Universidad de Buenos Aires

Facultad de Ciencias Económicas, Ciencias Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Titulo

Symbian OS y su Sistema de Seguridad Basado en Confianza

symbian

Autor

Luis Arturo Marroquín Rodríguez

Tutor

Rodolfo Baader

Fecha

Agosto 2011

Resumen

Symbian OS se diseñó como un sistema operativo orientado a objetos para las plataformas de teléfonos inteligentes. Tiene un diseño de microkernel que utiliza un núcleo o nanokernel muy pequeños, que implementa las funciones del kernel de una manera más rápida pero aún bajo una metodología primitiva. Symbian OS emplea una arquitectura cliente/servidor que coordina el acceso a los recursos del sistema con servidores que trabajan en función del usuario. Aunque está diseñado para teléfonos inteligentes, Symbian OS tiene muchas características de un sistema operativo de propósito general: procesos e hilos, administración de la memoria, soporte para un sistema de archivos y una infraestructura de comunicaciones extensa. Symbian OS implementa ciertas características únicas; por ejemplo, los objetos activos hacen mucho más eficiente la espera de los eventos externos, la falta de memoria virtual hace más desafiante la administración de la memoria y el soporte a la orientación a objetos en los controladores de dispositivos utiliza un diseño abstracto de dos capas.

Un teléfono móvil únicamente tiene un usuario, por lo que la política de seguridad, a diferencia de un entorno de escritorio, no trata de proteger a los diferentes usuarios unos de otros. Al contrario, lo que se audita dentro de las Unidades de Confianza de Symbian son las acciones que pueden llegar a practicar sus aplicaciones.

Palabras Claves

Symbian, Reglas de Confianza, Seguridad, Teléfonos Móviles, Sistemas Operativos, Vulnerabilidades.

Agradecimiento

En primer lugar, quiero hacer una especial mención de agradecimiento a mis padres Arturo y María Luisa, que siempre me brindaron ese apoyo incondicional y puntual, cuando más lo he necesitado; a mis hermanos y amigos con quienes pude contar en momentos de quiebre. En fin a todos quienes han sido mi escapatoria ante los problemas y dificultades impuestas por el fuerte trabajo llevado a cabo, y que sin ellos, todo esto hubiera sido prácticamente imposible.

Quiero también agradecer a mi tutor Rodolfo Baader por su ayuda, y su espíritu trabajador, durante la realización del trabajo final de Especialización. Gracias a sus correcciones y aportaciones técnicas, sus valoraciones objetivas, y sus consejos, ha hecho posible la finalización de este trabajo de investigación.

Muchas gracias a todos por su apoyo.

Índice

Lista de Figuras

Lista de Tablas

Introducción

En la actualidad más del 90% de los ordenadores en el mundo no están distribuidas en equipos de escritorio o portátiles; al contrario, se encuentran en sistemas embebidos o incrustados entre los cuales están los teléfonos celulares inteligentes o *smartphones*[1].

Según datos de la consultora Gartner[1], el sistema operativo para teléfonos móviles más vendido es *Symbian*, con 80,87 millones de unidades de *smartphones* en el 2009. El exponencial avance de estos nuevos aparatos ha permitido que el usuario se desempeñe de una manera más eficiente y productiva en varios aspectos de su vida.

Motivado por estas premisas, surge la idea de realizar una exhaustiva investigación del renombrado Symbian SO v9 con respecto a la forma de encarar potenciales intrusos, evaluar su plataforma de seguridad, y analizar la efectividad de su sistema basado en confianza.

Objetivo general:

El presente trabajo tratará sobre el análisis conceptual de la arquitectura de seguridad implementada por Symbian OS v9, su sistema basado en Unidades de Confianza, sus potenciales Vulnerabilidades y un análisis estándar de Robustez Seguridad y Estabilidad.

Objetivos específicos:

- Conocer:
  - Qué es la Virología Móvil
  - Cuáles son las principales Vulnerabilidades que afectan a Symbian OS v9.

- Atacar dispositivos que lleven Symbian OS v9 como su Sistema Operativo, perpetrando:
  - Escaneo de Red
  - Vulnerabilidades
  - Test de Penetración

[1] TANENBAUM, *Andrew*, Sistemas Operativos Modernos.

Capitulo 1

1. Symbian OS

1.1. Introducción Histórica

Symbian tiene una historia bastante corta. Sus raíces datan de los sistemas utilizados por dispositivos de bolsillo desarrollados en la década de los 90`s (Psion[2] y EPOC); aunque su debut fue apenas en el 2001, Symbian se ha desarrollado con rapidez a través de varias versiones.

1.2. Raíces de Symbian OS

La herencia de Symbian OS se remonta a 1980, cuando surgió la necesidad de simplificar el espacio de una computadora de escritorio (sin perder sus principales características), en algo más portable. Es conocido que este campo tuvo un lento inicio, sin embargo, las computadoras de bolsillo desarrolladas a mediados de 1990 estaban mejor adaptadas al usuario y a la forma en que las personas utilizaban los dispositivos móviles. Este tipo de computadoras se diseñaron en un principio como un Asistente Personal Digital o PDA (del inglés Personal Digital Assistant), pero evolucionaron para abarcar muchos tipos de funcionalidades. A medida que se desarrollaron, su comportamiento se asemejaba más al de los ordenadores de escritorio, incurriendo en las mismas necesidades como por ejemplo, la multitarea, la incorporación de capacidades de almacenamiento, entre otros; es decir, presentaban flexibilidad para la entrada y salida de datos.

Estos dispositivos de bolsillo evolucionaron para abarcar el mercado de las futuras comunicaciones celulares a inicios de los 90, dando paso a la creación de los teléfonos inteligentes. Para aquello, fue necesario el desarrollar sistemas operativos propietarios, que operasen dichos dispositivos inteligentes a medida que se realizaba esta fusión.

Psion Inc. fabricante de dispositivos PDA, produjo la Serie 3: una pequeña computadora con una pantalla monocromática que tenía una

resolución equivalente a la mitad de VGA y cabía en un bolsillo. Después de la serie 3, su sucesora la 3c fue lanzada en 1996, con capacidad infrarroja adicional, posteriormente la Serie 3mx salió en 1998, la cual contaba con mayor velocidad de procesamiento y memoria. Cada uno de estos dispositivos tuvo mucho éxito, en gran parte debido a la buena administración de energía y la interoperabilidad con computadoras de escritorio y otros dispositivos de bolsillo. La programación del sistema operativo con el cual operaban estos dispositivos, se basaba en lenguaje C, tenía un diseño orientado a objetos y empleaba motores de aplicaciones. Dichos motores se gestionaban a manera de servidores de las aplicaciones que los invocaban, y su administración se realizaba en función a las peticiones de sus aplicativos. Es decir a manera de un cliente/servidor. Esta metodología hizo posible la estandarización de una Interfaz de Programación de Aplicaciones o API (del inglés Application Programming Interface) y el uso de la abstracción de objetos para que el programador de aplicaciones dejara de preocuparse sobre los detalles tediosos como los formatos de datos.

