el desarrollo de las fases posteriores.
Identificar los mecanismos de seguridad	Prestador del Servicio	Identificar todos los mecanismos de seguridad, físicos, lógicos, políticas, etc. con las que cuenta el cliente. Para poder realizar la fase de modelado de amenazas.

Modelado de Amenazas

Esta fase consiste en la elaboración de un modelo de amenazas que pueden, en determinado momento afectar de manera directa o indirecta a la organización; este modelo, también es útil durante el análisis de las vulnerabilidades y posteriormente durante la fase de explotación. El estándar sugiere enfocarse en el hecho de que las amenazas puedan ser clasificadas de acuerdo a la importancia que tienen y en el hecho de que puedan presentarse en un futuro con los mismos resultados.

Se debe tener especial atención en: los activos de negocios, los procesos de negocios, en las amenazas propiamente y sus capacidades de las mismas, estos 4 aspectos deben ser claramente identificados y documentados.

A grandes rasgos se debe considerar el tener a la mano información relevante, identificar, categorizar los activos primarios y secundarios, identificar y categorizar las amenazas y finalmente, trazar el plan en que las amenazas serán usadas en contra de los activos primarios y secundarios, tomando en consideración el impacto que estas tendrían dentro de la organización.

Las acciones esenciales durante esta fase se resumen en la Tabla 5:

Tabla 5. Resumen de actividades durante la fase de Modelado de Amenazas

Acción	Autor	Descripción
Análisis de los activos de negocio	Prestador del Servicio	Identificar los activos (personal, información, etc.) que formen parte del negocio de la organización y que sean potenciales candidatos de ser los objetivos primarios por los atacantes, estos se relacionan directamente con el valor monetario de la organización. Se pueden clasificar de la siguiente manera: -Datos de la organización: como pueden ser políticas, planes, procedimientos, información de productos, mercadeo, información financiera, etc. -Información Técnica: el diseño de la infraestructura lógica, configuraciones, cuentas de usuario, cuentas privilegiadas, tecnologías, etc. -Empleados: información de empleados que puedan afectar directamente a la organización, en cualquier ámbito ya sea físico o lógico. O que conozca información sensible sobre el

		funcionamiento interno o externo de la organización. -Clientes: información de los clientes que puedan afectar las relaciones con la organización, o que puedan materializar una situación de fraude.
Análisis de los procesos de negocio	Prestador del Servicio	Esta parte consiste en identificar los procesos críticos y los no críticos que forma parte de las actividades primarias de la organización, creando escenarios en donde se identifique como las amenazas pueden dañar o repercutir de alguna manera a la organización en su estado actual. Puede organizarse de la siguiente manera: -Infraestructura tecnológica: Consiste en identificar todas las computadoras, redes, herramientas, etc. que sean usados durante los procesos de negocio, esto incluye los dispositivos móviles. -Información de soporte: Información que sirva de referencia o soporte en aspectos críticos de la empresa como puede ser toma de decisiones, legal, mercadeo, etc. -Personal: Todos las personas que forman parte de un proceso de negocio, suelen ser personas que aprueban las acciones, verifican, o cumplen con alguna normativa pueden ser documentadas dentro de esta parte. -Terceros: Cualquier tercero que tenga parte en un proceso de negocio, debe ser tomado en cuenta.
Análisis de amenazas	Prestador del Servicio	Se definen todas las amenazas, individuales o por grupo, que pueden ser clasificadas en términos de la relación que tienen con la organización (interna o externa).
Análisis de las capacidades de las amenazas	Prestador del Servicio	En este caso se hace un análisis técnico de las posibilidades que tienen las amenazas de comprometer los sistemas tomando, en cuenta sus recursos, comunicaciones, privilegios, incluso, las posibles motivaciones que pudieran tener, para crear un daño a la empresa.
Análisis de casos similares	Prestador del Servicio	Realizar una investigación sobre incidentes o noticias relacionadas de otras organizaciones que se desenvuelven en la misma industria, para crear una comparativa entre dicha organización y el cliente, con el fin de obtener

		información para las siguientes fases ya sea en el ámbito preventivo o correctivo.

Análisis de Vulnerabilidades

El análisis de vulnerabilidades es un proceso que tiene como fin, el identificar, validar y documentar todos los fallos, o flaquezas que tiene la organización que está siendo auditada.

Este análisis consta de 3 actividades esenciales, el testeo, la validación y la investigación. El testeo es el proceso de descubrir e identificar los fallos y flaquezas de la organización que pudieran ser usadas por un atacante (amenaza). Pueden ser del tipo tecnológico, físico e incluso humano. Este debe estar completamente apegado a los alcances acordados y a las metas que se pretenden cubrir con las pruebas; posteriormente sigue la validación, es un proceso en el que se agrupa y clasifica toda información obtenida durante el testeo, puede ser mediante IP, ID, MAC por el tipo del vulnerabilidad presente en el activo, etc. La razón va en función de que durante el proceso del testeo, pueden usarse varias herramientas para obtener la misma información de un mismo objetivo y finalmente; realizar una investigación exhaustiva sobre todas aquellas posibles amenazas que pudieran utilizar la vulnerabilidad presente en el cliente.

Las acciones esenciales durante esta fase se resumen en la Tabla 6:

Tabla 6. Resumen de actividades durante la fase de Análisis de Vulnerabilidades

Acción	Autor	Descripción
Testeo activo	Prestador del Servicio	Este tipo de evaluación, envuelve directamente la interacción con el activo que estará bajo prueba, pueden usarse herramientas automatizadas que ayuden al prestador del servicio a encontrar información valiosa, entre las herramientas principales se encuentran los escáneres basados en puertos, en servicios o la captura de banners de servicio. También incluye el uso de escáneres de aplicaciones Web y escáneres de vulnerabilidad de red. Estos pueden ser generales o enfocados directamente a una tecnología o ejecutados de manera personal por el prestador de servicio.
Testeo pasivo	Prestador del Servicio	Consiste en verificar los metadatos de los archivos con el fin de encontrar información valiosa como suelen ser direcciones internas de los servidores, IP´s y otra información que le permita al prestador del servicio obtener acceso. El análisis de tráfico también se incluye dentro del testeo pasivo, con el fin de obtener

		configuraciones de los dispositivos de interconexión de red como suelen ser switches y routers, a excepción del envenenamiento del router, no entra en esta categoría.
Validación	Prestador del Servicio	Esta fase consiste en categorizar todas las vulnerabilidades encontradas, se sugieren de dos maneras, en grupos o específicas, estas obtenidas con las herramientas que realizan dichos análisis y tomar en cuenta la forma en que las vulnerabilidades son categorizadas (ID, CVE, OSVDB). También incluye las pruebas manuales a diversas tecnologías como suelen ser VPN´s, Servidores Citrix, DNS, servicios web, servidores de correo electrónico, etc. Otra actividad importante es el desarrollo de vectores de ataque, lo que será de gran ayuda durante las fases de explotación y reportes. El hacer pruebas de las vulnerabilidades antes de explotarlas y confirmar manualmente las más importantes, forma parte de las buenas prácticas.
Investigación	Prestador del Servicio	Consiste en realizar investigaciones de diferentes fuentes para confirmar la información obtenida por las herramientas usadas para el análisis de las vulnerabilidades. En esta parte se encuentra la investigación pública en la que se puede considerar, bases de datos de vulnerabilidades como la CVE, los proveedores del software. Otros campos en los que se puede buscar son las bases de datos de exploits, contraseñas por default o débiles por parte de los programas, guías de hardening o configuraciones erróneas más comunes. Otro tipo de investigación es la privada, que consiste en realizar análisis exhaustivos de características particulares de las organización qué estará bajo prueba, esto consiste básicamente en 3 acciones: 1. Replicar mediante máquinas virtuales, los escenarios que estarán bajo prueba. 2. Evaluar las configuraciones en los entornos virtuales. 3. El Fuzzing a aplicaciones de la empresa. También debe ser considerado el potencial que tienen los ataques y el análisis de código (Ingeniería en reversa).

Explotación

La explotación es la fase en la que se establecen accesos con el sistema o con los recursos de la organización pasando por las restricciones de seguridad que esta pudiera tener, el éxito de esta fase depende íntegramente del análisis que se hizo sobre las amenazas y vulnerabilidades en fases anteriores.

Para esta fase debe tomarse en cuenta las medidas de seguridad que tiene la organización, cómo evadirlas, el diseño de ataques de explotación precisos, técnicas que ayuden a anular las medidas de seguridad y tratar de comprometer los activos de la organización desde todas las ópticas obtenidas en las fases previas.

Las acciones esenciales durante esta fase se resumen en la Tabla 7:

Tabla 7. Resumen de actividades durante la fase de Explotación

Acción	Autor	Descripción
Identificación de contra medidas	Prestador del Servicio	Esta acción corresponde a identificar todas las medidas de seguridad de la organización como son los antivirus, técnicas de ofuscación (codificar, empaquetar, cifrar), el factor humano, prevención de ejecución de procesos (DEP), firewalls y otras medidas.
Creación de técnicas de evasión	Prestador del Servicio	Aplicar las técnicas para evitar las medidas de seguridad, evitar las listas blancas, inyección de procesos, ataques que residen en memoria, técnicas de ofuscación de exploits y en general cualquier otra medida que evite ser detectado por los sistemas de seguridad.
Selección precisa de los objetivos y plan de ataque	Prestador del Servicio	Los ataques que se realizarán deben ser totalmente enfocados a objetivos particulares de tal manera que no alerte el cliente de las acciones realizadas por el prestador del servicio, con el fin de simular de la manera más real posible un ataque por un tercero malintencionado sobre la organización.
Selección de herramientas de ataque	Prestador del Servicio	Consiste en la selección de herramientas y programas que permitan realizar las acciones descritas en el plan de ataque, tanto desde el plano físico como el lógico, esto incluye la selección de los tipos de exploits, analizadores de tráfico, herramientas de ataques de redes inalámbricas, accesos físicos, ataques de ingeniería social, etc.

Ataque y documentación	Prestador del Servicio	Esta acción consiste en realizar los ataques acorde a todo lo realizado previamente, se debe documentar todas las acciones realizadas. El anexo 1C contiene más información al respecto.

Post-Explotación

El propósito de esta fase es determinar el valor del activo comprometido y mantener el control del mismo después de su uso. El valor del activo es determinado por la sensibilidad de la información contenida en este y por la facilidad que este ofrece para comprometer aún más la red u otro medio, en base a la consideración de varios factores que deben ser preferentemente las reglas de negocio establecidas durante la negociación previa. Todo este proceso debe ser documentado, la razón radica en su utilidad durante la fase de reportes.

Las actividades de esta fase, van en función que se tiene un equipo del cliente comprometido y se resumen como se puede apreciar en la Tabla 8:

Tabla 8. Resumen de actividades durante la fase de Post-Explotación

Acción	Autor	Descripción
Análisis de la infraestructura de la red interna	Prestador del Servicio	Consiste en realizar nuevamente todas las acciones de footprinting en la fase de obtención de información, siempre y cuando no se haya tenido acceso previamente. La finalidad es encontrar información valiosa de la red interna del cliente que podría ser usada por un atacante, incluye configuraciones, software instalado, configuración lógica, administración interna, etc. El anexo 1C detalla mejor esta sección.
Explotación desde la red interna	Prestador del Servicio	Una vez obtenida la información de la infraestructura se repite la fase de explotación desde la máquina comprometida, enfocándose en obtener accesos a otros segmentos de la red buscando atacar principalmente los activos de negocio y objetivos primarios de la organización. El anexo 1C detalla mejor esta sección.
Documentación	Prestador del Servicio	Todas las acciones realizadas durante esta fase deben ser bien documentadas, debe incluirse toda la información útil posible, esta se determina en función de las metas que ayuden al cliente a obtener una visión clara de los riesgos que corre su organización.

		El anexo 1C detalla mejor esta sección.
Limpieza	Prestador del Servicio	Una vez terminada todas las acciones de obtención de información y explotación, deben ser removidos todo el software usado, configuraciones, modificaciones, etc. Al estado previo de la fase de Explotación.

Reportes

Esta fase establece los criterios comunes que deben ser considerados para la elaboración de reportes, para que los mismos sean de utilidad para el cliente, la propuesta de soluciones y conclusiones. Debe tener dos enfoques distintos, el enfoque ejecutivo y el enfoque técnico. Esto con el fin de que el cliente pueda obtener una visión completa y concisa de los riesgos que corre su organización, junto con las soluciones que ayudarán a mitigar dicho riesgo.

Las acciones de esta fase se resumen en la Tabla 9.

Tabla 9. Resumen de actividades durante la fase de Reportes

Acción	Autor	Descripción
Reporte ejecutivo	Prestador del Servicio	Elaboración del reporte ejecutivo, enfocándose desde la perspectiva del negocio, debe ser ilustrativo, debe contener información que los altos directivos puedan comprender sin dificultad y que les permita identificar los riesgos financieros que corren por los resultados obtenidos durante las fases anteriores. Debe incluir la solución propuesta para la reducción de riesgos. El anexo 1D detalla mejor esta sección.
Reporte técnico	Prestador del Servicio	Este reporte debe incluir toda la información técnica recopilada durante todas las fases anteriores, esto incluye información como configuraciones, técnicas, acciones realizadas, consideraciones, objetivos envueltos, información obtenida a excepción de contraseñas. El anexo 1D detalla mejor esta sección.
Evaluación del riesgo/ exposición	Prestador del Servicio	La formulación de un cuadro de riesgos que presenta la organización del cliente en base a todo lo realizado durante la prueba de penetración, con el fin de ofrecer una visión general de los riesgos técnicos y financieros vistos por desde la perspectiva del prestador de

		servicios. El anexo 1D detalla mejor esta sección.
Entrega y revisión final	Prestador del Servicio Y Cliente	El prestador de servicios debe entregar y exponer los reportes finales al cliente, si el cliente está de acuerdo con lo presentado, se firma el fin del contrato.

CAPÍTULO II

Laboratorios de Pruebas en Android

Laboratorio No.1
"Virtualización Android"

Objetivo:

Instalar todo el SDK de Android en Windows, actualizar y conocer las principales características de esta herramienta. El SDK de Android constituye la principal herramienta de virtualización que será utilizada para cualquier prueba que se requiera hacer en un proceso de Pentesting.

Nota: En el Anexo 2A se encuentra más información de la virtualización en Android y sus características.

Requisitos:

Para la versión más robusta de Android SDK (Android Studio)
-Microsoft Windows Vista/7/8
-2GB en RAM (Se recomiendan 4GB)
-400MB en espacio del disco duro, esto solo comprende la instalación del IDE, sumando el espacio de todas las máquinas virtuales, se requieren al menos 25GB
-En caso de trabajar desde una máquina virtual de Windows, se debe estar configurado el adaptador de Red como NAT
-Resolución de pantalla mínima de 1280X800
-Java JDK7 o superior (Se recomienda la versión de 32 bits por encima de la de 64bits por ser más estable y presentar menos problemas)

Material:

-SDK de Android
-Conexión de Internet de banda ancha
-Computadora con todos los requisitos mencionados en el apartado anterior

Marco teórico:

El Kit de Desarrollo de Software o SDK en inglés (Software Development Kit) de Android y el Entorno de Desarrollo Integrado o IDE en inglés (Integrated Development Environment) de Android constituyen en las herramientas oficiales de desarrollo para aplicaciones de Android patentada y desarrollada por Google.

Estas herramientas poseen una amplia cantidad de funciones que permiten que solo se requiera conocer una única herramienta para desarrollar Apps en Android; entre las

principales funciones del SDK se encuentra un depurador de código, una biblioteca de funciones, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. El SDK se encuentra para todas las plataformas de sistemas operativos actuales, sin embargo; solo nos centraremos en la plataforma de Windows.

La ventaja más importante del SDK es el permitir emular mediante el simulador cualquier versión de Android, esta característica será utilizada durante el desarrollo del resto de los laboratorios del Pentesting de Android.

Existen dos versiones para poder descargar el SDK:

Android Studio: Es la versión completa del SDK de Android incluye todas las herramientas, incluyendo un IDE especializado, semejante a Eclipse (Studio sustituyó a este último), herramientas para la emulación avanzada y otras que facilitan el desarrollo de aplicaciones, como el monitoreo de CPU y uso de recursos de hardware, interfaz gráfica amigable, administrador de archivos, etc. Esta versión se recomienda para los usuarios que pretendan desarrollar Apps específicamente.

SDK Tools Only: Son todas las herramientas básicas para emular dispositivos Android, además provee las herramientas necesarias para iniciar comunicación con dispositivos reales, la posibilidad de instalar Apps externas de la tienda oficial, borras archivos, entre otras.

La diferencia entre usar una versión y otra radica únicamente, en que Android Studio posee una interfaz de usuario que permite tener al alcance todas las herramientas del que dispone el IDE, situación que en SDK Only se tiene que utilizar por separado y el hecho de que esta versión no posee compilador.

Android Debug Bridge

ADB es la abreviación de Android Debug Bridge o en español Puente de Depuración Android, esta es una aplicación del tipo cliente-servidor en línea de comandos que nos permite comunicarnos con una máquina virtual o con un smartphone real Android. Está formada por 3 componentes:

-Un cliente que corre en el host anfitrión que nos permite iniciar un Shell de comandos, podemos crear varios clientes de manera simultánea.

-Un servidor que corre en un proceso secundario en la máquina anfitrión que es el que administra la conexión entre el cliente y el demonio que corre en la máquina virtual Android.

-Un demonio que corre como un proceso secundario en el emulador o en el smartphone.

La herramienta ADB la podemos encontrar en la ruta <sdk>/platform-tools/adb; el cliente utiliza el puerto TCP 5037 para comunicare con el servidor. Una vez que el servidor inició una sesión, utiliza los puertos TCP 5555-5585 para establecer sesiones con cada instancia con la que se conecta, esta puede ser una MV o un dispositivo físico.

En caso de una MV no requiere ningún requisito o configuración adicional para poder establecer comunicación con el ADB, sin embargo; para un dispositivo real, se requiere adicionalmente:

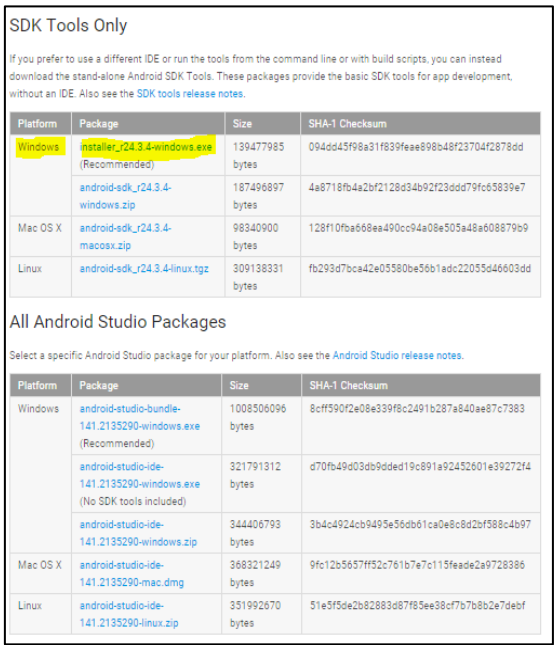
-Depurador USB Activado
-Pantalla desbloqueada
-Drivers USB del smartphone en la PC de trabajo

Procedimiento:

Instalación del SDK

1. Descargar la versión de SDK a Instalar. Esto lo hacemos desde la página oficial de Google: **http://developer.android.com/sdk/index.html.** Al abrir la página, en la parte final de la misma, podemos encontrar información como la mostrada en la Figura 1.

   Nota: Debemos tener en consideración la versión de nuestro sistema operativo y la versión que deseemos descargar.



SDK Tools Only

If you prefer to use a different IDE or run the tools from the command line or with build scripts, you can instead download the stand-alone Android SDK Tools. These packages provide the basic SDK tools for app development, without an IDE. Also see the SDK tools release notes.

Platform	Package	Size	SHA-1 Checksum
Windows	installer_r24.3.4-windows.exe (Recommended)	139477985 bytes	094dd45f98a31f839feae898b48f23704f2878dd
	android-sdk_r24.3.4-windows.zip	187496897 bytes	4a8718fb4e2bf2128d34b92f23ddd79fc65839e7
Mac OS X	android-sdk_r24.3.4-macosx.zip	98340900 bytes	128f10fba668ea490cc94a08e505a48a608879b9
Linux	android-sdk_r24.3.4-linux.tgz	309138331 bytes	fb293d7bca42e05580be56b1adc22055d46603dd

All Android Studio Packages

Select a specific Android Studio package for your platform. Also see the Android Studio release notes.

Platform	Package	Size	SHA-1 Checksum
Windows	android-studio-bundle-141.2135290-windows.exe (Recommended)	1008506096 bytes	8cff590f2e08e339f8c2491b287a840ae87c7383
	android-studio-ide-141.2135290-windows.exe (No SDK tools included)	321791312 bytes	d70fb49d03db9dded19c891a92452601e39272f4
	android-studio-ide-141.2135290-windows.zip	344406793 bytes	3b4c4924cb9495e56db61ca0e8c8d2bf588c4b97
Mac OS X	android-studio-ide-141.2135290-mac.dmg	368321249 bytes	9fc12b5657ff52c761b7e7c115feade2a9728386
Linux	android-studio-ide-141.2135290-linux.zip	351992670 bytes	51e5f5de2b82883d87f85ee38cf7b7b8b2e7debf

Figura 1. Centro de descargas para el SDK de Android
La sección remarcada en amarillo es la seleccionada.

2. Si descargamos el SDK Tools Only tendremos un instalador (installer_rXX.X.X-windows) y lo ejecutamos. En la Figura 2 se muestran las diversas ventanas de información durante la instalación; primero, al ejecutar el instalador arrancará un motor típico de instalación(a). En el segundo paso, el motor de instalación verificará que tenemos instalado Java (JDK) (b), si el motor de instalación lo detecta podremos continuar, en caso contrario; tendremos que verificar que el JDK de Java esté correctamente instalado. Como tercer paso nos pedirá las rutas y nombres de carpetas que la instalación del SDK tendrá; una vez ingresada dicha información, el motor de instalación comenzará a copiar los archivos necesarios(c). Finalmente, una vez instalado; podemos iniciarlo dando click en inicio, posteriormente en "Todos los programas", buscamos la carpeta Android SDK Tools y tendremos dos archivos, AVD y SDK Manager (d).

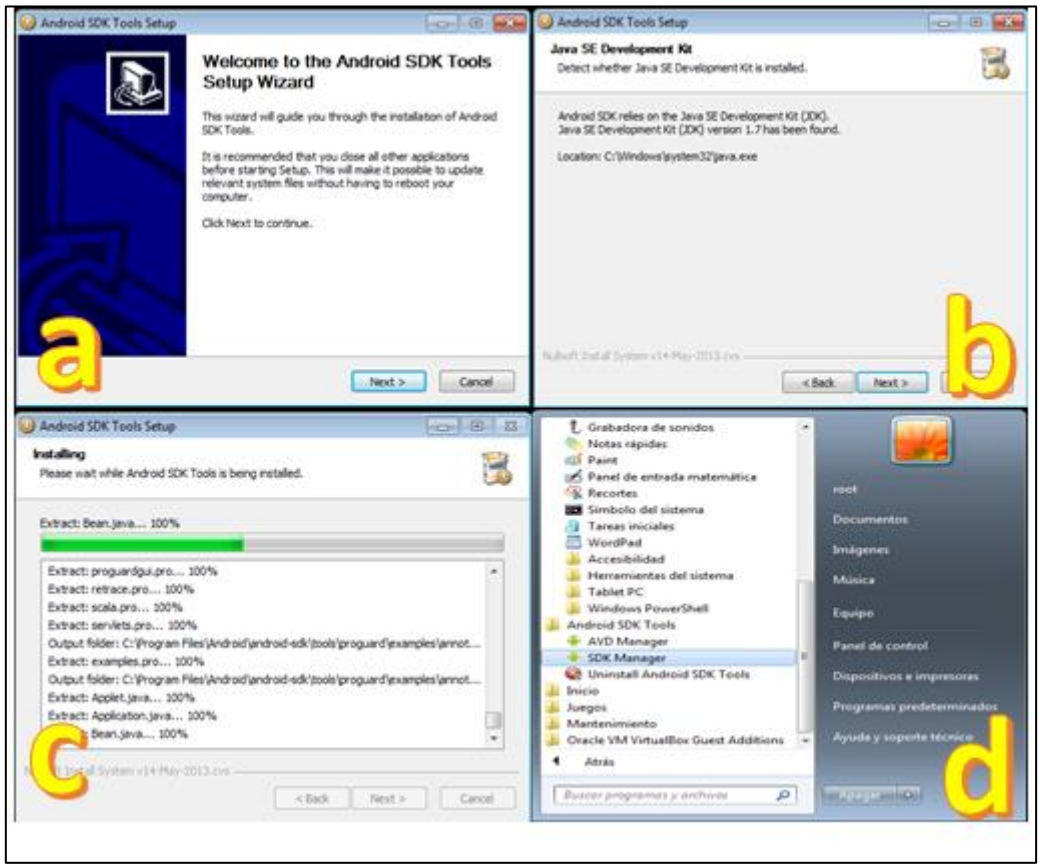


Figura 2. Proceso de instalación del SDK de Android
Versión SDK Tools Only en un SO de Windows 7.

Actualización y uso del SDK Manager

1. Al abrir el SDK Manager nos aparecerá una ventana similar a la mostrada en la Figura 3, la ventana principal corresponde a la ventana de actualizaciones, en esta podemos actualizar cualquier herramienta del SDK, la opción de descargar las APIs de cualquier versión de Android, sin importar si sea nueva o descontinuada; en el apartado Status podemos saber el estado de nuestros archivos, los 3 principales son Installed, Not

Installed y Uptade avaliable, esta última significa que tenemos instalada una versión anterior y que podemos actualizarla a una versión más reciente.

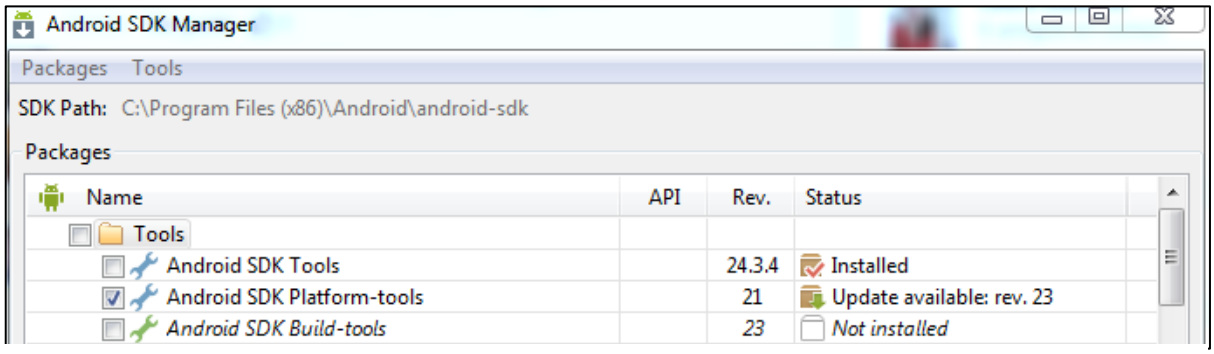


Figura 3. Ventana principal de SDK Manager

Podemos observar el status de la paquetería que estamos utilizando.

Instalar un entorno virtual Android

1. Abrimos el AVD Manager, desde la carpeta Android SDK Tools como muestra la Figura 4a. Otra manera de abrir el AVD Manager es desde la ventana del SDK Manager; seleccionamos el menú Tools y seleccionamos ManageAVDs… (b).

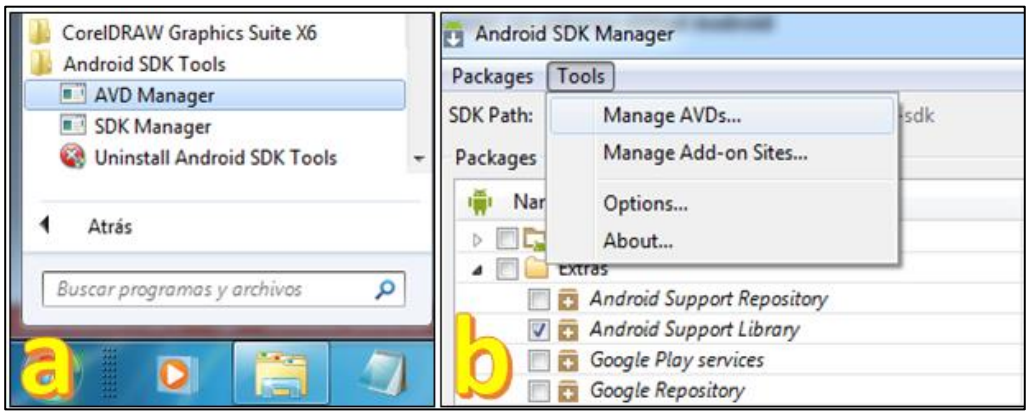


Figura 4. Formas de acceder a AVD Manager

Mediante la carpeta SDK Tools y por SDK Manager.

2. En la ventana de AVD Manager, tenemos dos menús (Figura 5):

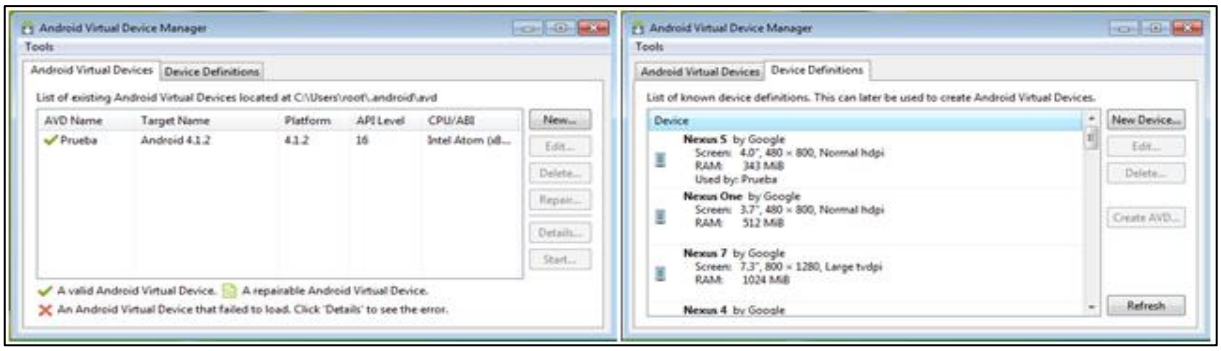


Figura 5. Ventanas de los dos menús de AVD Manager

El primero es el menú de Android Virtual Devices, el segundo corresponde a Device Definitions.

Android Virtual Devices: Nos muestra todas las instancias virtuales que tengamos creadas, muestra el estado de la misma, si se encuentra disponible o si presenta errores, también nos permite desde aquí; crear, editar, borrar, reparar, observar detalles o arrancar nuestras instancias virtuales.

Device Definitions: Son perfiles con características predeterminadas de los Smartphones de Android que circulan en el mercado para poder emularlos como una máquina virtual, al seleccionar un perfil podemos crear nuestra propia instancia virtual sin la necesidad de configurar sus opciones básicas; además nos permite crear perfiles nuevos y utilizarlas como plantillas predeterminadas para un uso posterior.

3. Desde el menú de Android Virtual Devices seleccionamos la opción Create, posteriormente tendremos una ventana con las siguientes opciones(Figura 6):

AVD Name: Nombre de la instancia virtual.

Device: Nos permite seleccionar un perfil predeterminado de configuración. Los perfiles que creamos en Device Definitions aparecerán en este menú.

Target: Versión de Android que tendrá la máquina virtual. En este menú aparecerán todas las APIs que se hayan descargado desde SDK Tools, es necesario tener al menos una descargada.

CPU/ABI: Hardware emulado de la máquina virtual. Es utilizado para opciones avanzadas.

Skin: Opción avanzada de control de hardware.

Front/Back Camera: Se pueden emular las cámaras con cámaras web.

Memory Options: RAM es la memoria de este tipo asignada a la MV, VM Heap es la memoria dinámica asignada al dispositivo.

Internal Storage: Es la memoria interna asignada al Smartphone emulado, toma este espacio del disco duro.

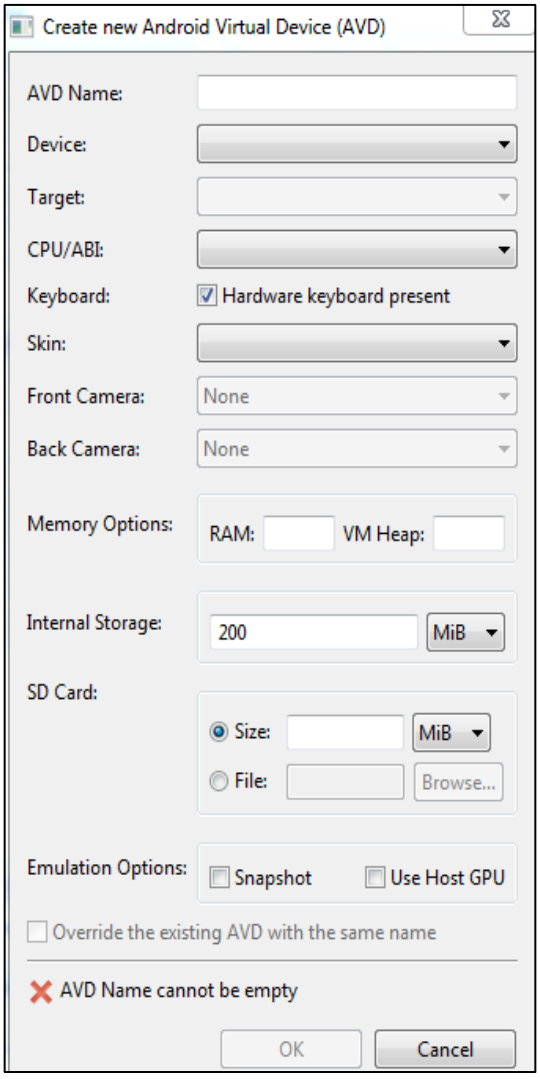


Figura 6. Ventana de configuración de una máquina virtual

SD Card: Emula una memoria de este tipo, toma este espacio del disco duro.

Emulations Options: Son las opciones para emular la MV, se puede usar las capacidades del GPU del host anfitrión, o usarla mediante Snapshots que nos permite ir guardando en todo momento nuestros avances y restaurarlas desde un punto determinado.

4. Una vez configurada y creada nuestra máquina virtual, en la ventana de Android Virtual Devices (Figura 5) la seleccionamos; seleccionamos la opción start; posteriormente en un una nueva ventana aparecerán los datos de configuración de nuestra instancia virtual; si no hay ningún problema, nos aparecerá una nueva ventana con nuestra MV, semejante a la mostrada en la Figura 7. Está ventana es nuestra instancia virtual, de lado izquierdo aparece la interfaz de usuario del dispositivo Android, está cambiará en función de la versión que estemos utilizando; del lado derecho nos aparecerá un menú con los distintos botones físicos que tiene un dispositivo real, como es el botón de encendido o los de control de volumen.

Figura 7. Máquina virtual de Android usando el SDK

Instalar aplicaciones en un entorno virtual mediante ADB Shell

1. Iniciamos una instancia virtual que tengamos creada o puede crearse una nueva, tomando en cuenta las consideraciones del apartado anterior.

2. Descargamos la App (.apk) que deseamos instalar, se recomienda que sea de un sitio confiable; posteriormente guardamos el archivo en la carpeta en donde se encuentra instalado adb tomando como referencia la figura 8.

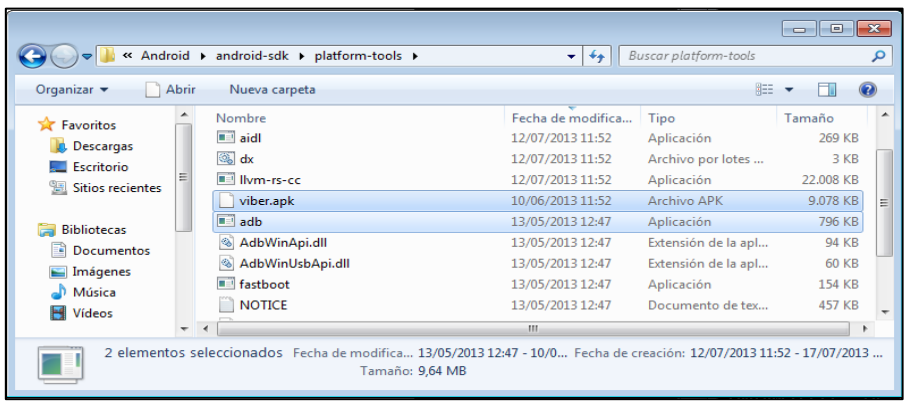


Figura 8. App de Android (.apk) guardada en la misma carpeta de la herramienta adb

3. Iniciamos una terminal de comandos (cmd) y nos posicionamos en la carpeta en donde se encuentra el adb y la App, para instalarla solo tecleamos el comando **adb install <nombre de la aplicación>.** En la Figura 9 podemos observar que después de teclear el comando, es requerida que se inicie una sesión con el dispositivo; en la 4ta línea podemos observar la carpeta del dispositivo en el que está instalado y en la última línea obtenemos el mensaje Success que indica que se ha instalado correctamente.

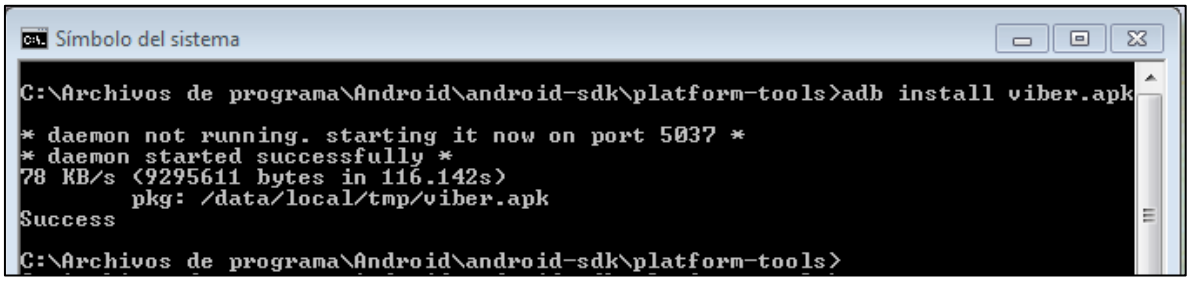


Figura 9. Instalación de una App mediante el comando abd install
La línea de comandos general en Windows 7.

4. Una vez instalada la App, podemos verificarla que esté la nueva aplicación en nuestra MV y que funcione correctamente. Desde la pantalla de aplicaciones, aparecerá el icono típico de la máquina virtual como se puede apreciar en la parte derecha de la imagen 10, al ejecutarla podemos observar; que inicia de forma típica aunque se trate de una máquina virtual. Las Apps que instalamos en máquinas virtuales tienen el mismo funcionamiento, lo que nos permite tener; la experiencia más acercada a un dispositivo real.

Figura 10. Verificación y ejecución de una App instalada mediante abd
La App funciona de la misma forma que en un smartphone real

5. Con el comando **uninstall** podemos desinstalar las aplicaciones que hallamos instalado. Utilizando el mismo método de línea de comandos por adb.

6. El adb también es una herramienta útil en la comunicación directa con una instancia de Android, mediante el comando **adb Shell** iniciamos una sesión la instancia de Android; podemos observar que fue efectiva, porqué el prompt de la línea de comandos cambia por un símbolo # cómo podemos observar en la Figura 11. Esta interfaz de comandos nos permite explorar los directorios de nuestro Android de una manera muy similar a un Shell de Linux con retorno.

Figura 11. Uso del comando adb shell

Los privilegios de este comando dependerán del tipo de usuario que utilicemos.

Resultados:

Esta primera práctica nos dio una introducción base sobre el SDK oficial de Android, permitió conocer las utilidades básicas de este, como crear entornos virtuales y el uso de herramientas como adb. No todas las versiones de Android en máquinas virtuales permiten el uso de todas las configuraciones, por ejemplo el bloqueo de pantalla por contraseña o acceso a las configuraciones.

Los entornos virtuales nos permiten privilegios y acciones que en smartphones reales no se nos permite, por ejemplo el uso del adb Shell; para obtener un acceso total y el uso de todas las herramientas se requiere el rooting previo en el dispositivo real. Esta práctica podría ser útil durante la fase de análisis de vulnerabilidades o modelado de amenazas, porque nos permite realizar pruebas previas, sin tener físicamente los smartphones o revelar las intenciones de la prueba para acercarnos a la mayor probabilidad de éxito.

Información extra:

- **Android Debug Brigde. Verificado el 20 de Septiembre del 2015.**
http://developer.android.com/tools/help/adb.html

- **Instalación de una MV Android por línea de comandos. Verificado el 20 de Septiembre del 2015.**
http://developer.android.com/tools/devices/managing-avds-cmdline.html

Laboratorio No.2
"Escalación de privilegios en Android. El rooting"

Objetivo:

Conocer las implicaciones teóricas y prácticas del rooting a dispositivos móviles Android. Abordando un panorama general de lo que implica este proceso; realizar una demostración del mismo con un sistema operativo actualizado y las características en uso de una máquina virtual.

Requisitos:

-Tener un dispositivo móvil Android o contar con una instancia virtual de Android
-El dispositivo o la instancia tiene que tener depuración por USB activada
-En caso de un dispositivo real, adicionalmente se debe contar con los drivers de conexión USB instalados, el cable USB. El dispositivo debe poder realizar una conexión directa con el comando adb Shell abordado en el laboratorio anterior

Material:

-Una ROM compatible con el modelo de hardware y la versión de Android que se quiere instalar
-Una instancia virtual de Android
-App super su
-Aplicación super one click
-Android SDK instalado
-Samsung Galaxy S3 mini

Marco teórico:

El rooting es el proceso que permite al usuario obtener los privilegios de root (administrador o súper usuario) del sistema, este término es usado de esta manera; debido a que el sistema operativo Android está basado en el sistema operativo Linux, en el cual existe este usuario. El tener privilegios de root en un sistema, permite realizar modificaciones y cambios que un usuario común no podría; en caso de los teléfonos inteligentes, permite al usuario el control total del dispositivo.