En 1996, Psion rediseñó todo el sistema operativo, basándose en una arquitectura de 32 bits, la cual era permisible a dispositivos señaladores de pantallas táctiles, utilizaba multimedios y tenía capacidades más completas de comunicación. El nuevo sistema ahora orientado a objetos contaba con la facilidad de poder migrarlo a distintas arquitecturas e implementarlo en diferentes dispositivos de esta gama. El resultado del esfuerzo de Psion fue la introducción del sistema conocido como EPOC Release 1. Este sistema se programó en C++[3] y estaba diseñado para ser orientado a objetos desde sus cimientos. También utilizó la metodología de los motores y expandió esta idea de diseño en una serie de servidores que coordinaban el acceso a los servicios del sistema y los dispositivos periféricos. EPOC en sus versiones ER3 y ER5, expandió las posibilidades de comunicación, abrió el sistema operativo a la tecnología multimedia, introdujo nuevas plataformas para interconectar elementos como las pantallas táctiles y generalizó la interfaz de hardware.

Para el 2000, la mayoría de las oportunidades para el desarrollo de nuevos equipos de bolsillo estaba en el negocio de los teléfonos móviles, donde los fabricantes ya estaban buscando un sistema operativo nuevo y avanzado para su próxima generación de dispositivos. Para aprovechar estas oportunidades, Psion y los líderes en la industria de los teléfonos móviles, entre ellos Nokia[4], Ericsson[5], Motorola[6] y Panasonic[7], formaron una empresa conjunta llamada Symbian para que asumiera la propiedad del sistema operativo EPOC y se encargara del futuro desarrollo de su núcleo. Este nuevo diseño del núcleo se conoce ahora como Symbian OS.

1.3. Características, Funcionalidad y Diseño de Symbian OS

1.3.1. Objetivos de Diseño

Según sus desarrolladores, el principal reto al momento de rediseñar el kernel para Symbian OS fue el mantener la herencia funcional de sus antecesores. Es decir que el nuevo kernel mantenga las siguientes características:

- Ser un Sistema Operativo tradicionalmente embebido.
- Adecuado para entornos con recursos limitados.
- Modular
- Portable
- Robusto y
- Brindar alta integridad y seguridad.

Es de suponer que el nuevo kernel iba a presentar funcionalidades y características nuevas con el objetivo de conseguir un óptimo rendimiento. Este gran objetivo se divido en varios subobjetivos y requisitos:

- Operaciones más rápidas.
- Longitud de interrupciones predeterminadas.
- Temporizadores con una resolución más fiable.

Luego consideraron como más se podría mejorar el rendimiento del Sistema Operativo, volcando dichos requerimientos en la siguiente lista:

- Fácil Portabilidad.
- Fiabilidad ante aplicaciones maliciosas o mal codificadas.
- Habilitar soluciones que corran en un mismo núcleo del procesador.
- Proporcionar un mejor y más estable entorno.
- Simplificarles el trabajo a desarrolladores externos.

Algo que también se tuvo que tomar en cuenta fue la compatibilidad hacia atrás para con sus sistemas antecesores principalmente con sus librerías de clase EUSER de EKA, que son la base de la interfaz del kernel de Symbian OS.

1.3.2. Arquitectura del Kernel de Symbian OS

Con los objetivos de diseño bien planteados, se obtuvo un Sistema Operativo con una arquitectura de muy alto nivel. En la Figura 1.1 se muestra el esquema interno del kernel de Symbian apilado en bloques de construcción, en donde se muestran también los servidores de archivos y ventanas.

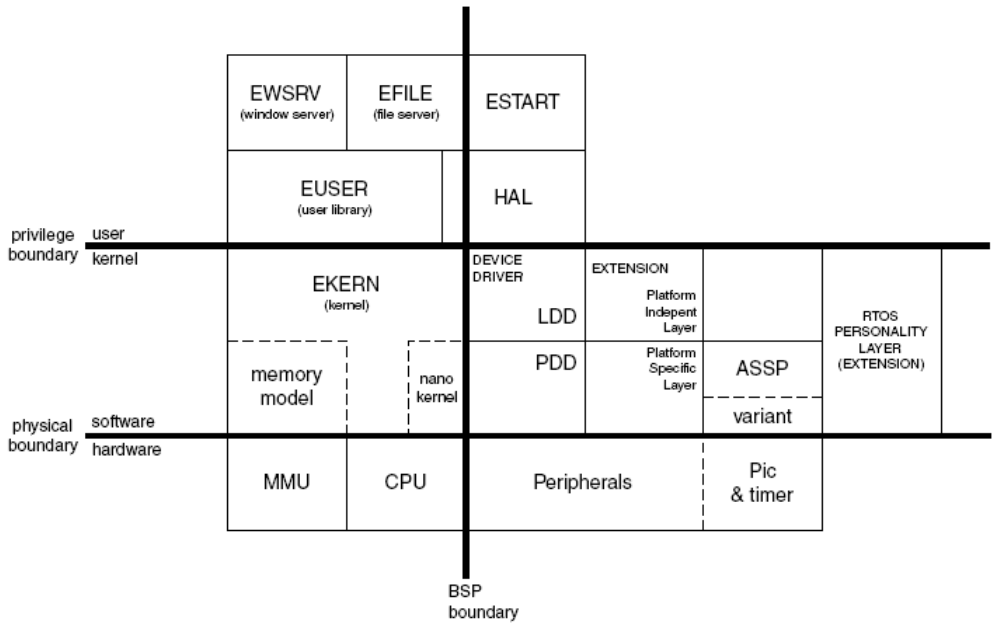


Figura 1.1 Revisión Arquitectura Symbian OS[2]

[2] http://www.developer.nokia.com/Community/Wiki/images/6/64/SymbianOSInternalsBook_12.1.png?20110117033930

1.4. Nanokernel

La principal función del nanokernel (llamado así por las funciones a bajo nivel que desempeña), es proporcionar simplicidad en sus procesos, sean estos a nivel de supervisión, administración o aplicaciones. El nanokernel es el encargado de manejar las interrupciones de procesos, temporizadores de API como NTimer y Nkern, es decir se preocupa de las interrupciones y excepciones.

Una particularidad interesante para mencionar es que el nanokernel, a diferencia del kernel de Symbian, es quien interactúa con la librería de usuario EUSER, encargada de gestionar las funciones del usuario, y los controladores para los dispositivos. Una consideración importante es la forma en la cual el nanokernel presenta su reporte de errores: este lanza un código de error seguido de la neutralización del proceso que se esté ejecutando.

Se conoce que el principal problema del nanokernel es la falta de interacción dinámica con la memoria, pero de requerirlo, el nanokernel simplemente asume que el espacio de memoria que necesita ya fue preasignado por un gestor de memoria del Sistema Operativo.

1.5. Kernel de Symbian OS

El kernel de Symbian OS es el encargado de proporcionar la funcionalidad que necesita el sistema operativo, esto a partir de hilos, procesos simples y los servicios de bajo nivel del sistema proporcionados por el nanokernel. Cuenta también con un rango más sofisticado de objetos de sincronización (al contrario de su nanokernel), apoya la herencia de prioridad en un hilo conductor, el cual hereda la prioridad del hilo que contenga la prioridad más alta en espera, esto sucede siempre y cuando dicha prioridad sea mayor a la propia.

Contrastando con el nanokernel, el núcleo de Symbian OS permite una interacción con la memoria del componente (teléfono inteligente) de forma dinámica por medio de un gestor que utiliza servicios de bajo nivel conocido como el modelo de memoria.