Ventajas

-Instalar/Desinstalar aplicaciones del fabricante y de fuentes externas no oficiales.
-Extender la duración de la batería por tener un control de las Apps que ejecuta el sistema.

-Actualizar el dispositivo a las nuevas versiones aunque oficialmente no se haya liberado para el dispositivo.
-Control total del hardware lo que permite optimizar sus funciones (red, touchpad, cámara, etc.)
-Control casi total del software lo que permite optimizar sus funciones.

Desventajas y Riesgos

-Se anula la garantía.
-Los virus y hackers tienen mayor facilidad de comprometer al dispositivo y realizar una gamma de acciones mucho más compleja y variada que un dispositivo sin rooting.
-De no realizarse correctamente el proceso, se corre con el riesgo de dejar inservible el dispositivo, esto sucede cuando se hace uso de las ROM´s.

Tipos de rooting

La clasificación del rooting puede hacerse en función de su permanencia, es decir; por la capacidad que tiene de conservar los privilegios de root, sin importar los cambios que hagamos en el equipo. Esta clasificación puede ser de 2 tipos:

Profundo o permanente: Este tipo de rooting consiste en reinstalar todo el sistema operativo por una versión modificada que tenga activada los privilegios de root por defecto. Esta versión modificada de Android es conocida como ROM; para realizar este tipo de rooting es necesario tener presente: el modelo del dispositivo, versión de Android, fabricante y en algunos casos, la operadora telefónica del dispositivo.

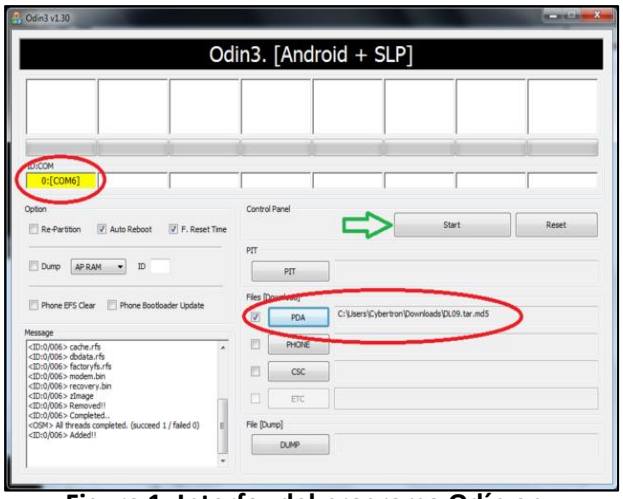


Figura 1. Interfaz del programa Odín en Windows 7

Para realizar este tipo de rooting es necesaria una herramienta que pueda administrar el bootloader del smartphone y que este se lo permita. Una de las más conocidas y populares, es Odín (Figura 1).

Esta herramienta permite instalar y manipular las ROMs que queramos instalar en nuestro smartphone.

Superficial o temporal: El rooting superficial consiste en mediante un programa o la aplicación de una serie de comandos, obtener privilegios de root; aprovechando una o varias vulnerabilidades de una App, que cuente con estos permisos o del mismo sistema operativo. Se conoce como temporal, porque al reiniciar el dispositivo, este vuelve a la normalidad; a partir de la versión 4.0 (Jelly Bean), este tipo de rooting es poco usual. En versiones anteriores entran aplicaciones como: z4root, gingerbreak, superoneclick; donde se puede aprovechar fallos del sistema operativo para adquirir privilegios de root. En la Figura 2 podemos observar la lista de procesos ejecutados en un smartphone, todos los que tienen como NAME, la ruta de la carpeta system/ corren con privilegios de root.

```
# ps
ps
USER     PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      27    1     740    316   c0158eb0 afd0d8ac S /system/bin/sh
system    28    1     808    264   c01a94a4 afd0db4c S /system/bin/servicemanage
r
root      31    1     668    256   c01b52b4 afd0e4dc S /system/bin/debuggerd
radio     32    1     5392   704   ffffffff afd0e1bc S /system/bin/rild
root      33    1     185256 25864 c009b74c afd0dc74 S zygote
media     34    1     22764  3644  ffffffff afd0db4c S /system/bin/mediaserver
root      35    1     812    316   c02181f4 afd0d8ac S /system/bin/installd
keystore  36    1     1616   408   c01b52b4 afd0e4dc S /system/bin/keystore
system    59    33    328372 44732 ffffffff afd0db4c S system_server
app_21    108   33    262460 20884 ffffffff afd0eb08 S com.android.inputmethod.l
atin
radio     113   33    272496 22836 ffffffff afd0eb08 S com.android.phone
app_20    118   33    274176 27404 ffffffff afd0eb08 S com.android.launcher
system    123   33    261900 19204 ffffffff afd0eb08 S com.android.settings
app_0     143   33    290908 29724 ffffffff afd0eb08 S android.process.acore
app_23    163   33    270548 23552 ffffffff afd0eb08 S com.google.process.gapps
```

Figura 2. Lista de procesos ejecutados en un smartphone Android, usando adb y el comando ps

Las máquinas virtuales y los privilegios de root

Una de las funcionalidades que tienen integradas las MV es tener privilegios de root sin la necesidad de tener que hacer cambios de ROM a la instancia virtual o realizar cambios de otra clase, esto nos permite la capacidad de subir archivos, bajar archivos, instalar aplicaciones, borrar archivos, etc. tal como sucede en Linux. Una ventaja que se tiene es que hay ROMs que se pueden probar mediante máquinas virtuales, previo a la instalación en el dispositivo, esto nos permite familiarizarnos con la ROM sin correr riesgo.

Procesos relacionados con el rooting

El adquirir privilegios de root de un dispositivo Android, requiere una serie modificaciones tanto en el funcionamiento del sistema operativo, como en el arranque del mismo dispositivo. Son necesarios estos cambios porque como se mencionó en el apartado de Android, no se pueden obtener recursos o privilegios adicionales una vez que los programas son instalados. El

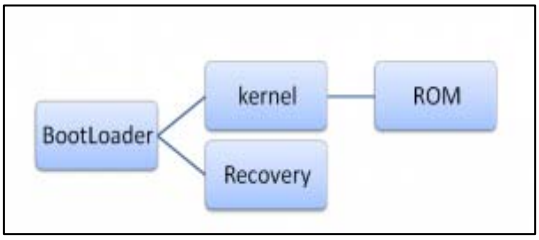


Figura 3. Proceso de arranque de un smartphone Android

proceso de arranque de un smartphone Android consta de varias fases en las que se realizan tareas específicas, en la Figura 3 podemos apreciar un breve diagrama de este proceso:

Bootloader: Es el primer programa en ejecutarse cuando se enciende un dispositivo móvil, este programa es el encargado de cargar el sistema operativo, el modo recovery y en algunos modelos en especial, el modo download del smartphone; este último es utilizado como una forma manual de actualizar a una versión de Android más reciente; desde la versión 4.0, el mismo sistema operativo se encarga de esta tarea.

Es necesario controlar y modificar el bootloader para realizar el proceso del rooting. La restricción de si el bootloader está liberado o no, depende del fabricante del smartphone, por ejemplo; el fabricante Samsung tiene liberado sus bootloader por defecto, pero en caso de HTC este lo tiene bloqueado y además se encuentra cifrado; en algunos otros casos el bootloader está bloqueado, pero nos permite cambiarlo o reinstalar una versión modificada; esto nos hace ver que no todos los smartphones se les puede aplicar el rooting, ciertos dispositivos son más fáciles que otros, por lo que se debe tomar en consideración este asunto desde el principio y averiguar las opciones posibles para realizar el rooting.

Recovery: Esta es una partición externa al sistema operativo Android y es cargada casi inmediatamente por el bootloader, es considerado como un OS primitivo que permite controlar el teléfono en su totalidad y permite ejecutar los comandos que permite el set de instrucciones de su línea de comandos, con esta partición podemos cargar y modificar las ROM de nuestro smartphone, algunos modelos tienen esta partición bloqueada, pero se trata de una minoría.

Kernel: Saber la versión del kernel es importante, porque su funcionamiento es distinto entre una versión y otra, por lo que no cualquier ROM podrá funcionar de manera correcta en nuestro smartphone, las ROM que se usan en la actualidad modifican el kernel para que este pueda funcionar correctamente con la versión que se instala; sin embargo, el riesgo añadido, es que este no funcione correctamente con el hardware, pudiese perder la comunicación con algún hardware del dispositivo, como puede ser la cámara, el bluetooth, algún sensor, etc.

ADB: El sistema operativo maneja dos tipos de usuarios, a cada uno se le permite una ejecución de comandos distinta, el usuario sin privilegios y root; el tipo de usuario es determinado por una propiedad en el sistema Android llamada **ro.secure**, si esta propiedad tiene de valor 0 el terminal adb puede ejecutar comandos como usuario root, en caso de que sea el valor 1 adb ejecuta comandos como un usuario no privilegiado. Este valor no puede ser cambiado, porque este se guarda en la carpeta raíz del sistema y estos valores son cargados al momento de la carga inicial del sistema operativo y solo pueden ser modificados con privilegios de root.

Caja de arena: Todas las aplicaciones de un usuario sin privilegios son ejecutadas en la caja de arena, esto evita que un programa sin privilegios pueda ejecutar un programa que tenga privilegios o que pueda ejecutar otra instancia de sí mismo con privilegios de root.

Procedimiento:

Superoneclick en una máquina virtual
Requisito: versiones anteriores a la 3.0

1. Iniciamos una máquina virtual en Android con una versión de 2.3 o inferior, 512MB de memoria, el resto de los parámetros quedan a elección del usuario. Una vez creada la ejecutamos y la mantenemos como ventana activa.

2. Descargamos la aplicación SuperOneClick y la ejecutamos con privilegios de administrador. En la Figura 4 podemos apreciar la apariencia de la ventana, esta aplicación utiliza dos vulnerabilidades para obtener privilegios de root, en el menú Exploit podemos verificar cuales utiliza. Entre las distintas opciones podemos observar las utilidades que podemos realizar con esta aplicación.

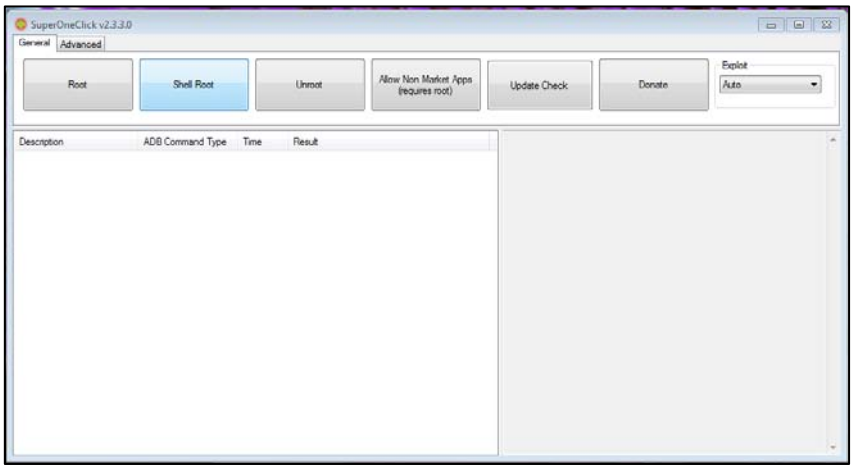


Figura 4. Ventana principal de SuperOneClick en Windows 7
Para que funcione correctamente debe ser ejecutada como Administrador.

3. Seleccionamos root y esperamos que el proceso termine. Podemos comprobarlo aplicando su desde el terminal emulator para comprobar que efectivamente tenemos permisos de root. Esto último lo conseguimos entrando a la opción DevTools del menú aplicaciones y en la última opción encontramos la aplicación, todo ello desde nuestra máquina virtual. Dentro de SuperOneClick en la parte derecha en gris podemos observar la serie de procesos que realiza la aplicación para obtener privilegios de root, podemos observar esto en el recuadro negro de la Figura 5.

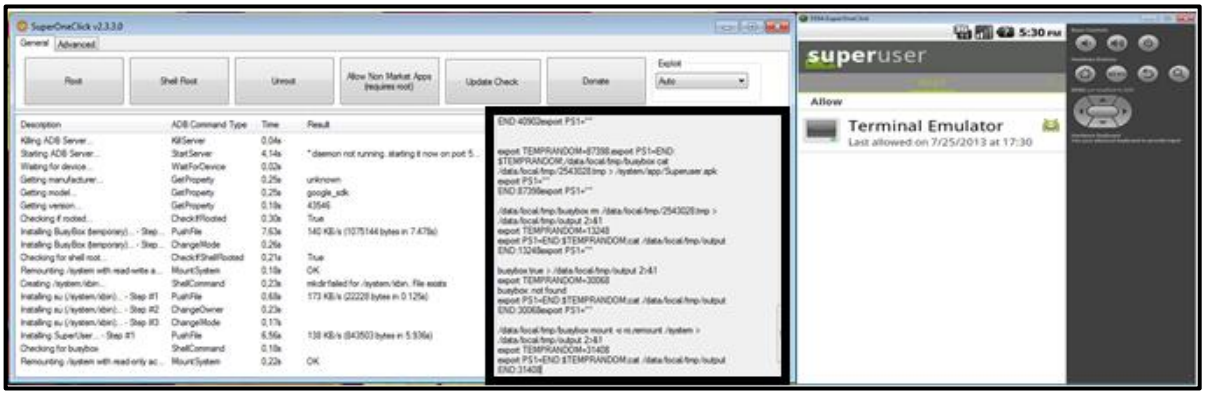


Figura 5. Rooting a una MV y uso de la aplicación Emulator

El recuadro negro, resalta la sección donde aparece la lista de comandos ejecutados.

Proceso largo para el rooting a un Samsung Galaxy S3 Mini

1. Descargamos e instalamos los siguientes programas.

-Descargar el programa Odín
http://forum.xda-developers.com/showthread.php?t=1738841
-Samsung Kies
http://www.samsung.com/es/support/usefulsoftware/KIES/JSP
-Una ROM para el modelo en cuestión
http://www.htcmania.com/showthread.php?t=540317

Nota: Es importante tener en cuenta el modelo del teléfono, en este caso, la banda telefónica influye en la elección de la ROM. Todos los enlaces fueron verificados el 25 de agosto del 2015.

2. Instalar los drivers USB del smartphone en cuestión para que lo reconozca nuestra pc, el programa Samsung Kies instala automáticamente estos drivers, una vez finalizada la instalación, podemos verificar que lo reconoce usando el comando **adb shell** (Recuerda que se debe estar posicionado en la carpeta en donde está el archivo adb).

3. Apagamos el smartphone y lo encendemos en modo download, para encenderlo en este modo, presionamos al mismo tiempo el botón de encendido, botón home y el de bajar volumen; una vez que el teléfono encienda en este modo nos dará una advertencia sobre los riesgos, pulsamos el botón subir volumen para aceptar, el smartphone cambiará de pantalla, esta nos muestra que está listo para reinstalar la ROM. Podemos apoyarnos de la Figura 6 para realizar este proceso.

Figura 6. Proceso para colocar el S3 mini en modo Download.
Fuente: HTCmania [16]

4. Abrimos ODÍN, conectamos el smartphone y este debe iluminarse la casilla ID: COM, y aparecer el mensaje Added en el panel Message. En el panel PDA damos el directorio en donde se encuentra nuestra ROM, finalmente pulsamos START, cuando en el panel Message aparezca el mensaje RES OK tendremos el Smartphone rooteado. En la Figura 1 observamos cómo debe ser está ventana previo al rooting.

Proceso corto para el rooting a un Samsung Galaxy S3 Mini

1. Descargamos e instalamos los siguientes programas.
   -Samsung Kies
   http://www.samsung.com/es/support/usefulsoftware/KIES/JSP
   -Samsung Galaxy S3 Mini Toolkit v1.1.
   http://forum.xda-developers.com/showthread.php?t=2185700

   Nota: Es importante tener en cuenta el modelo del teléfono, en este caso también influye la empresa que manufactura el dispositivo. Todos los enlaces fueron verificados el 25 de agosto del 2015.

2. Se ejecuta con permisos de administrador, las ventanas serán similares a las que se muestran en la Figura 7, será eligiendo la opción que corresponda, en primer lugar; seleccionaremos nuestro modelo (a). Si no hemos instalado los drivers del smartphone en este punto podemos realizar esa tarea, seleccionando la opción número 1 “Install Devices Drivers on your PC”, podemos ver esta opción en la parte 7b.

3. En la tercera ventana (c), seleccionamos la opción 1, instalar “Flash Insecure Image and Root device” y posteriormente escogemos el modo de súper usuario que se instalará (d), el programa tardará de 10 a 20 minutos, cuando finalice; tendremos nuestro dispositivo con privilegios de root.

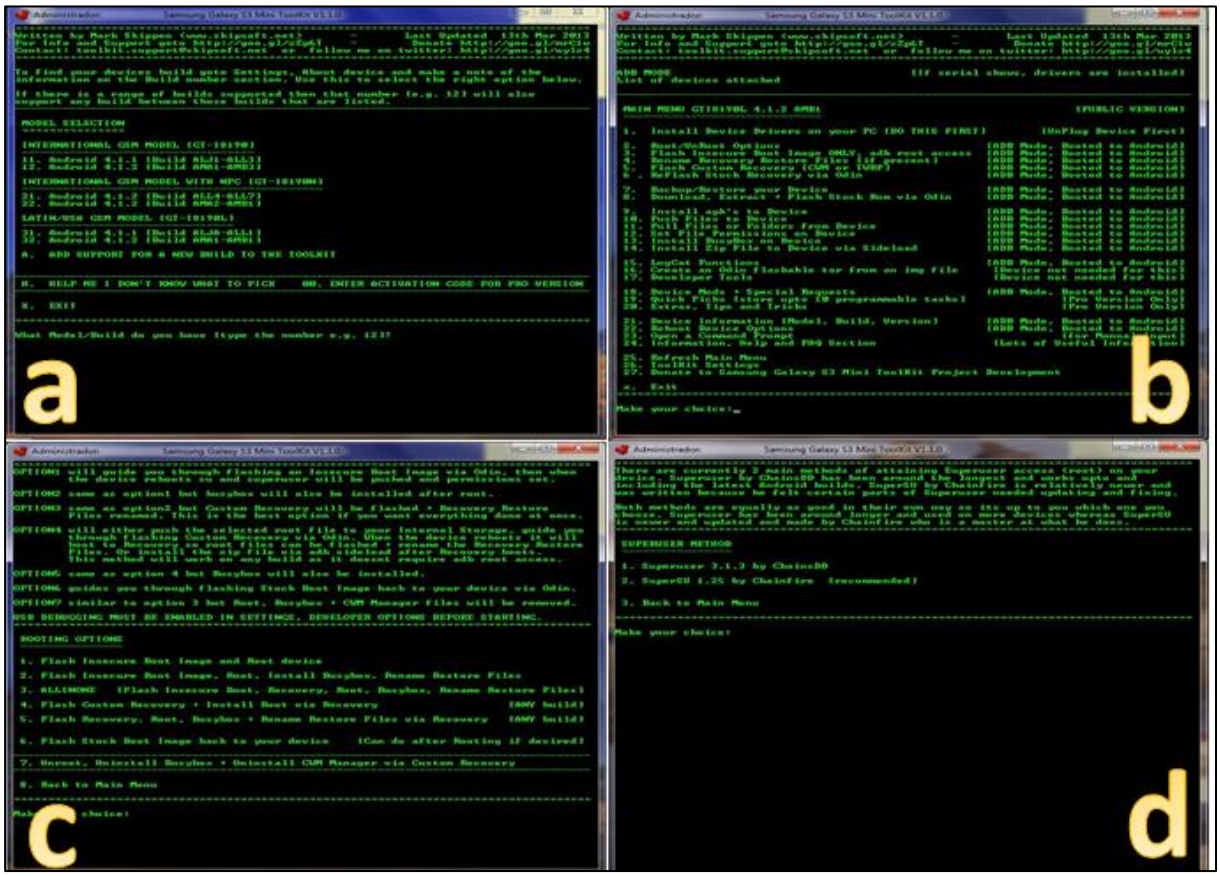


Figura 7. Rooting a S3 mini con la herramienta Samsung Galaxy Toolkit

Probar ROM´s externas al entorno de desarrollo de SDK.

1. Descargamos la ROM que pretendemos instalar.

2. Desde la carpeta tools del SDK, tecleamos el comando **Android créate avd –n <nombre> - t <Id de la versión>.** En la Figura 8 podemos ver el resultado de ejecutar este comando.
   Nota: Con el comando **Android list target** podemos ver los ID de las versiones**.**

```
C:\Program Files\Android\android-sdk\tools>android create avd -n ROM -t 3
Auto-selecting single ABI armeabi
Android 1.6 is a basic Android platform.
Do you wish to create a custom hardware profile [no]no
Created AVD 'ROM' based on Android 1.6, ARM (armeabi) processor,
with the following hardware config:
hw.lcd.density=240
vm.heapSize=24
```

Figura 8. Rooting a S3 mini con la herramienta Samsung Galaxy Toolkit

3. Copiamos nuestra ROM en la carpeta en donde está guardada nuestra máquina. Esta debe tener el mismo nombre, que el de nuestra máquina virtual que configuramos en el paso anterior.