La figura 1.2 muestra un diagrama de la estructura completa del kernel de Symbian OS.

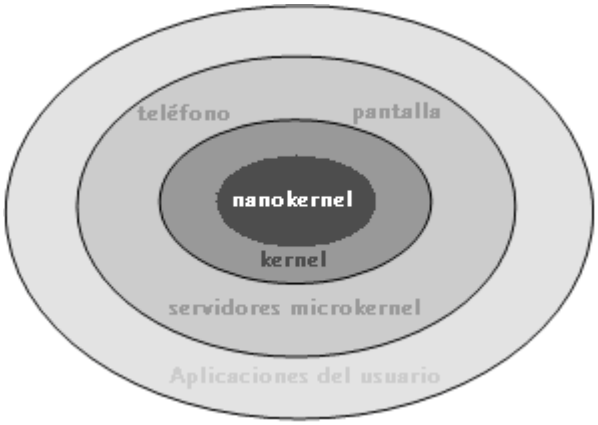


Figura 1.2 Niveles de estructura kernel Symbian OS[3]

1.5.1. Acceso a los recursos de cliente/servidor

Como se mencionó anteriormente, Symbian OS explota el diseño de su nanokernel e incluye un modelo cliente/servidor para acceder a los recursos del sistema. Las aplicaciones que necesitan acceder a los recursos del sistema son los clientes; los servidores son programas que el sistema operativo ejecuta para coordinar el acceso a estos recursos. Mientras que en Linux[8] podríamos llamar a la función *open* para abrir un archivo, o en Windows[9] usar un API de Microsoft para crear una ventana, en Symbian OS ambas secuencias son iguales: primero se debe invocar a un servidor, éste reconoce la conexión y gestiona las peticiones entrantes para realizar ciertas funciones. Por lo tanto, la función para abrir un archivo en Symbian OS ejecutará la siguiente tarea: buscar el servidor de archivos, llamar a *connect* para establecer una conexión con el servidor y después enviarle una petición *open* con el nombre de un archivo específico.

Hay varias ventajas de esta forma de proteger los recursos. En primer lugar, se adapta al diseño del sistema operativo como un sistema orientado a objetos y como un sistema basado en nanokernel. En segundo lugar, este tipo de arquitecturas es muy efectiva para administrar los múltiples accesos a los recursos del

[3] TANENBAUM, *Andrew*, Sistemas Operativos Modernos.

sistema que requería un sistema operativo multitareas y multihilo. Por último, cada servidor se puede enfocar en los recursos que debe administrar; además se puede actualizar con facilidad e intercambiar por nuevos diseños.

1.6. Características de Symbian OS

A pesar del tamaño de los dispositivos electrónicos a los cuales está destinado su uso, Symbian OS tiene muchas de las características de sus parientes más grandes. Aunque es factible el esperar el mismo tipo de soporte que se ve en los sistemas operativos de escritorio como Linux o Windows, lo más seguro es el encontrarse con estas características de una forma distinta.

Procesos e hilos.- Symbian OS es un sistema operativo multitarea y multihilo; es decir, muchos procesos se pueden ejecutar en forma concurrente, gestionando una comunicación paralela y apoyado en la utilización de varios hilos que se ejecutan en forma interna para cada proceso.

Soporte común de los sistemas de archivos.- De forma similar a los sistemas operativos más robustos, Symbian OS organiza el acceso al entorno de almacenamiento del sistema mediante el uso de un modelo de sistema de archivos. Cuenta con un sistema de archivos compatible con Windows, y, aunque utiliza de manera predeterminada un sistema de archivos tipo FAT-32, admite otras implementaciones de sistema de archivos mediante el uso de una interfaz de estilos como complemento, como por ejemplo FAT-16, NTFS y muchos formatos de tarjetas de almacenamiento como JFFS.

Redes.- Symbian OS acepta redes TCP/IP y varias interfaces más de comunicación, como serial, infrarroja y Bluetooth[10].

Administración de la memoria.- Aunque Symbian OS no cuenta con la capacidad de gestionar un modulo de memoria virtual (ni tiene las herramientas para ello), organiza el acceso a la memoria en páginas y permite reemplazarlas; es decir, traer páginas a la memoria, pero no intercambiarlas fuera de ella.

1.7. Comunicación entre procesos

En un entorno multihilo como es el de Symbian OS, la comunicación entre procesos es crucial para el rendimiento del sistema. Los hilos, en especial, se comportan en forma de servidores del sistema, permitiendo una interacción constante.

Un *socket* es el modelo básico de comunicación utilizado por Symbian OS. Es una línea de tubería de comunicación abstracta entre dos extremos. La abstracción se utiliza para ocultar los métodos de transporte y la administración de los datos entre los extremos. Symbian OS utiliza el concepto de *sockets* para comunicarse entre clientes y servidores, hilos y dispositivos y entre los mismos entornos del sistema (hilos en memoria).

El modelo del *socket* también forma la base de comunicaciones entre entornos de entrada y salida del dispositivo. De nuevo, la abstracción es la clave para que este modelo sea útil. Toda la mecánica del intercambio de datos con un dispositivo se administra mediante el sistema operativo, en lugar de que lo haga la aplicación. Por ejemplo, los *sockets* que funcionan a través del protocolo TCP/IP en un entorno de red se pueden adaptar con facilidad para trabajar en un entorno Bluetooth, cambiando los parámetros en el tipo de *socket* a utilizarse. Así el sistema operativo se encarga de la mayor parte del trabajo de intercambio de los datos al momento de incurrir en un cambio de este tipo.

Symbian OS implementa las primitivas de sincronización estándar que se encuentran en la mayoría de sistemas operativos de propósito general, utilizando también muchas formas de semáforos gestionados por el sistema. Éstos proporcionan la sincronización entre los procesos y los hilos.

1.8. Sistemas de Almacenamiento

Al igual que todos los sistemas operativos orientados a ser utilizados por usuarios, Symbian OS trabaja con un sistema de archivos no propietario, sobre el cual se trata a continuación.

1.8.1. Sistemas de archivos de Symbian OS

Debido a que Symbian OS es un sistema operativo diseñado para teléfonos móviles inteligentes, la necesidad de implementar por lo menos un sistema de archivos base fue imprescindible. Sin duda se volcó hacia FAT-32 debido a que interactúa con ese sistema de archivos en la mayor parte de sus medios de almacenamiento.

Sin embargo, la implementación del servidor de archivos de Symbian OS se basa en una abstracción, en una forma muy parecida a la del sistema de archivos virtual de Linux. La orientación a objetos permite conectar servicios que implementan varios sistemas operativos al servidor de archivos de Symbian OS, con lo cual se puede utilizar muchas soluciones basadas en distintos sistemas de archivos. Incluso permitiendo que diferentes implementaciones pueden coexistir en el mismo servidor de archivos.

Existe también soporte para implementaciones creadas para los sistemas de archivos NFS y SMB bajo un entorno propio para Symbian OS.

1.8.2. Seguridad y protección del sistema de Archivos

La seguridad de los teléfonos inteligentes es una interesante variación en la seguridad computacional en general. Hay varios aspectos de los teléfonos inteligentes que convierten la seguridad en un reto. Symbian OS ha realizado varias elecciones de diseño que lo diferencian de los sistemas de escritorio de propósito general y de otras plataformas de teléfonos inteligentes.

Es conocido que el entorno funcional adoptado por teléfonos inteligentes fue concebido para interactuar con un solo usuario, el cual no siempre requiere de algún tipo de identificación para acceder al

dispositivo. El usuario de un teléfono puede ejecutar aplicaciones, marcar un número telefónico y acceder a las redes; todo ello sin identificación. En este entorno, el uso de la seguridad basada en permisos es desafiante debido a que la falta de identificación significa que sólo es posible un conjunto de permisos: el mismo para todos.

En vez de permisos de usuario, la seguridad con frecuencia aprovecha otros tipos de información. En la versión 9 y posterior de Symbian OS, las aplicaciones reciben un conjunto de capacidades al instalarse. Este conjunto de capacidades otorgadas a una aplicación se comparan con el ingreso de credenciales al momento de solicitar acceso por parte de un usuario; en caso contrario, se rechaza su ingreso.

En Symbian OS existen otras formas de seguridad para los archivos. Hay áreas del medio de almacenamiento de Symbian OS que las aplicaciones no pueden utilizar sin una capacidad especial, la cual se proporciona solo a la aplicación que instala software en el sistema. El efecto de esto es que después de instalar las nuevas aplicaciones, estas quedan protegidas contra el acceso que no sea del sistema (lo cual significa que los programas maliciosos que no sean del sistema, como los virus, no pueden infectar a las aplicaciones instaladas). Además, hay áreas del sistema de archivos reservados de manera específica para ciertos tipos de manipulación de datos por parte de una aplicación (a estos se le conoce como jaulas de datos).

Para Symbian OS, el uso de capacidades ha funcionado bien, así como la propiedad de los archivos para proteger el acceso a los mismos.

Capitulo 2

2. Esquema de Seguridad de Symbian OS

2.1. La Cadena de Valor de Symbian OS

Una vez examinado el alto nivel de metas que pretendía abarcar la arquitectura de seguridad de la plataforma, es también pertinente revisar los principales desafíos con los que se enfrentaría. Para dicha tarea hay varios tipos de organización que contribuyeron con el desarrollo de Symbian OS para lograr un producto final orientado a la telefonía móvil.

La creación de una "cadena de valor" compuesta por el usuario final, el fabricante del dispositivo, y Symbian entre otros (todos los eslabones de la cadena de valor se los puede apreciar en la figura 2.1), como proveedor del sistema operativo, tuvo un papel crucial a la hora de construir una solución segura; este punto fue importante al momento de brindar confianza, seguridad y soporte para sus futuros clientes y usuarios.

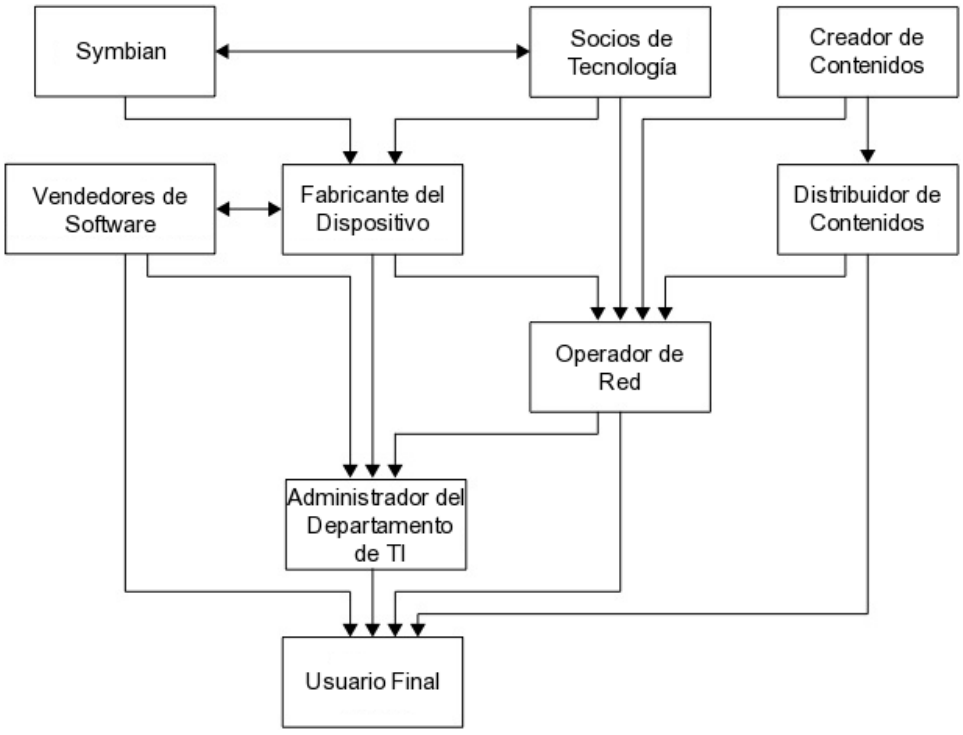


Figura 2.1 Eslabones de la cadena de valor

Partiendo del gráfico anterior es fácil suponer la estrecha relación que deben tener en especial, el distribuidor del sistema operativo “Symbian”, sus Socios de Tecnologías, y el Fabricante del Dispositivo; esto debido principalmente a la necesidad de crear una plataforma lo suficientemente segura y confiable. Por otro lado, el mayor objetivo de la integración de dichos actores a un proceso de creación de un producto final, es alcanzar una solidez en la toma de decisiones y no dejar pasar por alto ninguna brecha de seguridad tanto en software como en hardware.

Por otro lado, la participación tanto de vendedores de software como proveedores y distribuidores de contenidos para fomentar la difusión de Symbian como una plataforma segura, es vital dentro de su cadena de valor. Como característica principal el uso de certificados y firmas digitales propuesto por el consorcio Symbian, conllevó a que los desarrolladores de aplicaciones para dispositivos móviles tanto como los usuarios finales se sientan cómodos con las mejoras implementadas en las versiones más nuevas del sistema operativo de Symbian; palpando cambios muy significativos en temas de distribución, desarrollo y seguridad de aplicaciones.

Continuando con la cadena de valores, los administradores de departamentos de TI y Operadoras de Red, también contribuyen de forma significativa en la seguridad de dispositivos móviles; estos son los encargados de proveer la infraestructura necesaria para asegurarle al cliente que se encuentra conectado a un canal seguro, principalmente al momento de realizar actualizaciones o parcheo del sistema, transferencia de paquetes y/o comunicaciones en general.

Al final, pero no menos importante en la cadena de valores se encuentra el usuario final; quien a leves rasgos está encargado de velar por su bienestar, es decir debería estar consciente de las aplicaciones que necesita instalar en su dispositivo móvil. Este concepto se lo maneja a raíz de la aparición de aplicaciones maliciosas (malware), que se encuentran ocultas dentro de aplicaciones útiles, más conocidas como Caballos de Troya[11]. Por tal motivo algo que se debería tomar en cuenta es la creación de respaldos de la información del dispositivo

móvil en forma periódica, con el objetivo de prevenir perdidas de información por daños lógicos o físicos.