4. Por último iniciamos nuestra máquina virtual para ver la nueva interfaz, en algunos casos la configuración de la ventana cambiará conforme a la ventana típica. En la Figura 9, podemos apreciar este hecho. En la instalación de un ROM modificada, puede cambiar la interfaz, los controles, los programas precargados, etc. Esto dependerá del desarrollador de la misma.

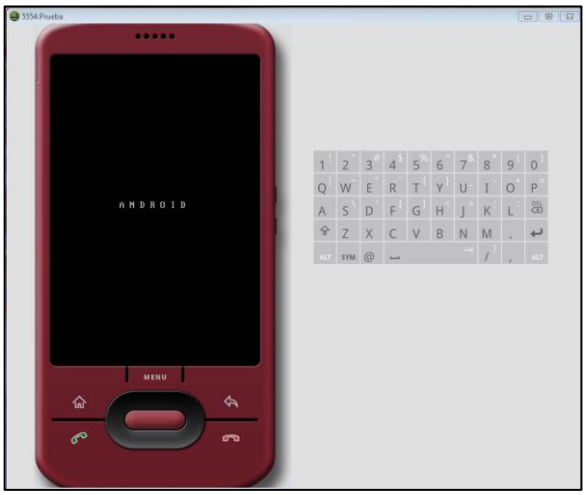


Figura 9. Ventana de nuestra máquina virtual modificada

Administración de privilegios de root de una máquina virtual

Nota: Este procedimiento es igualmente aplicable a un dispositivo real, con las consideraciones adicionales que indica esta práctica.

1. Descargar directamente de los enlaces: Superuser-3.0.7-efghi-signed.zip. El enlace fue verificado el 28 de agosto del 2015.

2. De la carpeta sacamos los archivos **su** y **Superuser.apk,** los pegamos en C:\Program Files\Android\android-sdk\platform-tools o en la carpeta donde tengamos instalado el SDK.

3. Iniciamos el ADB, posteriormente ejecutamos el comando **emulator** para iniciar nuestra instancia virtual. La sintaxis es: emulator –avd <Nombre máquina virtual> -partition-size <tamaño de la partición>. Este proceso se puede hacer de igual forma iniciando la máquina virtual desde AVD Manager. En la Figura 10 podemos observar el resultado de ejecutar los comandos desde ABD.

```
C:\Archivos de programa\Android\android-sdk\tools>emulator -avd Celular -partiti
on-size 512

C:\Archivos de programa\Android\android-sdk\tools>could not get wglGetExtensions
StringARB
could not get wglGetExtensionsStringARB
could not get wglGetExtensionsStringARB
could not get wglGetExtensionsStringARB
could not get wglGetExtensionsStringARB
could not get wglGetExtensionsStringARB
could not get wglGetExtensionsStringARB
could not get wglGetExtensionsStringARB
Failed to create Context 0x3005
emulator: WARNING: Could not initialize OpenglES emulation, using software rende
rer.
emulator: warning: opening audio output failed

emulator: emulator window was out of view and was recentered
```

Figura 10. Iniciar una máquina virtual desde ADB mediante línea de comandos

4. Posteriormente ejecutamos el comando **adb remount** en una nueva terminal de comandos, cuando el comando se haya ejecutado de manera correcta, en esta nueva terminal ejecutamos los siguientes comandos:
   adb push su /system/xbin/su
   adb shell chmod 06755 /system
   adb shell chmod 06755 /system/xbin/su
   adb install Superuser.apk
   Con este proceso estamos instalando el comando su y la aplicación de superuser para que podamos administrar nuestros permisos de root o conseguir que nuestro usuario actual, adquiera dichos privilegios en el momento que se le indique. En la figura 11 podemos observar la ejecución de dichos comandos.

```
C:\Archivos de programa\Android\android-sdk\platform-tools>adb remount
remount succeeded

C:\Archivos de programa\Android\android-sdk\platform-tools>adb push su /system/x
bin/su
324 KB/s (22364 bytes in 0.067s)

C:\Archivos de programa\Android\android-sdk\platform-tools>adb shell chmod 06755
 /system

C:\Archivos de programa\Android\android-sdk\platform-tools>adb shell chmod 06755
 /system/xbin/su

C:\Archivos de programa\Android\android-sdk\platform-tools>adb install Superuser
.apk
510 KB/s (843503 bytes in 1.612s)
        pkg: /data/local/tmp/Superuser.apk
```

Figura 11. Instalación del comando su y la App superuser, mediante ADB

4. Finalmente, desde nuestra máquina virtual, entramos a la venta de aplicaciones, seleccionamos el menú Dev Tools, posteriormente seleccionamos Terminal Emulator que se encuentra al final del menú y ejecutamos el comando **su,** si el comando funciona obtendremos permisos de super usuario. Una ventana emergente confirma este proceso. En la figura 12 se aprecia este hecho.

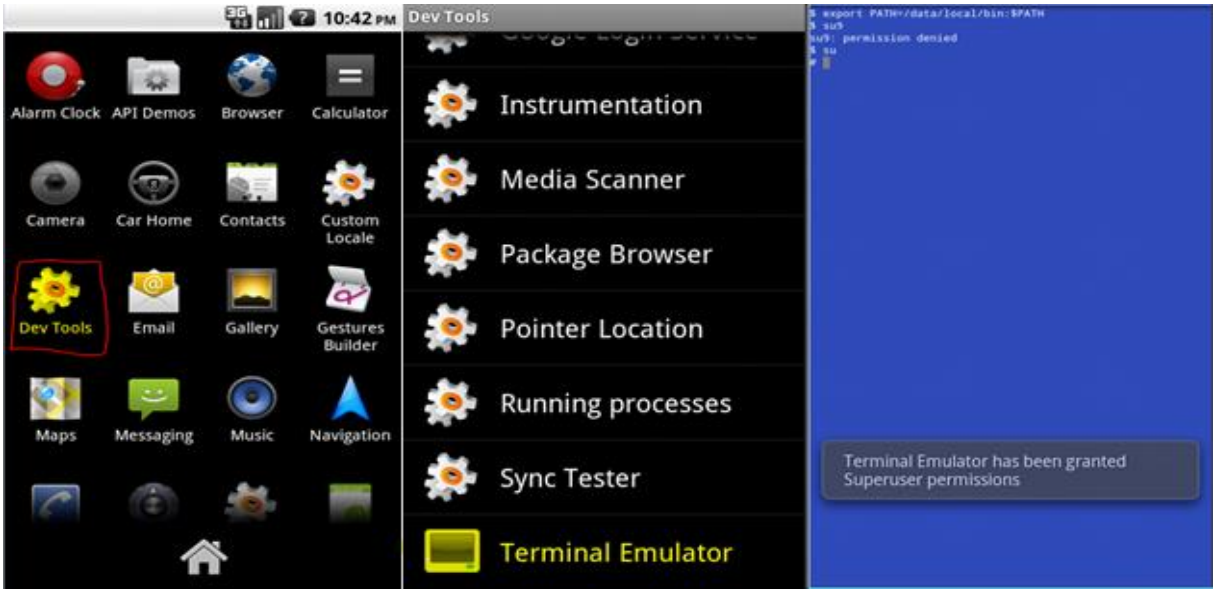


Figura 12. Ejecución del comando su en nuestra máquina virtual

Resultados:

Actualmente el rooting a un dispositivo se ha vuelto sencillo, en la mayoría de los casos; solo es necesario descargar un programa, seguir instrucciones y esperar. En un principio este proceso se trataba de algo complicado y con cierto nivel de riesgo, porque se requería realizar manualmente todo este proceso. A partir de la versión 3.0 de Android se sentó un antes y un después sobre lo que el proceso de rooting conlleva. Por ello, el proceso actual más común para realizar el rooting, es la reinstalación de la ROM del dispositivo.

Una pequeña cantidad de fabricantes, han dificultado la realización de este proceso; probablemente en protección de su código exclusivo que utilizan en sus dispositivos, sin embargo, en algunos casos, el usuario toma en cuenta este hecho en el momento en el que pretende adquirir un smartphone, tomando en cuenta que puede realizar su smartphone algo más personal y particular. Este hecho Google lo ha tomado en cuenta en el 2011 Google adquirió la empresa Motorola y parte de su política de venta en algunos modelos, es el permitir que el usuario adquiera su smartphone con el hardware de su elección; éste hecho remarca la importancia que tiene el nivel de personalización de un dispositivo.

La condición de root de un dispositivo Android, debe ser tomada en cuenta durante la fase de análisis de vulnerabilidades para que podamos tener en cuenta el grado de exposición que tiene el dispositivo y desarrollar la mejor posibilidad de éxito para acceder al mismo.

Información extra:

- Riesgos del rooting, artículo en inglés. Verificado el 28 de agosto del 2015. http://www.bullguard.com/bullguard-security-center/mobile-security/mobile-threats/android-rooting-risks.aspx

- Foro HTCmania para encontrar ROMs e información útil para el rooting en español. Verificado el 28 de agosto del 2015. http://www.htcmania.com/portal.php

- Foro cyanogenmod para encontrar ROMs e información útil del rooting en inglés, es considerado de lo más grandes en internet. Verificado el 28 de agosto del 2015. https://download.cyanogenmod.org/

- Foro XDA Developers para encontrar ROMs e información útil del rooting en inglés. Verificado el 28 de agosto del 2015. http://forum.xda-developers.com/

Laboratorio No.3
"Acceso no autorizado. Lockscreen by pass Android"

Objetivo:

Conocer los principales métodos de bloqueo de pantalla de los smartphones Android, enfocándose en algunas vulnerabilidades que estos métodos tienen, para poder evadirlos y tener acceso no autorizado al dispositivo. Al reconocer las características de las formas de bloqueo, el alumno podrá comprender el proceso y dificultad de tratar de eludirlas.

Requisitos:

-En algunos casos privilegios de root

Smartphone real:
-Depuración USB activada
-Drivers de instalación USB del smartphone

Material:

-SDK de Android
-Smartphone con los requisitos del apartado anterior
-Viber.apk Versión 2.3.6
-Un número de celular vigente
-Fotografía personal

Marco teórico:

Se conoce como Bypass cuando se altera el uso normal de datos o acciones de un dispositivo con el fin de modificar su comportamiento. Este término se aplica principalmente cuando se quiere tener acceso al smartphone, sin importar que se encuentre bloqueado por algún mecanismo de seguridad, este puede ser un patrón de dibujo, reconocimiento facial o de voz, contraseña, etc. Esta acción de evadir el bloqueo de pantalla de un smartphone de manera no autorizada, se conoce como Lockscreen Bypass.

En materia de seguridad, es importante conocer las características de los métodos de bloqueo de un smartphone, porque como hemos analizado, estos pueden contener información sensible que pudiera poner en riesgo a una organización, si esta información estuviese en manos equivocadas.

Se puede conseguir evadir los bloqueos de teléfono de diversas maneras; antes de realizar cualquier prueba, es conveniente tener en consideración cuestiones como las siguientes, qué nos permitirán dimensionar y comprender los métodos existentes que hay para realizar un bypass:

- Ningún producto es perfecto, por lo que tiene errores y/o límites que de superarse; pueden conseguir un comportamiento extraño al común.
- Las contraseñas o mecanismos de seguridad se guardan como información dentro del smartphone, si conocemos su ubicación y la forma de borrarlos; podemos eliminar dicha protección.
- No existe un mecanismo de seguridad 100% seguro. Por lo que se deben conocer las ventajas y desventajas de cada uno.
- Las aplicaciones o tendencias del dueño pueden ser un punto clave para evadir mecanismos de seguridad.

A continuación se presentan los métodos de seguridad de bloqueo de pantalla de los smartphones Android. Las consideraciones de seguridad son clasificadas por el mismo smartphone. Estos mecanismos son los más conocidos presentes en la mayoría de los smartphones Android, en algunos casos; los fabricantes diseñan métodos propios como parte de su desarrollo personal y solo aplican a sus modelos, un ejemplo de ello es el número 4 Knock Code.

Tabla 1. Métodos de bloqueo generales en Android

Método de Bloqueo	Seguridad	Versiones
1. Ninguna	Ninguna	4.4 en Adelante
2. Deslizar	Ninguna	Todas
3. Movimiento	Ninguna	Todas
4. Knock Code	Baja	**4.4 en Adelante**
5. Bloqueo por reconocimiento facial	Baja	4.0 en Adelante
6. Bloqueo reconocimiento facial y de voz	Baja	4.0 en Adelante
7. Patrón	Media	Todas
8. PIN	Alta	Todas
9. Contraseña	Muy Alta	Todas

En las Figuras 1 y 2 podemos observar la pantalla o el funcionamiento de cada uno de los bloqueos de pantalla aquí presentados.

1. **Ninguna:** Este método de bloqueo solo existe para evitar que el smartphone se auto-marque o realice acciones mientras se encuentre en la bolsa de su dueño o en su carcasa. Consiste en un botón central que al presionarlo hacia el contorno de la figura, desbloquea el dispositivo. Se puede configurar de tal forma que active una función adicional, dependiendo del movimiento realizado.

2. **Deslizar:** Este método tiene el mismo propósito que el anterior, solo se requiere deslizar la barra de izquierda a derecha para desbloquear el smartphone.

3. **Movimiento:** Este método de bloqueo consiste en presionar la pantalla y mover el smartphone hacía nosotros. Tiene la misma finalidad que el método anterior.

4. **Knock Code:** Es una tecnología propia de LG, en algunos de sus modelos se incorporó este método de bloqueo como un añadido especial. Consiste en dividir la pantalla en 4 secciones y el usuario debe tocar cada cuadrante en un orden determinado para desbloquear el smartphone.

5. **Reconocimiento facial:** Este método de bloqueo consiste en guardar un patrón de nuestra cara y posteriormente para desbloquear el smartphone solo se requiere colocar este a la altura de la cara.

   Para evadir este mecanismo con una foto del dueño del smartphone (de frente) podemos desbloquearlo, el mecanismo no reconoce si se trata de una persona o una foto, incluso el mismo sistema operativo nos advierte que una persona con rasgos similares puede desbloquear el smartphone. Actualmente hay algoritmos que nos piden adicionalmente abrir y cerrar los ojos, sin embargo; no es una medida efectiva.

Figura 1. Métodos de bloqueo en Android. 1. Ninguno, 2. Deslizar, 3. Movimiento, 4. Knock Code y 5. Reconocimiento facial

6. **Bloqueo facial y voz:** Este método es el mismo que el anterior, pero aparte añade reconocimiento por voz de una palabra de 3-5 letras.

7. **Bloqueo mediante patrón**: El bloqueo por patrón consiste en guardar un patrón de dibujo en base a puntos de referencia (círculos), puede tener desde 2 o 3 marcas como mínimo, hasta 9 o 16 dependiendo el tipo de smartphone. Evadir este tipo de patrón puede ser mediante adivinar el patrón (se puede observar a contra luz para ver los movimientos de los dedos) o probar con las combinaciones más fáciles que se acostumbran a usar.

8. **Bloqueo mediante PIN**: Este método consiste en guardar una secuencia de números que en las primeras versiones se podían hasta 4, sin embargo; en las versiones actuales, es posible utilizar más combinaciones.

9. **Contraseña:** Es una combinación de caracteres para formar una contraseña, es el método que mejor seguridad ofrece.

Figura 2. Métodos de bloqueo en Android. 6. Voz, 7. Patrón, 8. PIN y 9. Contraseña

Las Apps y sus privilegios

Toda App y en general cualquier proceso requiere de permisos y recursos para que esta funcione correctamente. Es necesario que sean bien definidos en su instalación, con la finalidad de tener una correcta administración de los recursos del sistema.

Todas las Apps tienen fallos, ya sea en su diseño o en su implementación, estos defectos pueden ser aprovechados para evadir mecanismos de seguridad u obtener datos. En el capítulo I se comentó que Android no verifica de manera estricta la cantidad de permisos que obtiene una App en su instalación. Estas circunstancias permiten, el robo de información o la obtención de privilegios que no son necesarios.

Existen Apps que tras actualizarse, cambian sus configuraciones, y estas mismas pueden evadir los mecanismos de seguridad (bloqueo), como ejemplo tenemos dos aplicaciones: Viber a principios del año 2013 tenía un problema porque al mandar mensajes

desbloqueaba el smartphone, y más recientemente en julio del 2013 Skype tenía un problema similar al momento de realizar llamadas.

Todo programa que funcione en un sistema, está guardado como un archivo en su respectivo, esto incluye los mecanismos de seguridad; de borrarse estos archivos, el mecanismo dejará de funcionar.

Procedimiento:

Método # 1 Desbloquear el reconocimiento facial y de voz. Método probado en la Versión de Android 4.1.2

1. Si solo es reconocimiento facial, basta con conseguir una foto frontal del dueño del smartphone y colocarla en frente del sensor para que la reconozca. Existen muchos métodos para obtener una fotografía, se puede hacer uso de redes sociales, tomarle incluso una foto, u mediante otros medios.

2. Si además tiene reconocimiento por voz, se requiere una grabación del dueño o de un tono voz parecida, que pronuncie la palabra que tiene guardada, la única dificultad en este caso es saber cuál es la palabra usada, sin embargo; como es de 3-5 letras no resulta muy difícil tratar de adivinarla.

Método # 2 Desbloquear cualquier método de bloqueo mediante estresar el Smartphone. Versión de Android 4.1.2 para dispositivos Samsung (Existe actualización que corrige el fallo)

1. Tener bloqueado el teléfono.
2. Presionamos en Llamadas de emergencia.
3. Contactos
4. Lo más rápido posible presionamos home
5. Botón de encendido.

El método consiste en realizar esta secuencia una gran cantidad de veces (En algunos casos más de 40) de ser exitoso evade el mecanismo de seguridad.

Video demostrativo:
https://www.youtube.com/watch?v=CEIZXRfnR1c

Método # 3 Desbloquear cualquier método de bloqueo mediante aplicaciones. Viber versión 2.3.6 (Marzo 2013, existe actualización que corrige el fallo) y Skype versión 3.2.0.6673 (Julio 1, 2013).

1. Desde otro smartphone mandar un mensaje.

2. Cuando lo reciba la víctima debe presionar el botón de la pantalla del mensaje como si fuese a escribir (el teléfono se bloqueará de nuevo).

3. Mandar nuevamente un mensaje a la víctima, cuando este llegue aparecerá el teclado, solo se debe presionar el botón cancelar y entonces se habrá saltado la pantalla de bloqueo.

Método # 4 One Pattern Click Unlock aplicación que borra contraseñas.
Requiere depuración USB activada. Versión Android 2.X y solo para dispositivos Samsung.

1. Descargar la aplicación.

2. Ejecutar la aplicación y rellenar los datos que se solicitan.

Figura 3. Interfaz de One Click Pattern Unlock

Método # 5 Borrando los archivos que guardan las contraseñas. Requiere privilegios de root y depuración USB activada. Versiones anteriores de la 3.0 no requieran privilegios de root.

1ra Forma – Cuando se conozca el archivo se procede a borrar

1. Conectar el dispositivo a la PC e iniciar el **adb Shell** (Debemos asegurarnos que contamos con privilegios). Este proceso también puede realizar con una máquina virtual, en este caso; iniciamos la máquina virtual y nos conectamos a ella.

2. Insertamos los siguientes comandos:

   rm /data/system/password.key
   rm /data/system/gesture.key

   Al ejecutarlos se borra el archivo que contiene la contraseña en caso del password.key borra las contraseñas y PIN, en caso del gesture.key borra los patrones de dibujo.

```
C:\Archivos de programa\Android\android-sdk\platform-tools>adb shell
# rm /data/system/password.key
rm /data/system/password.key
# rm /data/system/gesture.key
rm /data/system/gesture.key
#
C:\Archivos de programa\Android\android-sdk\platform-tools>
```

Figura 4. Ejecución de comandos para borrar archivos de contraseña

3. Reiniciar el smartphone y veremos que la contraseña ha sido borrada. **Nota:** En caso que sea bloqueo por patrón de dibujo, la pantalla de bloqueo seguirá apareciendo, pero cualquier dibujo será válido. El mismo resultado se aprecia en la máquina virtual.

2da Forma

1. Conectar el dispositivo a la PC e iniciar el **adb Shell** (Debemos asegurarnos que contamos con privilegios). Este proceso también puede aplicarse a una máquina virtual.
2. Teclear lo siguiente:

cd /data/data/com.android.providers.settings/databases
sqlite3 settings.db
update system set value=0 where name='lock_pattern_autolock';
update system set value=0 where name='lockscreen.lockedoutpermanently';
.quit

```
C:\Archivos de programa\Android\android-sdk\platform-tools>adb shell
# cd /data/data/com.android.providers.settings/databases
cd /data/data/com.android.providers.settings/databases
# sqlite3 settings.db
sqlite3 settings.db
SQLite version 3.7.11 2012-03-20 11:35:50
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> update system set value=0 where name='lock_pattern_autolock';
update system set value=0 where name='lock_pattern_autolock';
sqlite> update system set value=0 where name='lockscreen.lockedoutpermanently';
update system set value=0 where name='lockscreen.lockedoutpermanently';
sqlite> .quit
.quit
#
```

Figura 5. Ejecución de comandos para desactivar los parámetros

En este caso se desactivan los mecanismos de seguridad del dispositivo.

3. Reiniciar el smartphone y veremos que la contraseña ha sido borrada. Este método no funciona en todos los smartphones.