2.2. La Seguridad Como Una Propiedad Holística

Es pertinente citar que para lograr un esquema seguro, es necesario el compromiso de muchos actores; partícipes de la concepción de un entorno 100% seguro, estable y confiable a la vista y requerimientos de un potencial usuario final. A estos gestores se los puede apreciar con más claridad en la figura 2.2, desde un punto de vista más teórico que técnico.

Figura 2.2 Bloques de Seguridad[4]

Basado en la figura anterior es fácil suponer que el principal objetivo de crear una plataforma segura, es brindar confianza al usuario final. Este punto se fundamenta gracias al buen accionar y seguimiento de estándares, políticas y normas. Un error grave por parte del usuario es asumir que si uno de estos bloques de seguridad se encuentra activo en un entorno desconocido, este es potencialmente seguro. Esta premisa llegaría a cumplirse si y solo si, dicha característica o bloque de seguridad, se encuentre embebida en el sistema operativo, dando la apariencia de compatibilidad. Esta particularidad se debe a que dichos bloques de seguridad del sistema están estructurados bajo un estándar jerárquico organizacional.

[4] TANENBAUM, *Andrew*, Sistemas Operativos Modernos.

2.3. Seguridad en Symbian OS

Tomando el hecho de que tanto teléfonos inteligentes como PDA que llevan instalado Symbian como Sistema Operativo, poseen un entorno difícil de hacer seguro, por ello se especula con que dichos dispositivos no son lo suficientemente robustos al momento de brindar integridad para con los datos del usuario y disponibilidad para el equipo. Sumado a lo mencionado anteriormente se encuentra el hecho de que el usuario cuenta con la libertad de usar diversas funciones básicas e incluso algunas más complicadas como instalar aplicaciones en el sistema, todo esto sin la necesidad de solicitarle algún tipo de autenticación.

Según las habituales tendencias en teléfonos inteligentes que cuentan con Symbian como sistema operativo, los usuarios esperan acceder a cualquier tipo de aplicación, funcionalidad o información del equipo sin necesidad de una autenticación; es decir, sin necesidad de iniciar sesión ni de verificar su identidad. Aparte de este poco usual modo de operación (común entre dispositivos celulares), Symbian también es susceptible a virus, gusanos y demás programas maliciosos. Las versiones anteriores del sistema operativo Symbian ofrecían una seguridad tipo guardián es decir: el sistema pide al usuario permisos para instalar cada una de las aplicaciones que no se encuentren embebidas en el sistema. La idea principal de este diseño fue precautelar el origen de las potenciales aplicaciones que el usuario podría requerir instalar, a riesgo de que causen daños en el sistema, pero con la ventaja de que este era consciente de qué programas podría instalar y cuáles podrían ser maliciosos. En conclusión se confiaba en el buen criterio del usuario al momento de instalar y utilizar las aplicaciones del equipo.

Este diseño de guardián tiene mucho mérito. Por ejemplo, un nuevo teléfono inteligente sin más aplicaciones que las instaladas por defecto de fábrica sería un sistema que podría ejecutarse sin errores. Así al instalar solo aplicaciones que el usuario sabía que no eran maliciosas, se podría mantener la seguridad del sistema. El problema con este diseño es que los usuarios no siempre conocen las ramificaciones

completas del software que van a instalar. Hay virus que se enmascaran como programas útiles y realizan funciones de utilidad mientras instalan en silencio código malicioso. Los usuarios regulares no pueden verificar la confianza completa de todo el software disponible.

Esta verificación de la confianza es lo que provocó un rediseño completo de la seguridad de la plataforma para la versión 9 del sistema operativo Symbian. Esta versión de Symbian mantiene el modelo del guardián, pero asume la responsabilidad de verificar el software en vez de que lo haga el usuario; es decir adopta una postura similar a la de una entidad certificante en el ámbito de la emisión de certificados digitales. Ahora cada desarrollador de software es responsable de verificar sus propias aplicaciones por medio de un proceso conocido como firmado, el cual es posteriormente verificado por el sistema operativo Symbian. No todo software requiere esta verificación, sino sólo aquel que accede a ciertas funciones esenciales del sistema.

Cuando una aplicación requiere un firmado, éste se lleva a cabo a través de una serie de pasos[5]:

1. La casa desarrolladora de la aplicación debe obtener un ID de distribuidor de un tercero en quien se confía, es decir de una entidad certificante. Estos terceros de confianza están certificados por Symbian.
2. Cuando un desarrollador produce un paquete de software y desea distribuirlo, debe enviarlo a un tercero para que lo valide. El desarrollador envía su ID de distribuidor, el software y una lista de formas en que el software accede al sistema.
3. El tercero de confianza verifica que la lista de tipos de acceso del software esté completa y que no ocurra ningún otro tipo de acceso. Si el tercero puede realizar esta verificación, entonces firma ese software. Esto significa que el paquete de instalación tiene una cantidad especial de información que explica con detalle lo que hará en el sistema operativo de Symbian, y qué puede llegar a hacer.

[5] TANENBAUM, *Andrew*, Sistemas Operativos Modernos.

4. Este paquete de instalación se envía de vuelta al desarrollador del software, para que lo distribuya a los usuarios. Este método depende de la forma en que el software accede a los recursos del sistema. Esta idea de las capacidades está integrada en el kernel del sistema operativo de Symbian. Cuando se crea un proceso, parte de su bloque de control registra las capacidades que se le otorgan. En caso de que el proceso trate de realizar un acceso que no se liste en estas capacidades, el kernel denegará ese acceso.

Como resultado final se cuenta con aplicaciones firmadas y avaladas por la Corporación Symbian. Esta premisa llega a cumplirse gracias a un proceso de certificación previo, el cual permite distribuir aplicaciones firmadas; con este proceso se logra emular un sistema de confianza. Otro punto importante es la tarea de verificación, la cual está a cargo de un guardián automatizado integrado en Symbian OS el cual puede verificar el software que se va a instalar. Si la firma del paquete es válida, el kernel otorga los permisos a la aplicación al momento de su ejecución.

La figura 2.3 ayuda a comprender de mejor manera las relaciones de confianza con las que se maneja Symbian. Es pertinente citar que existen varios niveles de confianza integrados en el sistema operativo. De igual manera, hay ciertas aplicaciones que no acceden para nada a los recursos del mismo, y por lo tanto no requieren firma alguna por parte de la entidad certificante, un ejemplo de esto podría ser una aplicación simple que solo muestre algo en la pantalla, estas aplicaciones no son de confianza, pero tampoco necesitan serlo. El siguiente nivel está compuesto por las aplicaciones firmadas a nivel de usuario, las cuales únicamente reciben los permisos que necesitan. El tercer nivel de confianza está compuesto por los servidores del sistema. Al igual que las aplicaciones que corren a nivel de usuario, estos servidores tal vez sólo necesiten ciertos permisos para realizar sus labores.

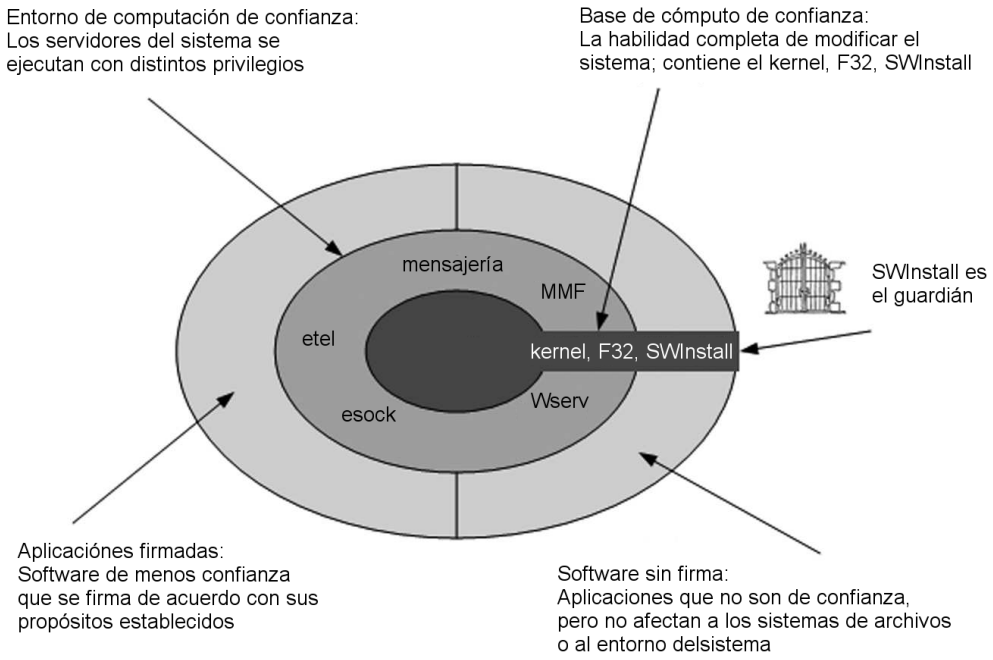


Figura 2.3 Relaciones de confianza de Symbian OS[6]

En una arquitectura de microkernel como con la que cuenta Symbian, tanto servidores como aplicaciones de terceros se ejecutan a nivel de usuario y por ende son de confianza. Por último, hay una clase de programas que requieren una confianza absoluta del sistema, este conjunto de programas tienen la habilidad total de modificar el sistema, esto debido a que dichas aplicaciones llevan consigo código del kernel, el mismo que les permite hacer las modificaciones necesarias para instalar su aplicación.

Es importante citar que los procesos que son necesarios para la instalación de una aplicación con nivel de privilegios altos, similar a la de un usuario, estarán conjuntamente detalladas con la petición de certificación de la aplicación, es decir Symbian conoce cuales serán los pasos a seguir para lograr una instalación exitosa de la aplicación, confiando siempre en que el desarrollador no incrusto una funcionalidad oculta a dicha aplicación con el fin de tomar posesión de información sensible del equipo.

[6] TANENBAUM, *Andrew*, Sistemas Operativos Modernos.

Hay varios aspectos de este sistema operativo que podrían parecer cuestionables. Por ejemplo, el hecho de que el sistema de firmas de Symbian OS reemplaza a los usuarios como el verificador de la integridad del software, conlleva a una verificación en tiempo real. Por ende, podría parecer que los procesos dificultan a programadores y desarrolladores al momento de testear sus aplicaciones, es decir se pensaría que para cada prueba en el hardware real se requiere un nuevo paquete de instalación firmado. Para responder a esto, Symbian OS reconoce un firmado especial para los desarrolladores. Un desarrollador debe obtener un certificado especial firmado digitalmente, el cual tiene un tiempo de validez (por lo general de 6 meses) y es específico para cada modelo de teléfono celular. Así, el desarrollador puede crear sus propios paquetes de instalación con el certificado digital que le fue asignado.

Además de esta función de guardián en la versión 9, Symbian OS emplea algo conocido como jaulas de datos, donde los datos se organizan en ciertos directorios. El código ejecutable existe sólo en un directorio en el que sólo la aplicación de instalación del software puede escribir. Además, los datos de un utilitario se pueden escribir sólo en un directorio, el cual es privado e inaccesible para otros programas.

2.4. Prevención de Incidentes de Seguridad

Existe la posibilidad de que las características de seguridad que se han mencionado anteriormente lleguen a convertirse en una molestia más que en una ayuda, sobre todo para laboratorios especializados en el desarrollo de aplicaciones propietarias. Como ya se ha comentado, el principal beneficiario de estas mejoras de seguridad es el usuario final, aunque cabe destacar la importancia de dichas mejoras al momento de ser adoptadas, especialmente por desarrolladores.

Es pertinente citar las áreas sensibles en las cuales se incrementó su nivel de seguridad, y estas son:

1. Al aumentar la confianza de los usuarios y la confianza en Symbian, la cadena de valor que se mencionó anteriormente

debería beneficiarse, así los usuarios estarán más dispuestos a instalar aplicaciones no propietarias, ya que cuentan con la garantía de que están protegidos de los peores efectos del malware. Esto prolifera la compra de software especializado convirtiendo al teléfono en un dispositivo más abierto a aplicaciones nuevas, es decir, brinda una mejor funcionalidad hacia el usuario final.

2. Los entornos de menor privilegio de acceso dentro de la arquitectura de seguridad de Symbian resultan en defectos de programación no intencionales de aplicaciones no propietarias (por ejemplo, un desbordamiento de búfer) tengan menos probabilidades de ocurrencia, protegiendo así la reputación del desarrollador.
3. La arquitectura de seguridad de Symbian brinda sus servicios de una manera totalmente transparente o mejor dicho abierta; esto con el objetivo de evitar seguridad por oscuridad. Con esta estrategia, Symbian conjuntamente con sus colaboradores encargados de la seguridad de la plataforma, confían en que se producirá software más especializado debido a que estos no tendrán que preocuparse tanto por temas de seguridad, y simplemente deberían enfocarse en su área de conocimiento.

2.5. Seguridad Perimetral SWInstall

El instalador de aplicaciones SWInstall ha evolucionado sustancialmente con el objetivo de soportar los procesos requeridos por las capacidades que a su vez son invocadas por las aplicaciones que el usuario requiera instalar. Hoy en día, el gestor de aplicaciones cuenta con toda una Infraestructura de Clave Pública aunque también fue diseñado con la idea se soportar diferentes tipos de políticas de seguridad.

2.5.1. Validación de Aplicaciones

Un archivo de instalación .SIS puede estar o no firmado digitalmente, y contar o no con todas las capacidades que potencialmente este requeriría para poder ser instalado. SWInstaler es el encargado de verificar si efectivamente una aplicación se encuentra firmada digitalmente para a continuación pasar a construir un certificado digital y enviarlo a una autoridad certificante de confianza. Existen ciertos controles de validación sobre las aplicaciones que requieran ser instaladas; primero, sobre el establecimiento de configuraciones propias de cada aplicación y segundo, sobre el estado de revocación de certificados digitales usando el método OCSP[12] (Protocolo de Estado de Certificados Digitales en Línea). Otra posibilidad es marcar el certificado digital que maneja el SWInstaller bajo un carácter de mandatorio. En el caso de que un certificado mandatorio exista, los paquetes que necesitan ser instalados deben ser firmados por dicho certificado; caso contrario la instalación se declara fallida. Dicho mecanismo agiliza la asociación de un certificado a una aplicación para que esta pueda ser instalada.