Resultados:

La forma de evadir los mecanismos de seguridad de los smartphones Android va cambiando con el tiempo, lo importante no es saber cómo desbloquearlo en un momento determinado, sino la razón del por qué suceden estos problemas. El comprender este punto resulta clave porque de ello depende que podamos desarrollar medidas de seguridad para erradicarlas o no. Los Bypass y sus soluciones, irán cambiando conforme pasa el tiempo; por ello siempre resultará importante tener conocimiento de nuestra infraestructura para evitar tener pérdidas, por el robo o el mal uso de la información.

Esta información puede ser utilizada durante las fases de análisis de vulnerabilidades y modelado de amenazas, para poder obtener los datos necesarios y posteriormente efectuar las pruebas en la fase de explotación.

Información extra:

- **One Click Pattern Unlock Tool.** El enlace fue verificado el 1 de septiembre del 2015. http://forum.gsmhosting.com/vbb/f162/one-click-pattern-unlock-1511380/

- **Noticia de Viber con vulnerabilidades de la versión 2.3.6.** El enlace fue verificado el 1 de septiembre del 2015. http://www.bkav.com/top-news/-/view_content/content/46264/critical-flaw-in-viber-allows-full-access-to-android-smartphones-bypassing-lock-screen

- **FROST el método para recuperar PIN congelando el Smartphone. Por la universidad FAU de Alemania.** El enlace fue verificado el 1 de septiembre del 2015. https://www1.informatik.uni-erlangen.de/frost

- **USB Teensy, un Arduino modificado para atacar por fuerza bruta a los bloqueos por PIN.** El enlace fue verificado el 10 de septiembre del 2015. http://jumpespjump.blogspot.mx/2013/06/online-brute-forcing-android-pin-lock.html

CAPÍTULO III

Laboratorios de pruebas en iOS

Laboratorio No.4
"Virtualización iOS y Jailbreak"

Objetivo:

Conocer la herramienta Xcode, SDK oficial de Apple para desarrollar Apps en iOS, conociendo sus características y funcionalidades. Adicionalmente se abordará el proceso teórico y práctico del Jailbreak, tomando en cuenta las principales características de este proceso.

Requisitos:

-Mac OSX 10.3 o superior
-iTunes

Material:

-Xcode
-Computadora iMac con el requisito anterior
-iPhone 3G con iOS 4.1.0
-iPhone 3GS con iOS 6.1.3
-iDevice que soporte iOS 8.0
-Una computadora con Windows
-iTunes instalado

Marco teórico:

Xcode es el IDE oficial de desarrollo de Apps para iOS de Apple. Este IDE permite compilar código para desarrollar Apps en diferentes lenguajes de programación como son: C, C++, Swift, Objective-C, Objective-C++, Java, Cocoa y Carbon. Xcode posee un depurador de código, optimizador de código, bitácora de errores, simulador de iOS, herramientas para exportar, adaptar y colocar código en la nube para el trabajo simultáneo entre varios desarrolladores.

Este IDE es gratuito, pero se encuentra únicamente disponible para computadoras con sistema operativo OS X; es posible desarrollar Apps desde otras plataformas, pero para que puedan ser comercializadas en la Appstore, deben contar con una firma digital que las respalde como aprobadas, esta firma solo se obtiene mediante Xcode.

Xcode tiene una interfaz de usuario amigable, enfocada en el propósito de tener en una sola ventana, todas las herramientas, vistas, menús que el desarrollador necesita

mientras programa una nueva App; en la Figura 1 podemos observar la ventana principal de Xcode.

La **Barra de herramientas** tiene la finalidad de manejar el tamaño y la visibilidad de las otras ventanas, adicionalmente podemos encontrar los controles básicos para poder utilizar el simulador de iOS.

En el **área de navegación** podemos encontrar la estructura de nuestro proyecto muy parecido a los IDE de Java como Eclipse, adicionalmente contamos con una barra de controles en la parte superior de esta ventana que nos permite visualizar otras vistas como son: La de pruebas, las de compilación, las de administración de errores, las de administración de recursos, entre otras.

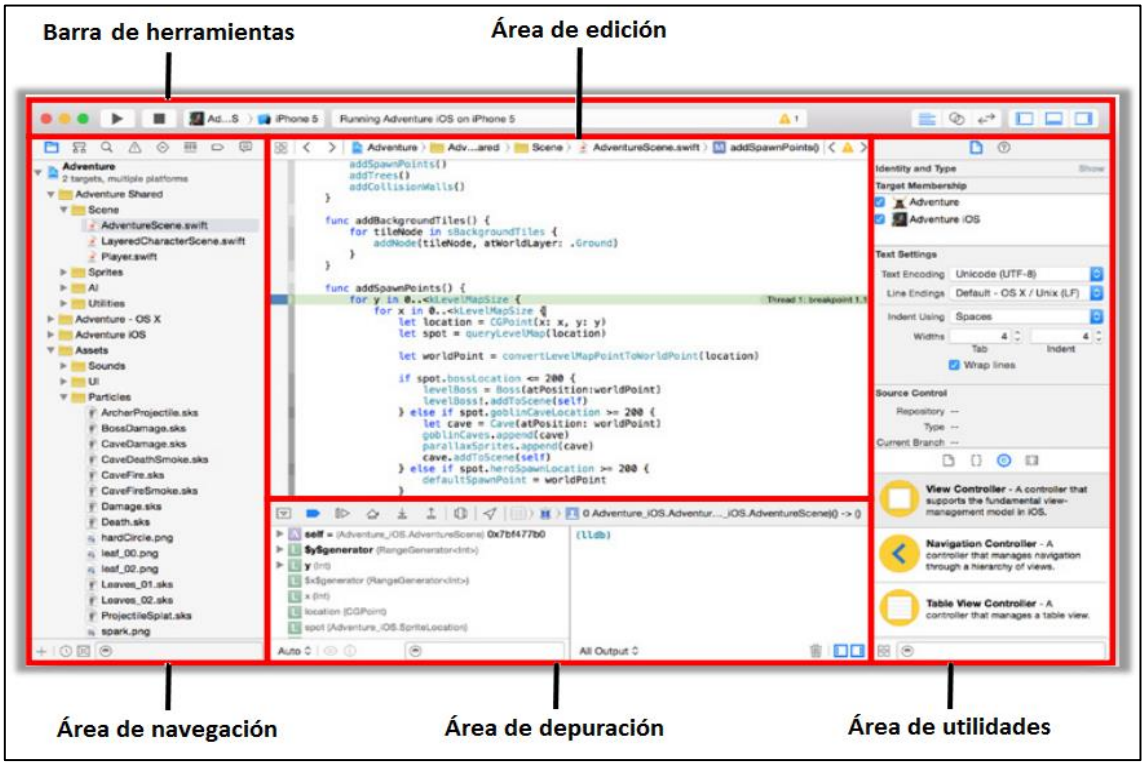


Figura 1. Ventana principal de Xcode
Fuente: Guía de usuario Xcode en inglés [22]

En la **ventana edición** podemos encontrar el contenido de cualquier archivo de nuestro proyecto y la posibilidad de modificar su contenido en él; a diferencia de cualquier otra ventana, esta se encontrará fija en el centro de la ventana principal. En el **área de depuración** podemos observar, nuestras variables y su funcionamiento, la depuración del programa y la ejecución del mismo.

En el **área de utilidades** podemos modificar las vistas, atributos y la forma en que funciona la interfaz de usuario de nuestro proyecto, también se incluyen herramientas y controles típicos que suelen encontrarse en las Apps.

Simulador iOS

Al igual que en Android Studio, Xcode posee un simulador para su plataforma móvil; esta nos proporciona un acercamiento inicial al comportamiento que tendría una App que nos encontremos desarrollando en un dispositivo real. La diferencia con Android Studio, va enfocada en que el simulador de iOS está específicamente desarrollada para realizar pruebas de rendimiento y depuración sobre Apps.

Una característica importante del simulador es que nos permite simular varios tipos de hardware, en distintos tipos de versiones de iOS, adicionalmente; mientras se realizan pruebas con la App, es posible saber el rendimiento y el consumo de recursos de la App, mientras esta se encuentre en ejecución.

Jailbreak

Es el proceso de obtener los privilegios de root en un iDevice mediante aprovechar una vulnerabilidad que el dispositivo tenga. El nombre de Jailbreak lo recibe porque al obtener privilegios de root, las restricciones de la caja de arena desaparecen, permitiendo al usuario acceder en su totalidad al software y hardware del dispositivo, la oportunidad de instalar Apps que no pertenecen a la Appstore, etc.

Por el tipo de vulnerabilidad explotada para alcanzar dichos privilegios, podemos encontrar dos tipos de Jailbreak, los persistentes (Untethered) que no se pierden los privilegios aunque se reinicie el dispositivo, este tipo de Jailbreak aprovecha fallos en el hardware o sectores de bajo nivel del sistema operativo o del mismo sector de arranque. El Jailbreak no persistente (Tethered) que al reiniciarse el dispositivo se pierden los privilegios, aprovecha vulnerabilidades principalmente de servicios primarios del sistema operativo, que al reiniciarse se restauran a su condición original; en este tipo de Jailbreak es posible no obtener un control total del dispositivo, de ello dependerá la vulnerabilidad utilizada.

El método para realizar el Jailbreak depende de dos factores, el hardware del dispositivo y de la versión de iOS. En un comienzo, el proceso del Jailbreak era un proceso complicado, porque requería del usuario, conocimiento técnico más avanzado; por ejemplo, modificar el bootloader del dispositivo, durante este proceso se corría el riesgo de realizarlo de manera equivocada y dañar el dispositivo de manera permanente. Con las versiones más actuales y ante el poco cambio del hardware de los dispositivos, se encontraron vulnerabilidades más sencillas de explotar y se automatizó el proceso, para la versión 8.0; el proceso del Jailbreak consiste en descargar un programa y ejecutarlo. En la Tabla 1 se puede observar las herramientas necesarias para realizar el Jailbreak dependiendo de la versión de iOS:

Tabla 1. Herramientas para realizar el Jailbreak

VERSIÓN	HERRAMIENTA
1.X	AppsnApp, iLiberty
2.X	PwnageTool, QuickPwn
3.X	PwnageTool, redsn0w
4.X	redsn0w, PwnageTool
5.X	Untheredal1n, redsn0w
6.X	Evasi0n, redsn0w
7.X	Evasi0n7, Pangu
8.X	Pangu8, TaiG

En términos generales un Jailbreak del tipo persistente afecta los siguientes procesos de iOS:

Boot ROM: Es un código de solo lectura que se utiliza como motor de arranque del dispositivo, adicionalmente; se encuentran una llave pública para verificar la integridad del resto del sistema operativo y cualquier otro componente, al descifrarse pueden realizarse modificaciones al sistema operativo.

Low Level Bootloader: Es el primer archivo de arranque de iOS tiene la finalidad de verificar la integridad de los siguientes archivos de arranque (Por cuestión del tamaño de memoria) y entrar en modo DFU (Device Firmware Upgrade).

Next Stage Bootloader: O también conocido como iBoot es el archivo de arranque de alto nivel que funciona como un modo Recovery, desde este punto se empieza a carga el kernel de iOS y el resto de los archivos del SO.

Procedimiento:

Uso general y hola mundo en iOS Simulator

1. Desde el escritorio MAC podemos utilizar Finder pulsando las teclas **Cmd +Tab** y tecleamos Xcode para que busque la aplicación, si la tenemos instalada aparecerá en la ventana de búsqueda y la seleccionamos (Figura 2a).

2. Abrimos Xcode y seleccionamos un nuevo proyecto (Create New Xcode Project). Podemos observar este paso en la Figura 2b.

3. Seleccionamos la opción “Single View Aplication” de la sección de iOS (Figura 2c).

4. Rellenamos la siguiente ventana con los datos de nuestro proyecto, en esta ventana podemos observar diversos campos, en Language podemos seleccionar el lenguaje de programación a utilizar en nuestro proyecto, Objective C o Swift; en Devices podemos seleccionar si se trata de una iPad o un iPhone o como observamos en la Figura 2d podemos seleccionar Universal, está opción crea plantillas estándar que permiten a la App funcionar en ambos tipos de dispositivos.

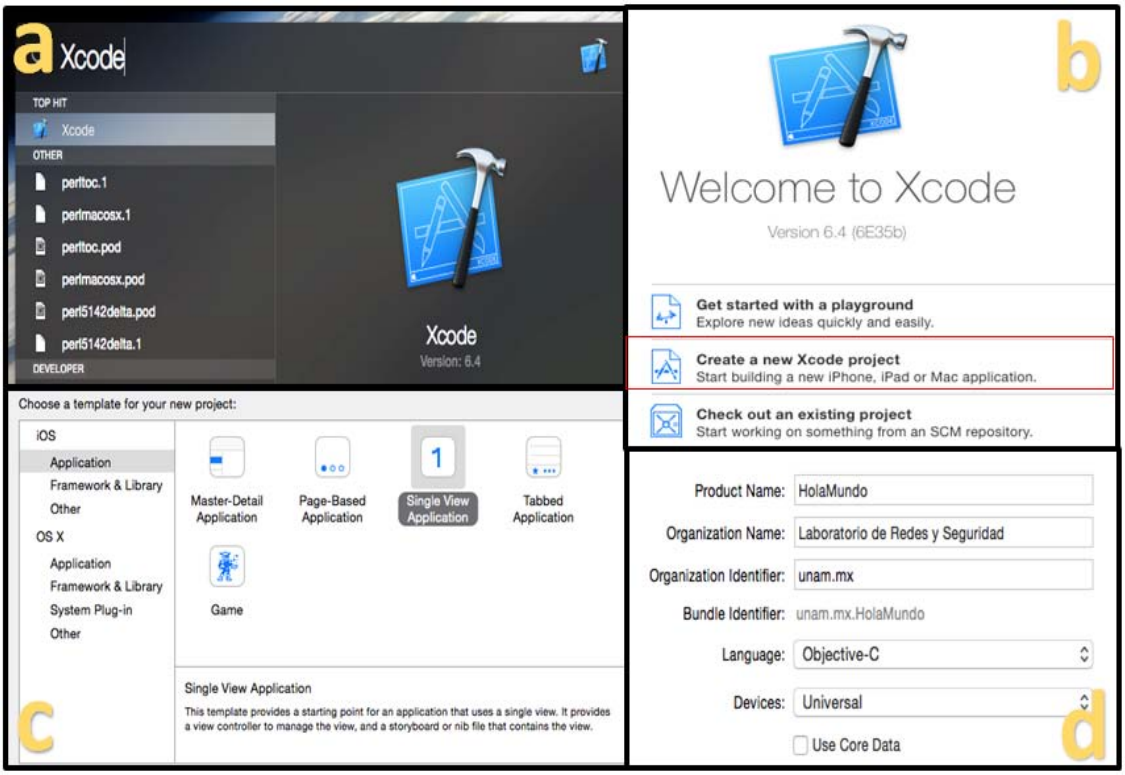


Figura 2. Creación de nuevo proyecto en Xcode

5. Posteriormente aparecerá la ventana principal de Xcode como la mostrada en la Figura 1. La primera acción es probar el simulador iOS. En la barra de herramientas podemos observar en que dispositivo trabajará el simulador, podemos cambiarlo seleccionando el botón y cambiando de dispositivo, para iniciarlo seleccionamos el botón de play o run de la misma barra de herramientas. Otra opción es el acceso rápido con las teclas **Cmd + R.**

6. Si es la primera vez que funciona el compilador, nos pedirá que autoricemos nuestra computadora como de confianza, al aceptar esta petición, una nueva ventana iniciará con el simulador de iOS, por defecto, solo trae las aplicaciones que todos los iDevices tienen por defecto.

7. Para programar el Hola Mundo seleccionamos en el área de navegación la carpeta HolaMundo o el nombre que colocamos, posteriormente abrimos la carpeta y seleccionamos el archivo Main.storyboard.

8. Posteriormente en el área de utilidades en la parte inferior se encuentran los controles gráficos Ulkit, en esta sección se encuentra una barra de búsqueda, buscaremos y colocaremos los siguientes controles:

- Crear un Label y a la derecha en las opciones cambiar Text a "Hola Mundo".
- Crear un Text Field y colocarlo justo debajo del Label.
- Crear un Button (Round Rect Button), colocarlo debajo del Text Field y cambiar su Title a "Hola".

Al terminar de colocar estos controles, iniciamos nuestro simulador iOS y aparecerá en pantalla, algo similar a la Figura 3.

Figura 3. Simulador iOS con los objetos creados.

9. En la barra de herramientas, en la parte superior derecha, seleccionamos el segundo botón “Show the Assistant Editor” de manera que veamos a la vez dos editores. Esto podemos observarlo en la Figura 4.

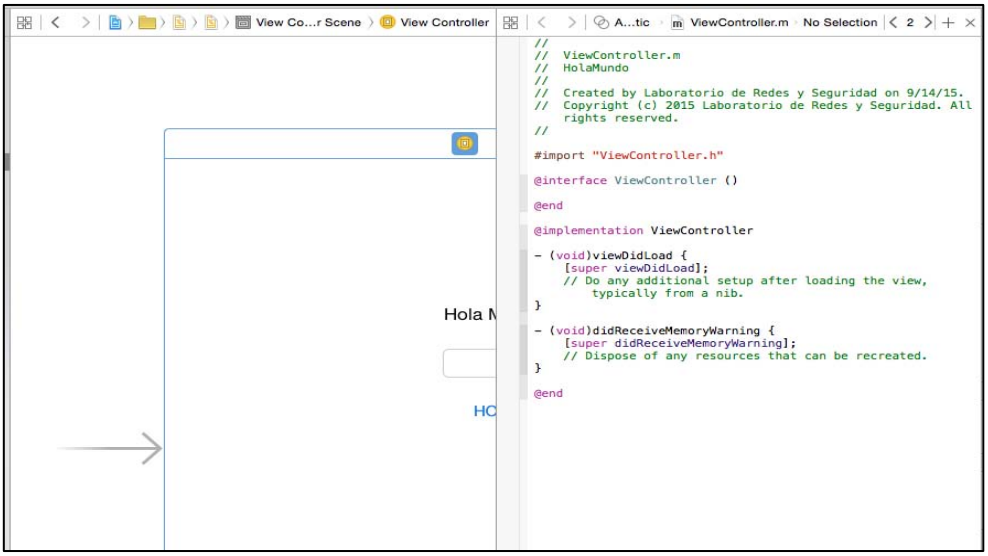


Figura 4. Xcode utilizando la opción Assistant Editor.

10. A la derecha del botón Assisant tenemos una ruta del Storyboard, modificaremos dos archivos de nuestro proyecto, el primero es ViewController.h, en este crearemos los

controles necesarios para nuestros objetos mediante propiedades. En la Figura 5a podemos observar cómo quedará este archivo.

11. Posteriormente seleccionamos el archivo HolaMundoViewController.m y en este se creará el método que será utilizado por el botón para realizar una acción determinada, en este caso desaparecer la etiqueta de Hola Mundo. En la Figura 5b podemos ver la versión final del código.

a

```
//
//  ViewController.h
//  HolaMundo
//
//  Created by Laboratorio de Redes y Seguridad on 9/14/15.
//  Copyright (c) 2015 Laboratorio de Redes y Seguridad. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *Label;
@property (weak, nonatomic) IBOutlet UITextField *texto;
@property (weak, nonatomic) IBOutlet UIButton *boton;
@end
```

b

```
//
//  ViewController.m
//  HolaMundo
//
//  Created by Laboratorio de Redes y Seguridad on 9/14/15.
//  Copyright (c) 2015 Laboratorio de Redes y Seguridad. All rights reserved.
//

#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController
- (IBAction)accionHola:(id)sender {
    self.Label.text = self.texto.text;
}

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

Figura 5. Códigos necesarios para programar un método en objective C

12. Finalmente iniciamos el simulador de iOS y probamos que funcione dando click en el botón Hola, el mensaje hola mundo deberá desaparecer.

Jailbreak mediante redsn0w a un iPhone 3G con iOS 4.1

1. Descargamos el programa redsn0w del siguiente enlace: http://www.iphonehacks.com/download-redsn0w

2. Conectamos el dispositivo y verificamos que el programa lo detecte. Se puede observar esto en la Figura 6a.

3. Seleccionamos Jailbreak y esperamos un momento a que cargue los datos necesarios. Posteriormente seleccionamos las opciones que deseamos instalar y pulsamos siguiente (Figura 6b).

4. Apagamos nuestro Smartphone y lo colocamos en modo DFU. Podemos apoyarnos del programa para realizar este proceso (Figura 6c).

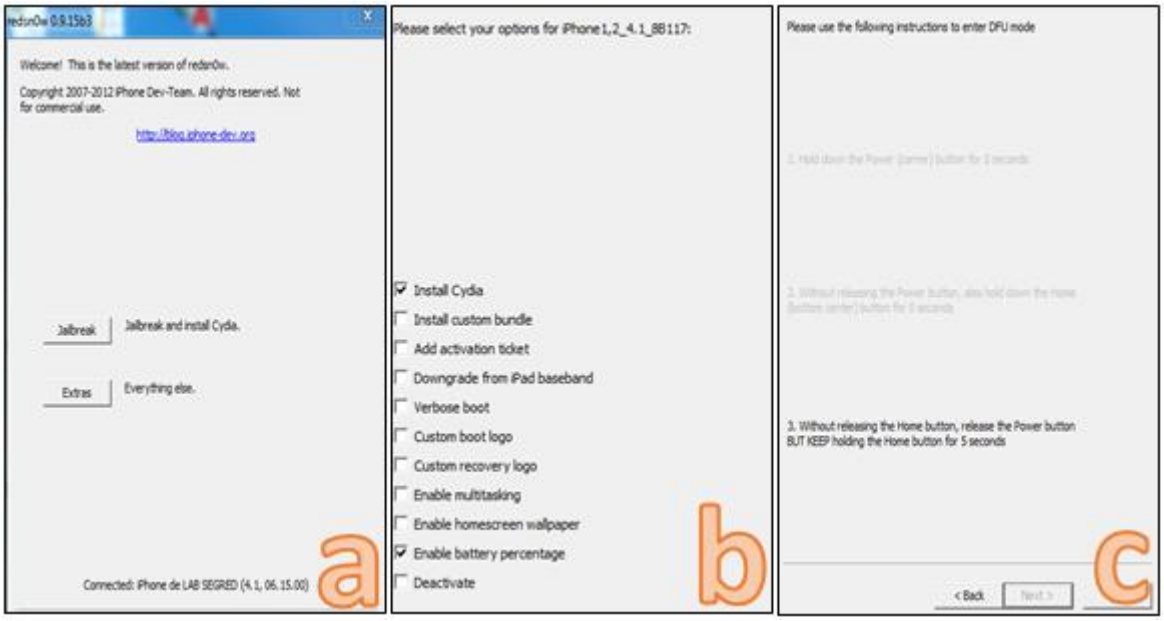


Figura 6. Uso de redsn0w para realizar Jailbreak

5. Una vez inicia el iPhone en modo DFU el programa realizará en automático el Jailbreak al dispositivo. Es importante que durante este proceso no se desconecté el iPhone, ni se apague la computadora. El Smartphone ira cambiando de pantallas durante el proceso de Jailbreak.

6. Finalmente se reiniciará el iPhone pidiendo la configuración de primer arranque, una vez terminada podemos observar que tiene el Cydia instalado, indicativo de que el proceso de Jailbreak fue exitoso. Podemos observar los últimos dos pasos en la Figura 7.

Figura 7. Jailbreak a un iPhone 3G

Jailbreak mediante p0sixspwn a un iPhone 3GS con iOS 6.1.6

1. Descargamos el programa p0sixspwn de internet. Una posibilidad puede ser el siguiente enlace. http://p0sixspwn.com/

2. Se descomprime el archivo RAR que se descargó, y se ejecuta p0sixspwn.exe con privilegios de administrador y con compatibilidad de Windows service pack 3(Desde la ventana de propiedades configuramos esto).

3. Pulsamos la opción Jailbreak (Figura 8).

4. Reiniciamos el dispositivo.

5. El programa reiniciará el dispositivo en varias ocasiones, al final tendremos el mismo resultado que con redsn0w, en la ventana aparecerá el mensaje de Done!

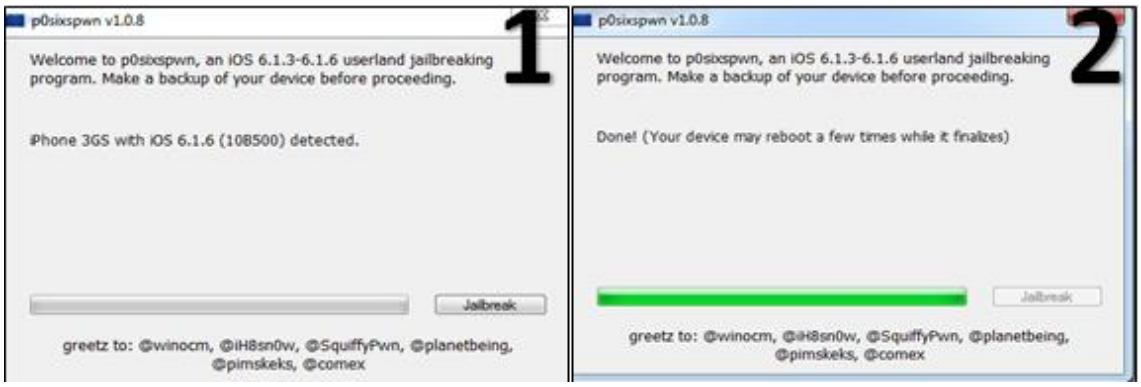


Figura 8. Jailbreak a un iPhone 3GS con p0sixpwn.

Jailbreak mediante Pangu8 a una iPad Air 2

1. Descargamos el programa Pangu8 desde la página oficial:
   Link: http://en.pangu.io/

2. La ejecutamos con privilegios de administrador.

3. Desbloqueamos el dispositivo, lo ponemos en modo avión y lo conectamos a la computadora.

4. Si reconoce el dispositivo, aparecerá en verde “Start Jailbreak”.

5. Posteriormente nos recordará realizar un respaldo del iDevice y dejar únicamente el sistema operativo sin ninguna aplicación. Seleccionamos la opción Alredy Did y Jailbreak comenzará. Tomará de 5 a 20min el proceso de Jailbreak. El proceso terminará y tendremos la aplicación Cydia instalado en nuestro dispositivo (Figura 9).

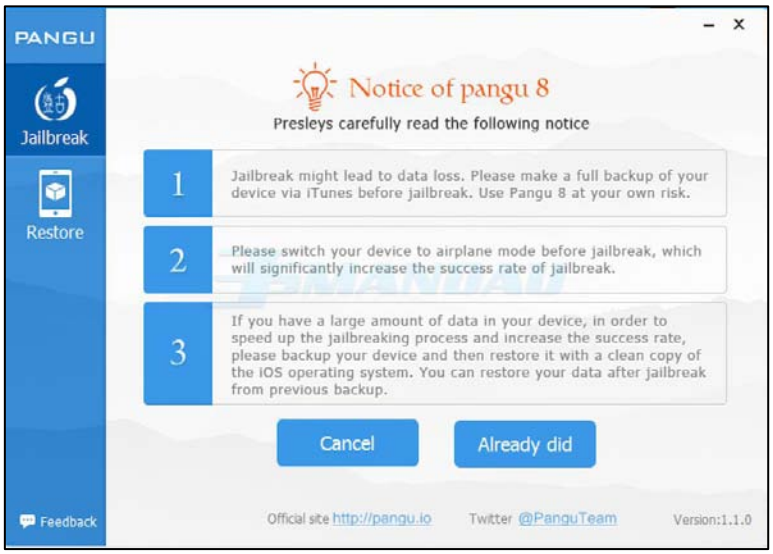


Figura 9. Jailbreak a una iPad Air 2 con pangu8.

Resultados:

La virtualización en iOS es más limitada que en Android, esto se debe a que Apple no permite conocer información adicional a la necesaria para poder desarrollar Apps. La virtualización en iOS no podría ser utilizada en un proceso de pentesting, porque no aportaría información relevante al proceso de auditoría. Solamente Xcode sería útil cuando se requiere probar el funcionamiento y rendimiento de una App desarrollada para la organización, para conocer si esta no tiene una falla de seguridad.

En cuestión del Jailbreak, resulta muy útil esta condición por diversas razones; la primera es la accesibilidad de los datos que se puede tener, de esta forma pudiera ser posible instalar malware en el dispositivo para robar información del dispositivo y como segunda razón; utilizar el mismo dispositivo para poder realizar pruebas de penetración a otros segmentos de red de manera encubierta. El Jailbreak sería útil durante las fases de explotación y post-explotación.

Información extra:

- **Repositorio de imágenes de iOS para el proceso de Jailbreak de dispositivos descontinuados.** Verificado el 14 de septiembre del 2015. http://www.iphoneros.net/ios/

- **The iPhoneWiki, enciclopedia de términos relacionados con iOS y su estado de arte.** Verificado el 14 de septiembre del 2015. https://www.theiphonewiki.com/wiki/Main_Page

Laboratorio No.5
"Acceso no autorizado. By pass en iOS"

Objetivo:

Conocer las medidas de seguridad que cuentan los dispositivos Apple para evitar el acceso no autorizado, enfocándose en algunas vulnerabilidades que estas tienen, para poder evadirlas y tener acceso no autorizado al dispositivo. Al reconocer las características de las formas de bloqueo, el alumno podrá comprender el proceso y dificultad de tratar de eludirlas.

Requisitos:

-Mac OSX 10.3 o superior
-iTunes

Material:

-Xcode
-Computadora iMac con el requisito anterior
-iPhone 4 y un respaldo del mismo.
-iPhone 3GS con iOS 6.1.3
-Una computadora con Windows
-iTunes instalado

Marco teórico:

Apple se ha caracterizado como una organización que cuida sus patentes, esto le ha sido de utilidad; porque tiene un control completo sobre el desarrollo de sus productos y el alcance en que los desarrolladores pueden llegar, esto se vuelve útil porque depende de lo que Apple desarrolle o implemente para protegerlos y no de terceros.

Apple se ha enfocado en que sus principales medidas de seguridad se encuentre en la nube, para ello su servidor centralizado iCloud permite bloquear remotamente el dispositivo y evitar que un tercero utilice el dispositivo u obtenga información sensible de su propietario original.

En cuestión del acceso local del equipo, existen métodos de bloqueo para evitar el acceso no autorizado, los principales métodos de bloqueo de pantalla son los siguientes:

1. **Código:** Es un tipo de bloqueo parecido al PIN de Android, es una cifra de 4 números que evita el acceso al dispositivo, sin embargo; es posible configurar este tipo de

bloqueo de tal forma que sea posible utilizar una contraseña más grande o incluso una contraseña con caracteres (Figura 1), mejorando la seguridad del dispositivo. En las versiones más actuales de iOS, el iDevice se bloquea después de 5 intentos fallidos por fracciones de tiempo; al fallar el intento número 10, el dispositivo queda totalmente bloqueado y es necesario autenticarse con las credenciales de Apple con las cuales fue dado de alta el dispositivo o en su defecto; hacer el mismo proceso desde iTunes.

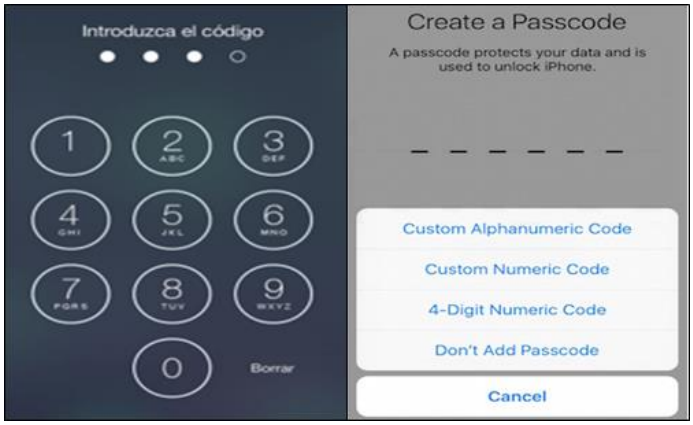


Figura 1. Bloqueo por código y opciones de cambiarlo en iOS9

2. **TouchID:** Este método tuvo su concepción a partir de la versión 8.0, consiste en guardar una copia de la huella dactilar del usuario y usarlo como contraseña; para conseguir esto, fue necesario que Apple sustituyera su botón home tradicional, por un dispositivo biométrico que mediante láser pudiera guardar una matriz de la copia de la huella dactilar (Figura 2). Apple le dio publicidad a esta tecnología al incursionarla como medio de autenticación para realizar compras y otras acciones. El bloqueo por código debe estar activado y es usado como reserva en caso de que el TouchID no funcione.

Figura 2. Bloqueo por TouchID y opciones de cambiarlo en iOS8

El problema de seguridad en iOS

La seguridad de iOS tiene vulnerabilidad que se ha hecho visible en los últimos años, como se comentó en el capítulo I, desde el mismo diseño de iOS se procuró que la seguridad fuera uno de sus pilares y adicionalmente; no estuviera al alcance de cualquier agente, para que pudiera conservarse dicha seguridad. Lamentablemente, la actualización o la creación de nuevas medidas de seguridad por parte de Apple ha sido pobre; esto ha causado que un problema de seguridad encontrado en iOS tarde mucho en ser corregido, además; este problema de seguridad afecta a todos los dispositivos que tienen esa versión y como todos los dispositivos móviles de Apple usan el mismo sistema operativo, todos se ven afectados por el mismo problema.

Un ejemplo de ello es el TouchID, su aparición es reciente (2013), sin embargo; al final de la práctica podemos encontrar un artículo sobre una consultora de seguridad que ya consiguió evadirla, con una técnica no muy complicada. Este hecho muestra un problema mayor que tienen los dispositivos Apple, mal diseño del hardware. Un problema de hardware afecta a toda la línea de ese dispositivo, sin importar que iOS sea actualizado o no; es por esta razón, que el Jailbreak persistente se ha vuelto muy común en todos los iDevices más recientes, porque en esencia, conservan el mismo hardware o el cambio solo va enfocado en cuestión de rendimiento (siempre han usado la misma familia de procesadores, por ejemplo).

A diferencia de Android que tiene un hardware muy robusto, las principales técnicas de bypass a los dispositivos Apple consisten mediante el estrés físico, estos fallos son muy complicados de corregir y se han vuelto en un problema para Apple porque versiones recientes como la 8.0 o la recién liberada 9.0 permiten que mediante estrés físico, tener acceso a los contactos y fotos del usuario con facilidad, si bien el acceso no es total, si representa en una situación preocupante porque se trata de información sensible del usuario.

Los respaldos de iTunes

iTunes es el software que nos permite sincronizar, agregar, borrar, restaurar, actualizar entre otras, con nuestros dispositivos Apple. También es utilizado para autenticar la propiedad del dispositivo y restaurarlo en caso de ser necesario. Una de las características principales de iTunes es la capacidad de hacer copias de seguridad de nuestros dispositivos, esto incluye datos personales, Apps, correos, contraseñas, entre otras; por default, estas copias de seguridad se encuentran cifradas de manera robusta, sin embargo; es posible obtener datos valiosos de estos respaldos, aunque no se consiga descifrar el respaldo, esto constituye en otra problemática de seguridad, porque una cantidad de datos importante se encuentra expuesta; sin mencionar, que si este respaldo no se encuentra cifrado deja en exposición toda la información del dispositivo.

Procedimiento:

Hacer un bypass mediante estrés físico iPhone 3G con iOS 4.1

1. Con la pantalla apaga, lo encendemos y presionamos la opción de desbloquear.

2. Seleccionamos la opción llamada de emergencia.

3. Presionamos el botón de regreso sin soltarlo y después el de encendido hasta que nos aparezca la opción de apagar (3 segundos).

4. Soltamos el botón de retroceso, cancelamos la opción de apagar y dejamos presionado el de encendido otros 3 segundos.

5. Pulsamos el botón de cancelar la llamada de emergencia e inmediatamente el botón de home.

6. Es posible que la pantalla del dispositivo se ponga negra y al final nos dé acceso al directorio y desde este punto acceder a las fotos guardas del dispositivo (Figura 3)

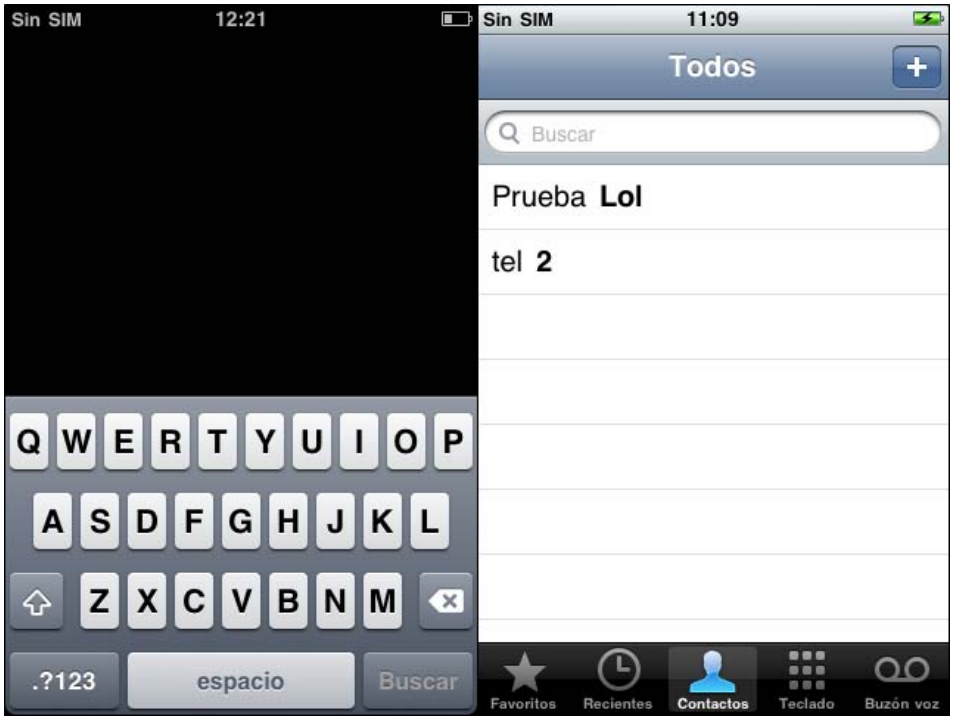


Figura 3. Bypass físico a un iPhone 3G con iOS 4.1

Obtener información sensible mediante un respaldo anterior de un iPhone 4 con cifrado

Nota: Este proceso puede realizarse de igual manera en Windows, teniendo en cuenta que la ruta de los respaldo de iTunes es: \Users\(nombredeusuario)\AppData\Roaming\Apple Computer\MobileSync\Backup\ y el tener un editor de archivos .plist.

1. Acceder a un respaldo anterior desde iMAC de la siguiente ruta: ~/Librería/Application Support/MobileSync/Backup/

2. Cada carpeta existente en esta última se trata de un respaldo de un dispositivo distinto.

3. Una vez adentro buscamos el archivo info.plist y lo abrimos con Xcode (Figura 4).

4. Como se puede observar en la figura 4 podemos obtener datos como el IMEI, teléfono y por lo tanto compañía, identificadores, versión de iOS, etc.

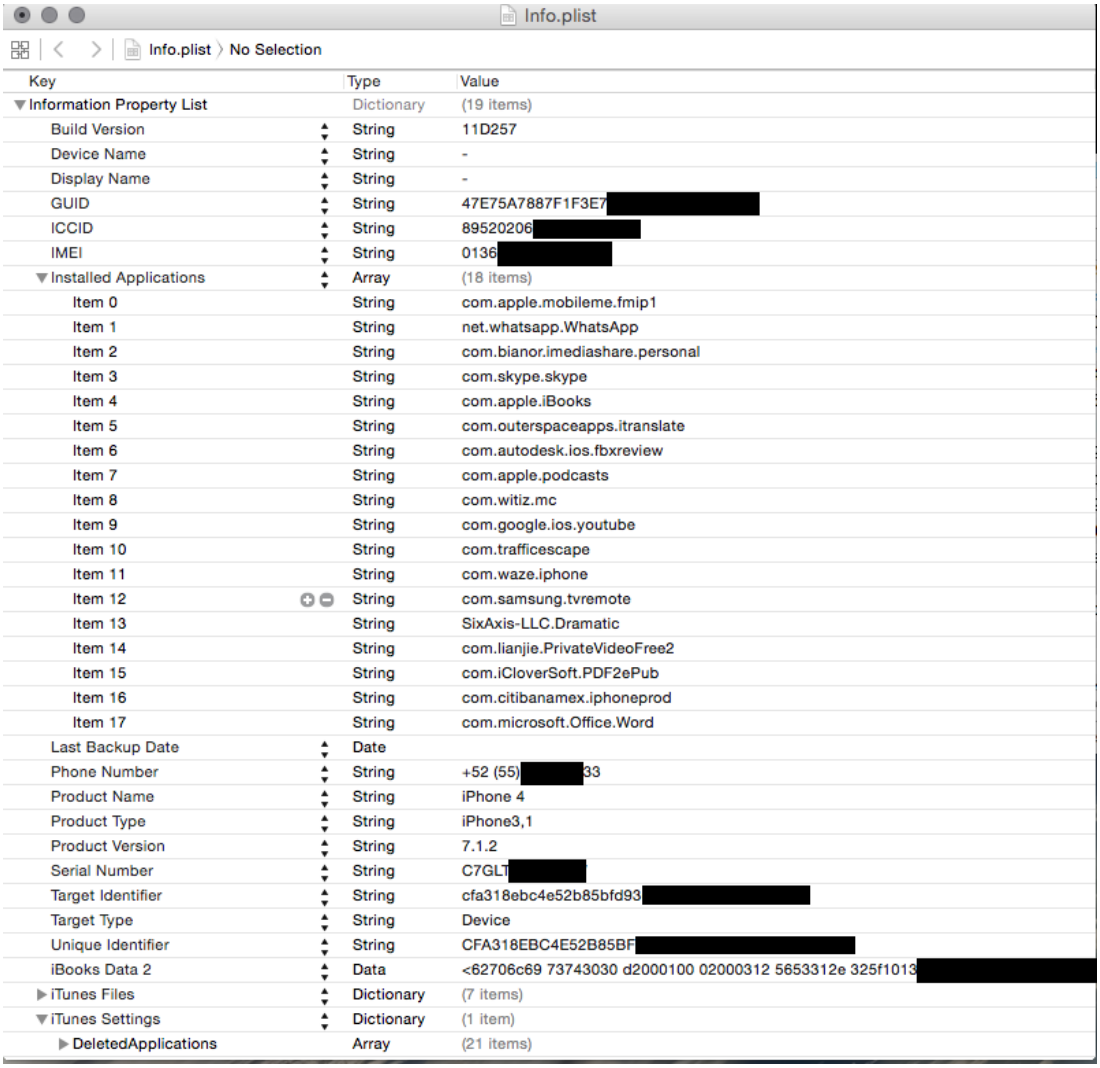


Figura 4. Obtención de datos sensibles de un respaldo de iPhone 4

Resultados:

El bypass en iOS se ha vuelto una problemática para Apple; su error de diseño de hardware ha afectado en gran manera el rendimiento que tienen sus dispositivos, sin importar que estos tengan un sistema operativo muy robusto; es por ello que se vuelve importante que Apple considere este punto para futuros desarrollos.

En cuestión de la auditoria de seguridad los bypass físicos pueden ser muy útiles en la fase de explotación, porque el adquirir información como la presentada en esta práctica pudiera mejorar ataques de ingeniería social o suplantación de identidad, estas pudieran ser el factor clave en adquirir el acceso a información sensible. El usuario constituye en el eslabón más débil en la cadena de seguridad y solo se requiere la forma de poder engañarlo.

Información extra:

- **Bypass físico a un iPhone6 con iOS 9.0.** Verificado el 21 de septiembre del 2015. https://www.youtube.com/watch?v=_giVIDKwRr4

- **Bypass a la tecnología de TouchID de Apple.** Verificado el 21 de septiembre del 2015. https://blog.lookout.com/blog/2013/09/23/why-i-hacked-Apples-touchid-and-still-think-it-is-awesome/

ANEXOS Y CONCLUSIONES

Anexo 1 Auditorías de Seguridad

A. Negociación Previa

Aspectos esenciales del contrato

1. Alcances

Esta sección tiene como fin determinar todas las acciones y la profundidad de las mismas que se realizaran durante el Pentest, además que se documentaran específicamente todas y cada una de ellas, esta sección debe contestar la preguntar ¿Qué hacer?, tomando en cuenta los siguientes aspectos:

-**Duración del proyecto (Fecha de Inicio y Término, Extra**): Definir el tiempo que durará el proyecto, tomando en cuenta su fecha de inicio y su fecha de término, cláusulas en las que se indique la forma de actuar en caso de que se invierta más tiempo en dicho proyecto. Para el prestador del servicio se le aconseja como buena práctica estipular un 20% más del tiempo del planeado.

-Actividades a realizar y la profundidad de las mismas: Esta sección aborda que actividades se realizarán y la profundidad de las mismas, además de las técnicas usadas para hacer dichas actividades, por ejemplo, si usarán pruebas de estrés a los activos, el uso de ingeniería social, etc.

-Actividades no previstas: Esta sección determina en términos generales la forma de actuar del prestador del servicio para actividades que no estén estipuladas en el contrato, pueden clasificarse de la manera mejor convenida entre las partes. La recomendación establece considerar dichas actividades para el final del proyecto, de ser posible postergarlas para el final de la prueba. Lo esencial es apegarse a los términos convenidos y posteriormente realizar lo que no se haya contemplado.

-Activos del cliente implicados: La lista de objetivos de los activos (personas, computadoras, servidores, infraestructura móvil, etc.) que forman parte de la prueba, y es importante que el cliente los defina para saber la cantidad de trabajo que será empleado para dicha tarea. Debe verificarse que los objetivos proporcionados por el cliente realmente pertenezcan a dicha organización para evitar atacar a equipos erróneamente y esto traiga como consecuencia, problemas legales; mucho más si se atacó un equipo que forma parte de un gobierno o un organismo público; esto puede hacerse usando el comando whois en cualquier distribución de Linux.

2. **Metas**

Deben considerarse las metas que pretende cumplir dicho proyecto tomando en cuenta un sistema jerarquizado, con el fin de que el plan de trabajo del prestador del servicio se enfoque en las metas primarias y posteriormente en las secundarias, así sucesivamente hasta cubrir todas las metas acordadas. Debe considerarse un análisis de negocio en la que se establezca el grado de seguridad que tiene el cliente previo a la realización de la prueba.

3. **Contacto entre el cliente y el prestador de servicios**

Esta parte establece tener una relación de contacto entre el cliente y el prestador del servicio en caso de que surjan inconvenientes, deben tener al menos un contacto de emergencia que tenga entre los principales datos: nombre completo, responsabilidad dentro de la organización, que tenga autorización de tomar decisiones respecto a la forma de llevar la prueba, al menos dos formas de comunicarse con él de manera permanente, una manera segura de intercambio de información.

Además debe considerarse dentro de los contactos: todo el equipo del prestador del servicio, el líder del grupo, dos contactos de personas que puedan proporcionar datos técnicos de cada área que formará parte de la prueba, dos contactos de los terceros, un contacto con un responsable gerencial de la organización.

Además se pude establecer con el cliente reuniones periódicas en las que se vallan presentando los avances, de ser usada la transferencia de información electrónica es indispensable que se encuentre cifrada.

4. **Reglas de negocio**

Esta sección debe contestar la pregunta ¿Cómo hacer?, consiste en las consideraciones de cómo se realizarán las actividades previamente seleccionadas tomando en cuenta las siguientes reglas básicas:

- **Línea de Tiempo**: Mediante el diagrama de GANTT u otra técnica, tener una resolución de cómo se empleará el tiempo. Además de tomar en cuenta bajo que horarios y condiciones podrán ejecutarse las pruebas.

- **Localización de los activos:** Tener una relación de la localización física de los activos y otras consideraciones geográficas de la organización, la infraestructura móvil o la fija, más si se piensa poner bajo prueba a diversas sucursales de la organización.

- **Uso de la información sensitiva:** Establecer la forma en la que se actuará con la información sensitiva obtenida durante la prueba.

- **Profundidad en la obtención de la información:** Establecer la profundidad implicada en la obtención de información, o por ejemplo, si una vez comprometido un sistema el grado de empeño que se pondrá en obtener información, escalar privilegios, etc.

- **La forma en que actuará el cliente:** Establece la forma en que actuará al cliente ante las pruebas, si aplicará técnicas de evasión, alarmas, procedimientos, etc.