2.5.2. Cálculo de Capacidades

La asociación del certificado digital de SWInstaller y el grupo de capacidades que se le han otorgado a una aplicación se llaman meta-capacidades. El instalador es el encargado de calcular el grupo de capacidades que van a ser otorgadas a una aplicación para posteriormente convertirse en sus meta-capacidades; este proceso es satisfactorio siempre y cuando estas se encuentren correctamente firmadas y validadas.

Existen parámetros de configuración que determinan el otorgamiento de capacidades adicionales a una aplicación; esto es factible siempre y cuando el número de capacidades iniciales sea menor que el número de capacidades requeridas por la aplicación. SWInstaller nunca instalará una aplicación que contenga menos

capacidades que las requeridas, esto también ocurre si contiene más de las necesarias o simplemente ninguna.

2.6. Configuración del Instalador

Al momento en que una aplicación se encuentra en etapa de desarrollo, todo el proceso de configuración y puesta a punto son determinadas de manera independiente. Dichos procesos de configuración pueden ser manipulados a futuro de una forma limitada. Por tal motivo, el fabricante del dispositivo es quien decide si el usuario final cuenta con la libertad de asignar las capacidades extras que requiera una aplicación al momento de su instalación. Gracias a las características mencionadas anteriormente, es fácil suponer que un modelo de credenciales no es 100% necesario al momento de instalar una aplicación que no se encuentra firmada digitalmente; pero que si cuenta con las capacidades necesarias para hacerlo.

Dicho comportamiento es algo que las OEMs (fabricante original del equipo), no están dispuestas a permitir como un mecanismo válido de instalación de aplicaciones, principalmente porque se percibe la posible intromisión de aplicaciones que no se encuentran firmadas digitalmente, es decir no son confiables.

2.7. Proceso de Instalación

El paquete de instalación de Symbian contiene un Controlador de Firmas, mismo que acarrea información necesaria para la instalación de la aplicación; cuenta también con un hash[13] criptográfico de cada uno de los archivos instalables. Este hash se encuentra firmado digitalmente, y es continuamente validado por uno de los certificados mandatorios con los que cuenta el dispositivo.

La integridad del Controlador de Firmas y de los hashes de los archivos se mantienen confiables gracias a su consecuente firmado digital, y a que el instalador de la aplicación almacena un log de dicha instalación en el dispositivo.

Capítulo 3

3. Capacidades

3.1. Definición

Una capacidad es un símbolo que debe ser presentado con el fin de obtener acceso a un recurso específico del sistema. En Symbian OS, estos recursos se transforman en servicios, los cuales son gestionados por una API del sistema, mismas que pueden requerir capacidades diferentes para acceder a servicios restringidos, por ejemplo, las funciones proporcionadas por un servidor o controlador del dispositivo, o simplemente el poder acceder a la configuración del sistema. El hecho de que un proceso cuente con una capacidad, indica que este es de confianza y que se encuentra plenamente avalado en las acciones que se le están permitidas realizar.

Symbian OS utiliza las capacidades mencionadas anteriormente para representar los privilegios de acceso con la idea de que, al momento de encontrarse con software que posee una capacidad para acceder a recursos sensibles del sistema, esta pueda acceder sin problema pero bajo un entorno de trabajo limitada o restringida de acuerdo a su nivel de capacidades.

Es pertinente conocer que el kernel de Symbian OS es quien guarda una celosa lista de las capacidades de los procesos que se encuentran en ejecución. Una aplicación puede tener una, ninguna o todo un conjunto de capacidades pertenecientes a un proceso, el cual puede pedir al kernel una comprobación de las mismas antes de decidir si llevar a cabo o no un servicio en su nombre. Symbian OS define 20 tipos de capacidades, categorizadas por privilegios específicos. La idea de contar con diferentes tipos de capacidades es lograr un equilibrio entre la cantidad y complejidad de estas, las cuales se encargan de dar cierto grado de control en cuanto a lo que aceptación de privilegios se refiere. Al contrario de lo que se mencionó anteriormente el tener una sola

capacidad que maneje todo el entorno de autorizaciones lo cual sería muy extremista para este tipo de sistema operativo.

Symbian OS es compatible con tres grandes categorías de capacidades: las que posee el TCB (del inglés Trusted Computing Base), las propietarias del sistema y las del usuario. Esto se ilustra en la Figura 3.1. Suponiendo, en el caso de encontrarse con software de terceros, el instalador de la aplicación actúa a manera de guardián con el objetivo de validar si efectivamente dicho software se encuentra en condiciones de utilizar las capacidades que necesita para concebir la ejecución de su código; en caso contrario, se niega la petición de uso de la capacidades.

Una analogía de trabajo muy similar al firmado digital.

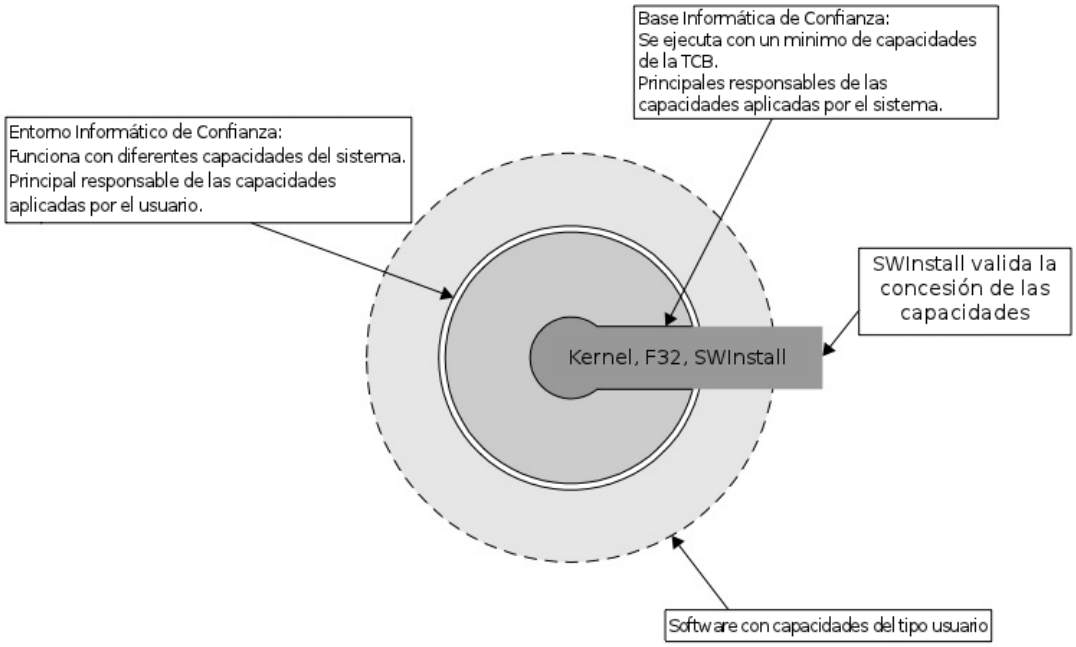


Figura 3.1 Categorización de las Capacidades[7]

Las capacidades cumplen con ser discretas y ortogonales. Esto se refiere a que no son un conjunto jerárquico de tokens de acceso que van añadiendo más y más privilegios hasta alcanzar el nivel de TCB. Al contrario, estos recursos deben ser controlados por una sola capacidad específica, e incluso los procesos básicos de confianza deben poseer dicha capacidad para que el acceso a ese recurso sea exitoso; es decir,

[7] TANENBAUM, *Andrew*, Sistemas Operativos Modernos.

las capacidades no se superponen. Cabe destacar que cada operación a realizarse requiere diferentes capacidades, incluso si esas operaciones se implementan utilizando la Interfaz de Programación de Aplicaciones del propio sistema operativo Symbian. Por ejemplo, diferentes capacidades pueden ser requeridas por un dato influyendo: el sector en donde se encuentre, de si es del tipo usuario o del sistema, e incluso si este dato ya había sido requerido anteriormente bajo las mismas condiciones.