- **La autorización de las pruebas**: Debe establecerse de manera explícita todas las pruebas que se realizarán en la prueba y que estas sean conocidas y aprobadas por el cliente.

- **Consideraciones legales:** Todas las leyes implicadas que puedan formar parte de la prueba, entre las más importantes están las de los terceros, leyes nacionales, extranjeras e internacionales y cómo afectan estas el desarrollo de la prueba.

5. **Términos de pago**

Debe considerarse explícitamente la forma de pago del cliente al proveedor del servicio, los horarios, calendarios, tarifas, sanciones e incluso la tarifa usada en caso de realizar trabajo adicional al estipulado en el contrato.

6. **Glosarios y anexos**

Deben establecerse todos los glosarios y anexos que ayuden al cliente a comprender mejor la forma en la que se llevará a cabo la prueba por parte del prestador del servicio.

Técnicas para la obtención de la información

Se deben formular cuestionarios, que ayuden a los testers a identificar todas las necesidades que el cliente quiere cubrir, y a su vez formular cuestionarios que le permitan como alcanzar los objetivos en las distintas áreas que tienen que ver con la prueba. Entre las principales están:

-Cuestionario hacia la penetración de la red.
-Cuestionario hacia la penetración de servicios web.
-Cuestionario hacia la penetración de las redes inalámbricas.
-Cuestionario hacia la penetración física.
-Ingeniería Social.

-Gerentes o Directores de los negocios.
-Administradores de Sistema

Pueden aplicarse técnicas como Delphi entre otras, la finalidad de aplicar dichas técnicas, es que el proveedor de servicios pueda obtener la mayor cantidad de información útil; que le ayudara a la planeación de las pruebas que realizará al cliente.

B. Recopilación de Información

Open Source Intelligence (OSINT)

OSINT es un método para obtener información útil mediante los recursos y servicios públicos, este define 3 niveles de interacción:

Pasiva: Es la búsqueda de información en la que el objetivo no se entera de la actividad, básicamente consiste en la consulta de información archivada o almacenada que ofrece la misma organización o por terceros. El riesgo de este nivel de interacción es obtener información errónea o que no sea útil.

Semi-Pasiva: Consiste en obtener información por métodos que parecen ser comportamientos normales. Información de archivos o servicios que no tienen restricción, solicitar informes sobre servicios y comportamientos generales de la organización entre otras.

Activa: Es el uso de métodos que proporcionen información sin importar que el objetivo se percate o no de ello. Escaneo de puertos, identificación de servicios, sistemas operativos, etc. forman parte de este nivel de interacción.

OSINT divide la búsqueda de información por sectores, los principales son:

Organización

Esta parte identifica información a nivel organización no se centra en personas sino en características generales de la organización como pueden ser:

Física: Direcciones físicas, seguridad física en sus instalaciones, documentos de sus propiedades, infraestructura, sus relaciones sociales y corporativas, identificar a sus clientes, sus proveedores, forman parte de este rubro. La meta es identificar los servicios críticos de la organización e información que pueda ser usada en ingeniería social.

Lógica: Esta parte comprende tener información útil sobre los socios, clientes, competidores como pueden ser nombres, direcciones, tipo de relación con el objetivo, información financiera, información general de su infraestructura lógica, etc. Tener un esquema con las relaciones sociales entre los miembros de la organización, el sector en el que se desempeña, su línea de productos, estrategias de mercado, fechas importantes para la organización, ofertas de empleo que ofrecen, afiliaciones ya sea en al ámbito caritativo o político, datos legales, RFQs, RFP, organigrama, identificación de personal clave, certificaciones y licencias están incluidos en este rubro.

Electrónica: Información de metadatos, mercadeo como pueden ser campañas pasadas, estilos, colores, etc., bloques o direcciones IP que pertenecen a la organización, direcciones de correo electrónico, infraestructura móvil, infraestructura externa, tecnologías que usa, accesos remotos, tecnologías y métodos de seguridad de la organización.

Financiera: Consultar sus reportes, análisis de mercado, su capital, etc. En caso de las organizaciones en Estados Unidos, el uso de la base de datos EDGAR, si es en México y cotiza en bolsa de valores, sus reportes financieros que la misma organización pública.

Individual

Esta información se recopila a nivel empleado, teniendo como prioridad los empleados que cuentan con privilegios dentro de la organización o que su rol es crítico. Las áreas a considerar son:

Historia: En este sentido entra su historial legal, afiliación política, grado de estudios, certificaciones, etc.

Localización física: identificar su lugar de labores, horarios, etc.

Recursos Informáticos: Su correo electrónico, redes sociales, recursos que utiliza, teléfono, tipo de usuario, si usa dispositivo móvil, etc.

C. Consideraciones específicas de las fases de Explotación y Post-Explotación

Reglas del Negocio

Protegiendo al Cliente:

-No se deben modificar los servicios que el cliente considere “Esenciales”, a no ser que haya acordado previamente.

-Todas la modificaciones hechas a los sistemas del cliente deben ser bien documentadas y tener un control exacto sobre las máquinas que fueron restablecidas de las que no.
-Todas las acciones realizadas en las máquinas comprometidas deben tener seguimiento e incluirse en los reportes.

-El uso de datos sensitivos como contraseñas para obtener mayores privilegios o su uso para la prueba de penetración deben tener un seguimiento y solo usarse si previamente se acordó con el cliente el uso de las mismas.

-Las contraseñas no deben ser incluidas en el reporte final.

-Todos los métodos usados por el prestador de servicios que puedan afectar el desempeño del equipo del cliente deben ser desactivados en cuanto sea posible.

-Toda la información obtenida durante la prueba de penetración debe ser cifrada.

-Toda información sensitiva usada en los reportes debe ser estrictamente controlada.

-Toda la información debe ser destruida una vez que el cliente acepta el reporte final.

-Toda la información que se presenta al cliente debe cumplir con todas las leyes aplicables.

-No se deben comprometer datos del cliente que estén asociados con servicios provistos por terceros.

-Toda la información debe ser salvada y con un hash de numeración.

-Las bitácoras no deben ser destruidas o modificadas sin autorización previa del cliente.

Protegiendo al Prestador de Servicios:

-Asegurarse que todas las acciones realizadas estén documentadas y firmadas por el cliente.

-Tener una copia de las políticas de uso del cliente.

-Confirmar que el cliente use las regulaciones gubernamentales sobre el uso que se le da a la información.

-Cifrar toda la información almacenada en dispositivos de almacenamiento externos.

-Negociar con el cliente todas las pruebas en las que estén involucrados la participación de servicios provistos por terceros.

-Verificar las leyes concernientes al uso y almacenamiento de audio y video.

Análisis de la Infraestructura

Configuración de Red

-Interfaces: IP, Máscaras y Puertas de Enlace.
-Ruteo: Interfaces, Tablas de Ruteo (Estáticas y Dinámicas), Tablas ARP.
-Servidores DNS: Las bases de datos DNS, posibilidad de pivoteo.
-Solicitudes DNS en caché: Identificar credenciales, información reciente que puede servir como pivote para obtener mayor cantidad de información.
-Servidores Proxy: Información útil sobre el tráfico de la red, posibilidad de usarlos como pivote.
-Solicitudes ARP: Información útil sobre las máquinas que interactúan con la máquina comprometida.

Servicios de Red

-Escucha de Servicios: Descubrir e identificar los servicios que no fueron localizados durante el escaneo inicial, sobre todo aquellos que puedan contener información sensitiva.
-Conexiones VPN: Identificar las conexiones y las sesiones VPN para poder atacar otras partes de la red con el fin de identificar información sensitiva.
-Directorios de Programas: Identificar la información que sea útil como pueden ser: cuentas de usuario, servicios e información útil para realizar ataques de ingeniería social.
-Vecinos: El uso de protocolos de descubrimiento como son el CDP o LLPD que pueda ayudar a identificar la red interna.

Programas Instalados:

-Herramientas de inicio: Identificar las aplicaciones que inician con el sistema ayuda a identificar el propósito del sistema y las medidas de seguridad con las que cuenta la red.
-Herramientas instaladas: Las aplicaciones que son usadas en el sistema ayuda a identificar el propósito del sistema y las medidas de seguridad con las que cuenta la red.
-Servicios de Seguridad: Identificar servicios de seguridad que cuenta la red, IDS/IPS, HIDS/ HIPS, antivirus y firewalls, etc.
-Recursos compartidos: Obtener datos como pueden ser archivos compartidos, listas de control de accesos, permisos, usuarios, información sensitiva como contraseñas, nombres de usuario, etc. La capacidad de subir y compartir archivos maliciosos.

Servidores de Bases de Datos:

-Nombres de las Bases de Datos: Identificar los nombres de las bases de datos ayudan a saber qué tipo de información contienen, información útil para priorizar los objetivos.
-Tablas: Los nombres de la tabla, metadatos, comentarios, campos y tipos, los nombres de las columnas.
-Usuarios, contraseñas, grupos, roles, permisos.

Directorios de Servidores:

-Comprometer un servidor para tener acceso a todos los usuarios a los que ofrece dicho servicio.
-Extraer la lista de objetos (usuarios, contraseñas, máquinas, etc.)
-Nombres: Esta información permite priorizar ataques.
-Servicios Implementados: Permite obtener información de las respuestas no atendidas, los permisos de los archivos, actualizaciones, aplicaciones y versiones.
-Instalación de puertas traseras o modificación de procesos que hagan vulnerable al servicio.

Certificados:

-Identificar firmas digitales, certificados de cifrado.
-Control del Servicio de Certificados: Creación de nuevos certificados, revocación de certificados, modificación de la lista de revocación.

Administración de código fuente:

-Encontrar los proyectos de software.
-Verificar acceso y modificación a los códigos fuente del software que usa la compañía.
-Identificar todos los desarrolladores.
-Identificar y documentar las configuraciones.

Servidor DHCP:

-Identificación de las concesiones otorgadas.
-Identificación de las configuraciones y opciones.
-Posibilidad de modificar los servicios.

Virtualización:

-Identificación de máquinas virtuales (Nombre, Configuraciones, OS)
-Identificación de contraseñas y certificados digitales de los administradores.

-Identificación de la configuración del software virtual.
-Identificación de la configuración de los hosts.
-Posibilidad de tener acceso a la información guardada en máquinas virtuales y de modificar o cancelar los servicios.
-Analizar el riesgo de la posibilidad de un ataque de denegación de servicio.

Mensajería:

-Identificación los directorios de servicios.
-Comprometer las credenciales de usuarios.
-Acceso a información confidencial.
-Identificación de los hosts de la red.
-Identificar las relaciones de negocios entre usuarios y sistemas.

Administración y monitoreo:

-Identificación protocolos y servicios de administración como puede ser: SNMP y los Syslog.
-Identificación de servicios como: SSH, Telnet, RDP, Terminales, Software de administración virtual.

Sistemas de respaldo:

-Identificación de los host y de los sistemas.
-Identificación de los servidores.
-Credenciales de acceso.
-Acceso a la información de respaldo.

Servicios en Red (RADIUS, TACACS, etc.)

-Identificación de usuarios, host y sistemas.
-Credenciales de acceso.
-Identificar los riesgos de causar una denegación de servicio.

Manejo de la Información Sensitiva:

-Key-logging.
-Capturas de pantalla.
-Captura de tráfico de red.

Manejo de usuario:

-En el sistema (Historial de archivos, llaves de cifrado, archivos de oficina, configuraciones, medios de almacenamiento, etc.)
-En los navegadores (Historial de navegación, favoritos, historial de descargas, credenciales, proxies, plugins, extensiones, etc.)
-Mensajería Instantánea.

Configuraciones de los sistemas:

-Políticas de Seguridad.

Objetivos de alto valor:

-Completar la información de los objetivos considerados de alto valor.

Ex filtración:

-Mapeo de todas las rutas a las que se puede aplicar ex filtración.
-Pruebas a rutas de ex filtración.
-Medición de la seguridad de los controles.

Persistencia:

-Instalación de una puerta trasera que requiera autenticación.
-Uso de contraseñas, certificados y canales seguros para la conexión con la puerta trasera.

Pivoteo en el sistema comprometido:

-Subir herramientas.
-Uso de herramientas locales.
-Escaneo ARP.
-Enumeración DNS.
-Ataques de fuerza bruta.
-Ejecución de exploits remotos.
-Obtención de credenciales y accesos.

Pivoteo a través del sistema comprometido:

-Port Forwarding
-Proxy con la red interna.

-VPN con la red interna.
-Ejecución de exploits remotos.
-Abuso de las credenciales de acceso comprometidas.

Limpieza:

-Remover todos los ejecutables, scripts y archivos temporales.
-Regresar los valores originales de las configuraciones modificadas.
-Remover puertas traseras, usuarios creados, etc.

Infraestructura móvil:

-Presencia de redes con dispositivos móviles.
-Programas instalados, versiones y modelos.
-Tipo de información que manejan los mismos.
-Roll de los usuarios que manejan los dispositivos.

D. Consideraciones específicas de la fase Reportes

Reporte Ejecutivo:

Esta sección abarca los objetivos específicos de la prueba de penetración negocios.

Antecedentes:

Esta sección aborda todos los objetivos de la prueba. Detalla los términos que se relacionan con la negociación previa, contramedidas, riesgos y las metas de prueba.

Se deben archivar e incluir todos los intentos por comprometer un sistema y deben abordar, la efectividad de la acción, además de identificar como dicha acción tiene relación con las metas definidas en la negociación previa.

Clasificación del Riesgo:

Mediante la información provista durante la negociación previa debe crearse la forma en la que se clasificará el riesgo, pueden hacerse de diversas formas e incluso seguir algunos que han sido aprobados como son FAIR, DREAD. También se deben tomar en cuenta la frecuencia y el impacto que tendrían dichas amenazas.

Hallazgos generales:

Esta sección tiene como fin presentar un cuadro básico y estadístico sobre los problemas más comunes que se presentaron durante la prueba. El uso de gráficos para ilustrar los objetivos penetrados, resultados de la prueba, procesos, escenarios de ataque, ataques efectivos y otras consideraciones previstas en la negociación previa.

Recomendaciones:

Esta sección establece las recomendaciones que ayudarán a la reducción general del riesgo, de acuerdo a la clasificación previamente hecha y que será de utilidad para la elaboración del plan estratégico.

Plan Estratégico:

Este plan puede incluir una metodología para resolver los problemas de inseguridad encontrados acorde con el riesgo que estos implican, con el fin de erradicarlos usando un orden, otra recomendación es bajo el sistema de tiempo (corto, mediando, largo plazo).

Reporte Técnico:

Este tipo de reporte aborda los detalles técnicos de la prueba y todos los componentes que están relacionados con la negociación previa, este reporte describe los alcances, información, vectores de ataque, impactos y soluciones de la prueba.

Introducción:

-Personal involucrado durante la prueba por ambas partes
-Información de contacto
-Vienes envueltos en la prueba.
-Objetivos
-Alcances
-Profundidad
-Enfoques
-Clasificación y estructura de las amenazas

Recopilación de la información:

Deben presentar toda la información obtenida durante la fase de recopilación de información, debe estar enfocada al medio de la empresa, se sugiere hacerlo usando 4 categorías:

Inteligencia Pasiva: Es donde el prestador de servicio busca información pública de los objetivos, es decir; información que no requiere de comprometer a un sistema, buscando información de la organización en Internet y en otros medios externos de la organización.

Inteligencia Activa: Es donde el prestador de servicio busca información mediante escaneo de puertos, un mapeo de red, cualquier acción en la que haya un contacto con la red del cliente.

Inteligencia Corporativa: Información de la estructura de la organización, unidades y procesos de negocio, etc.

Inteligencia del Personal: Información de los empleados que tenga relación con la organización como suelen ser departamentos, correos postales y electrónicos, organigramas, etc.

Análisis de Vulnerabilidades:

Se identifican las vulnerabilidades y amenazas que ponen en riesgo la empresa, debe ser cuidadosamente clasificada. Un ejemplo podría ser el siguiente:

-Niveles de clasificación de las vulnerabilidades.

-Vulnerabilidades técnicas (Capas del modelo OSI, identificadas por scanner, identificadas manualmente)

-Vulnerabilidades lógicas (Vulnerabilidades que no pertenecen a las capas del modelo, tipos, lugar de procedencia)

-Resultados.

Explotación:

Debe explicarse cómo se confirmaron las vulnerabilidades identificadas mediante la explotación, los accesos y privilegios ganados a los objetivos de la organización (cliente).
Los requerimientos mínimos que deben incluir son los siguientes:

-Línea de tiempo de la fase de explotación.

-Objetivos seleccionados.

-Actividades de la fase

-Ataques Directos:

-Objetivos disponibles y no disponibles durante la explotación.
-Información individual del host.
-Ataques realizados
-Ataques satisfactorios
-Nivel de acceso ganado y obtención de privilegios
-Solución (Referencias, técnicas, sugerencias)

-Ataques Indirectos:

-Phishing (Línea de tiempo, objetivos, efectividad, nivel de acceso ganado)
-Lado del Cliente (Línea de tiempo, objetivos, efectividad, nivel de acceso ganado)
-Navegadores (Línea de tiempo, objetivos, efectividad, nivel de acceso ganado)

Post-Explotación:

Se deben detallar las acciones realizadas una vez que se ha logrado comprometer equipos del cliente para la obtención de mayor cantidad de información y la capacidad de comprometer más objetivos del cliente una vez que se logró tener acceso a un equipo de la organización. Se recomienda considerar la siguiente información:

-Obtención de privilegios (Técnicas y métodos usados)
-Adquisición de información crítica.
-Valor de la información.
-Acceso a los sistemas de negocios.
-Acceso a los sistemas con datos protegidos.
-Persistencia
-Ex filtración
-Efectividad de las contramedidas.

Exposición/Riesgo:

Esta parte del reporte consiste en crear una evaluación del riesgo, una vez concluido todas las fases anteriores, con el fin de hacer un análisis completo sobre todo lo implicado, con el fin de advertir al cliente de todas las problemáticas que cuenta su organización. Debe cubrir las siguientes secciones:

-Evaluar la frecuencia de los incidentes.

-Estimación de la capacidad de las amenazas.

-Estimar la seguridad de los controles.

-Compendio de las vulnerabilidades (nivel de habilidades y nivel de acceso requerido).

-Estimación de las pérdidas por incidente (primario, secundario, etc.).

-Riesgos derivados (amenazas, vulnerabilidades).

-Remover todo el software malicioso y las acciones realizadas.

E. Guías técnicas especializadas para el Pentesting

El uso de buenas técnicas y metodologías durante el Pentesting, resulta esencial para alcanzar el objetivo de dicho proyecto, consientes de esto, el grupo desarrollador del estándar considerado en este documento, ofrece un conjunto de guías técnicas útiles para todas las fases del Pentesting, estas incluyen: referencias, comentarios, consideraciones, sugerencias de software (gratuito y comercial) que ayudan a la correcta ejecución de los procedimientos necesarios.

Esta guía incluye a grandes rasgos:

Fase	Descripción
Herramientas Básicas	Sistemas operativos, software de virtualización, escáneres, herramientas físicas, software básico para el Pentesting.
Recopilación de la Información	Información de OSINT, información corporativa de Estados Unidos, consideraciones avanzadas para la creación de perfiles, sitios frecuentes visitados por los usuarios, software, comandos para la obtención de información, etc.
Análisis de vulnerabilidades	Ofrece un conjunto de herramientas automatizadas para este fin, escáneres web, analizadores de tráfico, validadores de vulnerabilidades y propuestas de vectores de ataque.
Explotación	Presenta herramientas, técnicas, consideraciones de tecnologías, productos comunes, estándares, guías y comandos que ayudan a realizar pruebas de explotación de la manera más amplia y precisa posible.
Post Explotación	Consideraciones, comandos básicos y sugerencias que son propias de la fase de post explotación.
Reportes	Consideraciones generales para la elaboración de reportes.

Sitio Web de las guías técnicas de PTES:

http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines
Última visita realizada el 15 de Agosto de 2015. Penetration Testing Execution Standard 2012.

Anexo 2 Virtualización

La virtualización es crear mediante software, un entorno lo más acercado posible a la realidad, de un recurso computacional que físicamente no poseemos; este puede tratarse:

-Sistema Operativo
-Hardware
-Recursos de Red y Locales

La virtualización es una opción recomendada cuando es requerido el realizar pruebas o disponer de servicios que no tenemos físicamente en nuestra infraestructura. En cuestión de la realización de las pruebas, es conveniente; porque podemos obtener resultados más cercanos a la realidad, que cuando se usa un simulador; pero a su vez no se pueden realizar en el entorno sobre el cual van a trabajar por cuestiones de rendimiento o por seguridad o cuando la infraestructura ya se encuentra operando en la organización y se corre el riesgo por realizar dichas pruebas. Las ventajas que podemos obtener de la virtualización son las siguientes:

-Se reutiliza hardware. Costo
-Rápida implementación. Tiempo
-Administración centralizada. Tiempo
-Mejora procesos de clonación y copia. Costo y Tiempo
-Entorno controlado. Seguridad
-Ahorro de energía. Costo

Por otra parte, si no se tiene el conocimiento para implementar esta tecnología o no es requerida para el fin que queremos alcanzar, pudiera tenerse las siguientes desventajas:

-Mayor conocimiento técnico. Costo
-Reducción en los recursos disponibles. Costo
-El anfitrión se vuelve crítico. Seguridad.
-Reducción en las posibilidades de implementar seguridad. Seguridad
-Limitación de algunas funciones. Rendimiento

En cuestión de pruebas de seguridad, la virtualización resulta en algo útil, porque podemos replicar cualquier ataque de una manera similar a como se realizaría en el entorno objetivo. Como vimos en el capítulo I, durante la fase de análisis de vulnerabilidades, la

virtualización resulta en una herramienta apropiada para comprobar la efectividad del modelado de amenazas que se plantea con anterioridad y de esta manera incrementar la probabilidad de tener éxito durante la fase de explotación y post-explotación.

En lo que respecta a virtualización en Android e iOS para realizar pruebas de seguridad, podemos realizar una serie de pruebas que servirían como inicio para el desarrollo y aprendizaje en esta área, además; podemos tener un desarrollo activo sin tener físicamente los smartphones o en ese momento se encuentran fuera de nuestro alcance. Como se puede apreciar en las siguientes tablas, la cantidad de pruebas que podemos realizar, utilizando la virtualización de estos sistemas operativos:

Android		
Tipo de Ataque	**Virtual**	**Físico**
Lock Screen Bypass	✓	✓
Rooting	✓	✓
Ingeniería Social	✓	✓
Robo de Información	✓	✓
Hacking Bluetooth		✓
Hacking GPS		✓
Smartphone Pentest Framework		✓

iOS		
Tipo de Ataque	**Virtual**	**Físico**
Lock Screen Bypass		✓
Jailbreak		✓
Ingeniería Social	✓	✓
Robo de Información	✓	✓
Hacking Bluetooth		✓
Hacking GPS		✓
Smartphone Pentest Framework		✓

La cantidad de pruebas posibles para realizar en iOS es menor en Android, la razón es que Apple, se ha encargado de desarrollar su hardware y software de manera cerrada, esto dificulta el desarrollo de pruebas y laboratorios de seguridad; ya que es necesario contar con hardware y software de Apple para poder desarrollar dichos laboratorios, sin embargo; una vez superando esta restricción, es posible alcanzar un control total del dispositivo y realizar cualquier tipo de prueba que se desee.

Conclusiones

Este proyecto resultó personalmente para mí en una experiencia muy enriquecedora, porque logré comprobar en viva voz el resultado de uno de los refranes de Confucio, "Me lo contaron y lo olvidé; lo vi y lo entendí; lo hice y lo aprendí". En este proyecto conseguí aprender a realizar diversas pruebas de seguridad con los smartphones y el saber sus implicaciones; por ejemplo en el ámbito de usuario común, todos sabemos que procesos como el Jailbreak o el Rooting son comunes y que en la actualidad son fáciles de realizar. Pero resulta muy importante que un profesional en computación y sobre todo en seguridad informática comprenda en un grado aceptable las características y las implicaciones que tienen estos procesos tanto en los dispositivos como en su impacto social. En el ámbito profesional el saber este tipo de pruebas, pueden incrementar la eficacia al realizar actividades como el desarrollo de medidas de seguridad y el proceso de auditorías de seguridad, ambas actividades son comunes para cualquier ingeniero que trabaje en la seguridad informática.

El propósito de este proyecto es el crear una guía de inicio para los ingenieros de computación en el mundo de los dispositivos móviles, porque si bien no es una temática que se imparta como una materia común en la facultad, si resulta importante el tener al menos, una idea de dónde empezar, porque los dispositivos móviles están presentes en muchas tareas que se realizan en la sociedad, donde al salir de la escuela se tendrán que considerar como prioridad en el trabajo profesional. A principios del año en curso, se presentó un curso con lo expuesto en esta tesis y en general tuvo un impacto muy bueno en las personas que lo tomaron, ellas adquirieron un criterio más amplio y completo sobre lo que ya tenían presentes ellos mismos, esa era la idea, el enriquecer el conocimiento. En la actualidad pueden considerar los dispositivos móviles como un área de estudio importante y actual, de esta manera creamos conciencia en los ingenieros mexicanos de lo primordial que resulta estar familiarizado con las tecnologías del presente y del futuro.

Personalmente me llevo un mar de conocimiento en comparación del granito de arena que se pretende aportar. Se espera que este proyecto continúe creciendo y permita a muchos más estudiantes de la facultad, adquieran las habilidades necesarias para el día del mañana en esta área del conocimiento, por esa razón se enfocó en sentar muy buenas bases teóricas y las referencias para aprender más; si bien las prácticas no son del todo las más recientes, son muy útiles para comprobar lo aprendido en la teoría y fáciles de replicar; el motivo nunca fue enseñar a realizar tutoriales de hackeo, porque de éstos existen muchos en You Tube y otras fuentes, sino en el hecho de comprender y saber porque suceden los mismos, el saber esto nos permite crear vanguardia sobre este tipo de actividades, cualidad que todo buen ingeniero deberá poseer para alcanzar ese título.

Glosario

- **Amenaza**: todo elemento, acción o persona capaz de atentar contra la seguridad y el buen funcionamiento de un sistema.
- **API**: en español Interfaz de Programación de Aplicaciones o API (por sus siglas en inglés), es un conjunto de subrutinas, funciones y procedimientos (métodos en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software.
- **App**: en términos generales es cualquier programa utilizado en una computadora, pero en este documento se enfoca a los utilizados por los dispositivos móviles.
- **ARM**: es una arquitectura RISC (Ordenador con Conjunto Reducido de Instrucciones, RISC por sus siglas en inglés) desarrollada para aplicaciones de baja potencia o con pocos recursos computacionales, este tipo de arquitectura está presente en los dispositivos móviles.
- **Códec**: es la abreviatura de codificador-decodificador. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos (stream) o una señal.
- **CPU:** Unidad Central de Proceso o CPU(por sus siglas en inglés) es el hardware dentro de una computadora u otros dispositivos programables, que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema. Esta tarea la realiza el microprocesador de la computadora.
- **Demonio**: demonio (UNIX y Linux), servicio (Windows) o programa residente (MS-DOS) es un tipo especial de proceso informático no interactivo, se ejecuta en segundo plano en vez de ser controlado directamente por el usuario.
- **Desbordamiento**: es un error de software que se produce cuando un programa no controla adecuadamente la cantidad de datos que se copian sobre un área de memoria reservada para ese propósito.
- **DNS:** Sistema de Nombres de Dominio o DNS (por sus siglas en inglés) es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada. Su función más importante, es traducir (resolver) nombres comprensibles para las personas en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos.
- **Driver:** controlador de dispositivo o driver en inglés, es un programa informático que permite al sistema operativo utilizar hardware adicional (periférico) para una mejor experiencia en el uso de una computadora.
- **Exploit**: es un fragmento de software, fragmento de datos o secuencia de comandos y/o acciones, utilizada con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo, como puede ser el acceso no autorizado.

- **Framework**: es una estructura conceptual y tecnológica de soporte definido, incluye módulos concretos de software, que sirven de base para la organización y desarrollo de software, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas.
- **Fuzzing:** se trata de un conjunto de pruebas semiautomáticas o automáticas donde se proveen datos al azar, inválidos o no esperados en la sección de entrada de datos del software. De esa forma es posible comprobar la seguridad en lo que respecta a la validación de datos.
- **GPU:** es un coprocesador dedicado al procesamiento de gráficos, utilizado para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas.
- **IDE**: Entorno de Desarrollo Interactivo o IDE (por sus siglas en inglés), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software. Las principales herramientas son: un editor de código fuente, herramientas de construcción automáticas y un depurador de código.
- **ISP:** Proveedor de Servicios de Internet o ISP (por sus siglas en inglés), es la empresa que brinda conexión a Internet a sus clientes.
- **Jailbreak**: es el proceso propio en los dispositivos Apple de anular o evadir los mecanismos de seguridad impuestos, con la finalidad de utilizar los recursos de diversas formas.
- **Kernel**: es un software que constituye en la parte fundamental del sistema operativo, y se define como la parte que se ejecuta en modo privilegiado.
- **Librería**: o biblioteca, es un conjunto de funciones, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.
- **POSIX:** es una norma escrita por la IEEE. Dicha norma define una interfaz estándar del sistema operativo y el entorno, incluyendo un intérprete de comandos (shell), y programas de utilidades comunes para apoyar la portabilidad de las aplicaciones a nivel de código fuente.
- **Proceso**: es un programa que se encuentra en ejecución en un determinado momento.
- **Renderizado:** proceso de generar una imagen o vídeo mediante el cálculo de iluminación partiendo de un modelo en 3D.
- **Root**: es el nombre convencional de la cuenta de usuario que posee todos los derechos en todos los modos. Normalmente esta es la cuenta de administrador, porque tiene la capacidad de realizar cualquier cambio en el sistema.
- **Rooting**: es el proceso que permite obtener control privilegiado (root) en dispositivos Android.

- **Sandbox**: o caja de arena, es un mecanismo para ejecutar programas con seguridad y de manera separada. A menudo se utiliza para ejecutar código nuevo, o software de dudosa confiabilidad proveniente de terceros.
- **SDK**: Kit de Desarrollo de Software o SDK (por sus siglas en inglés) es un conjunto de herramientas de desarrollo de software que le permite al programador o desarrollador de software crear aplicaciones para un sistema concreto.
- **Smartphone**: o teléfono inteligente, es un teléfono celular de mayor capacidad construido para un sistema operativo móvil, con la finalidad de proveer servicios como la flexibilidad y portabilidad.
- **Socket**: es una conexión del tipo abstracta que utiliza una dirección IP y un puerto de capa 4 que permite a 2 agentes establecer una comunicación e intercambiar datos.
- **VPN:** Red Privada Virtual o VPN (por sus siglas en inglés) es una tecnología de red que permite crear y utilizar conexiones seguras (cifradas) entre dos agentes para intercambiar datos de manera muy particular entre estos agentes.
- **Vulnerabilidad**: son características propias de un sistema (hardware o software) que permiten que un atacante pueda comprometer la integridad, disponibilidad o confidencialidad del sistema, causando un daño negativo en el mismo.
- **XML**: Lenguaje de Marcas Extensible o XML (por sus siglas en inglés) es un lenguaje utilizado para almacenar datos de forma legible y poder estructurar documentos o programas de gran tamaño, permite el intercambio de datos entre distintas plataformas.
- **XMPP:** Protocolo Extensible de Mensajería y Comunicación de Presencia o XMPP (por sus siglas en inglés), es un protocolo abierto y extensible basado en XML, originalmente ideado para mensajería instantánea

Bibliografía

[1]. Rivera,J. & Van der Meulen, R. (2014). *Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013*. 12/09/2015, de Gatner Sitio web: http://www.gartner.com/newsroom/id/2665715.

[2]. Guía Local. (2013). *México: Los mexicanos prefieren teléfonos con Android*. 12/09/2015, de Guía Local Blog Sitio web: http://guialocal.com/blog/es/2014/01/02/mexico-los-mexicanos-prefieren-telefonos-con-android/.

[3]. Developers. (2015). *Dashboards*. 12/09/2015, de Google Inc. Sitio web: https://developer.android.com/about/dashboards/index.html.

[4]. Gupta, A. (2014). *Learning Pentesting for Android Devices*. United Kingdom: Packt Publishing.

[5]. Android Team. (2013). *Android Anatomy and Physiology*. 12/09/2015, de Google Inc. Sitio web: http://androidteam.googlecode.com/files/Anatomy-Physiology-of-an-Android.pdf.

[6]. Wikipedia. (2008). *Android (operating system)*. 12/09/2015, de Wikipedia Sitio web: https://en.wikipedia.org/wiki/Android_(operating_system).

[7]. Levin, J. (2012). *Mac OS X and iOS Internals: To the Apple's Core*. United States: Wrox.

[8]. Wikipedia. (2008). *iOS*. 12/09/2015, de Wikipedia Sitio web: https://en.wikipedia.org/wiki/IOS.

[9]. Wikipedia. (2005). *Darwin (operating system)*. 12/09/2015, de Wikipedia Sitio web: https://en.wikipedia.org/wiki/Darwin_(operating_system).

[10]. Apple Developer. (2014). *Core OS Layer*. 12/09/2015, de Apple Inc. Sitio web: https://developer.Apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/CoreOSLayer/CoreOSLayer.html#//Apple_ref/doc/uid/TP40007898-CH11-SW1.

[11]. The Penetration Testing Execution Standard Team. (2014). *The Penetration Testing Execution Standard*. 12/09/2015, de MediaWiki Sitio web: http://www.pentest-standard.org/index.php/Main_Page.

[12]. Developers. (2015). *SDK Index*. 12/09/2015, de Google Inc. Sitio web: https://developer.android.com/sdk/index.html.

[13]. Developers. (2015). *Android Debug Bridge*. 12/09/2015, de Google Inc. Sitio web: http://developer.android.com/tools/help/adb.html.

[14]. Vanessaem. (2014). *[APP] SuperOneClick v2.3.3 - Motorola Exploit Added!*. 12/09/2015, de XDA Developers Sitio web: http://forum.xda-developers.com/showthread.php?t=803682.

[15]. MacGyver_93. (2013). *Instalar Custom ROM (paso a paso y de forma gráfica).* 12/09/2015, de HTCmania Sitio web: http://www.htcmania.com/showthread.php?t=553572.

[16]. Wikipedia. (2008). *Android rooting*. 12/09/2015, de Wikipedia Sitio web: http://en.wikipedia.org/wiki/Android_rooting.

[17]. PaulOBrien. (2009). *[Testing] Getting an emulator up and running a full ROM with the Market etc*. 12/09/2015, de Modaco Sitio web: http://www.modaco.com/forums/topic/289928-testing-getting-an-emulator-up-and-running-a-full-rom-with-the-market-etc/.

[18]. Apple Developer. (2014). *Xcode Overwiev*. 12/09/2015, de Apple Inc. Sitio web: https://developer.Apple.com/library/mac/documentation/ToolsLanguages/Conceptual/Xcode_Overview/Xcode_Overview.pdf.

[19]. Sadeghi, A. & Davi, L. (2012). *Overview on Apple iOS*. 12/09/2015, de Cased Sitio web: https://www.trust.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_TRUST/LectureSlides/ESS-SS2012/9_iOS_-_hand-out.pdf.

[20]. IPhone Wiki. (2015). *iBoot (Bootloader)*. 12/09/2015, de The iPhone Wiki Sitio web: https://www.theiphonewiki.com/wiki/IBoot_(Bootloader).

[21]. Apple Developer. (2014). iOS Simulator Guide. 12/09/2015, de Apple Inc. Sitio web: https://developer.Apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/iOS_Simulator_Guide.pdf

[22]. Edge, C. (2015). Learning iOS Security. United Kingdom: Packt Publishing.

[23]. Avala, F. (2014). Desarrollo Apps iOS 8. 14/09/2015, de PixyBit Sitio web: https://www.pixybit.es/curso/desarrollo-Apps-ios/capitulo-6.html

[24]. Apple Inc. (2013). Códigos de iOS. 14/09/2015, de Apple Inc. Sitio web: https://support.Apple.com/es-es/HT4113

[25]. Apple Inc. (2013). Usar Touch ID en iPhone y iPad. 14/09/2015, de Apple Inc. Sitio web: https://support.Apple.com/es-es/HT201371