3.2. Tipos de Capacidades

3.2.1. Capacidades de la Base Informática de Confianza TCB

TCB (del inglés Trusted Computing Base), al ejecutarse bajo un nivel máximo de privilegios, requiere se le conceda el uso de todas las capacidades; esto debido a que la TCB es la encargada de hacer lo que nadie más está avalado a hacerlo; por ejemplo la creación de nuevos ejecutables. A este particular estado de capacidades se lo llama 'Tcb', mismas que son concedidas únicamente a TCB.

Existen procesos que cuentan con este tipo de capacidades, mismos que pueden crear nuevos ejecutables e inclusive reasignar las capacidades que inicialmente le fueron asignadas. Dicha característica puede ser mal aprovechada por software malicioso o malware para crear y ejecutar una aplicación cargada con capacidades que este mismo haya otorgado. Así es fácil deducir que el punto débil de Symbian podría ser el otorgamiento de capacidades, por lo cual es sensato pensar que solo se deberían conceder capacidades del tipo ´Tcb´ bajo condiciones estrictamente controladas.

3.2.2. Capacidades del Sistema

Es uno de los grupos más grandes de capacidades con los que cuenta el sistema operativo Symbian. Este grupo de capacidades son las que permiten a un proceso acceder a operaciones sensibles,

mismas que de ser manejadas de una forma indebida, podrían poner en peligro la integridad del dispositivo móvil. A pesar de que las capacidades del sistema no son particularmente significativas para el usuario final, si lo son para el software que este vaya a requerir usar en su dispositivo. Para las eventuales aplicaciones que necesitasen de las capacidades del sistema, es necesario que cuenten con la concesión de las mismas, siendo estas creadas por la ROM del teléfono o en su defecto que la aplicación que las requiera haya sido firmada por una autoridad de confianza.

El responsable de hacer cumplir la mayoría de las capacidades del sistema es la TCB (por ejemplo, los documentos que serán administrados por el servidor de archivos). Algunos controladores del dispositivo también forman parte de la TCB, mismos que requieren de las capacidades del sistema para acceder a ellos (por ejemplo, CommDD). Existe otro tipo de capacidades del sistema que son administradas por el TCE (del inglés Trust Computer Environmentpor sus siglas en inglés), en donde se gestionan los controles de acceso a los servicios de más alto nivel, como por ejemplo, NetworkControl.
En la Tabla 3.1 se resumen las capacidades del sistema disponibles.

Capacidades	Privilegios Concedidos
Todos los Archivos	Acceso de lectura al sistema de archivos y acceso de escritura a directorios privados.
CommDD	Acceso directo a todos los controladores de comunicación del equipo.
DiskAdmin	Acceso al sistema de archivos de administración de operaciones que afecta a más de un archivo o directorio (integridad global del sistema de archivos).
Drm	Acceso al contenido protegido por DRM.
Multimedia DD	Acceso a funciones multimedia criticas, como por ejemplo acceso a controladores asociados a un dispositivo o acceso a librerías multimedia.
NetworkControl	Acceso o modificación del control de protocolo de red.
PowerMgmt	Puede matar cualquier proceso, apagar periféricos

	que no se encuentran en uso, enviar a estado de suspensión al dispositivo, encenderlo, o apagarlo completamente.
ProtServ	Permite a un servidor registrar un proceso con un nombre reservado por el sistema.
ReadDeviceData	Acceso de lectura a operadores de red confidenciales, información sobre el fabricante del teléfono móvil y configuración sensible del dispositivo.
SurroundingsDD	Acceso a controladores de dispositivos lógicos que proveen información sobre el entorno del operador móvil.
SwEvent	Cuenta con la habilidad de simular pulsaciones de teclas, inserción de pastillas de memoria y capturar dichos eventos para cualquier programa.
YTrustedUI	Cuenta con la habilidad de crear una sesión de usuario fidedigna, permitiéndole mostrar cuadros de dialogo bajo un entorno aparentemente seguro para el usuario final.
WriteDeviceData	Acceso de escritura a los controles de comportamiento del dispositivo.

Tabla 3.1 Capacidades del Sistema

3.2.3. Capacidades del Usuario

Conjunto reducido de capacidades fáciles de entender, diseñadas significativamente para el usuario final del teléfono móvil; estas capacidades manejan principalmente conceptos de seguridad y toma de decisiones. Como principio general, al asignar una capacidad del tipo usuario a un proceso, se evita que este pueda atentar en contra de la integridad del teléfono móvil y que la instalación por parte del usuario final de una aplicación de terceros no interfiera con el buen desempeño del dispositivo. De todas maneras el usuario debería permanecer atento al comportamiento del software adicional que haya instalado, ya que, por ejemplo, la pérdida de crédito en su plan celular puede ser una señal de que algo no está funcionando correctamente.

Capacidades como NetworkServices o RearUserData pueden ser afectadas por las anormalidades expuestas anteriormente. Las Capacidades del tipo usuario normalmente se conceden a un software no propietario que esté haciendo uso de los servicios prestados por la TCE. Esta es responsable de comprobar y hacer cumplir las capacidades del usuario, y también de realizar los servicios solicitados por el software no propietario que lo requiera.

A pesar de que estas capacidades fueron diseñadas para ser comprensibles al usuario final del teléfono móvil, podrían no ser apropiadas para su uso de manera regular, ya que dependerían del entono en el que se encuentre trabajando el dispositivo. Es conocido que la arquitectura de seguridad del sistema operativo es bastante flexible, por lo cual no se rige por una sola política de seguridad. Por tanto los fabricantes de teléfonos móviles deben manejar con mucha discreción las políticas de seguridad a implementar en sus dispositivos. Cuando las políticas de seguridad mencionadas anteriormente lo permiten, el usuario final de un teléfono móvil, puede autorizar el uso de ciertas capacidades por parte de aplicaciones que no han sido avaladas por una entidad certificante, al momento de ser instalado.

En la Tabla 3.2 se resumen las capacidades del usuario disponibles.

Capacidades	Privilegios Concedidos
LocalServices	Acceso a servicios del tipo Bluetooth o infrarrojos a través de lanzadores del sistema.
Location	Acceso a los datos que almacenan la localización del teléfono móvil.
NetworkServices	Acceso a servicios remotos (como por ejemplo acceso a la red Wi-Fi[13]).
ReadUserData	Acceso de lectura a datos confidenciales del usuario.

UserEnvironment	Acceso a datos sobre el usuario en tiempo real.
NetworkControl	Acceso y modificación de los controles del protocolo de red.
WriteUserData	Acceso de escritura a datos confidenciales del usuario.

Tabla 3.2 Capacidades del Usuario

3.3. Reglas de Funcionamiento de las Capacidades

Al momento en que un desarrollador crea un binario (por ejemplo un .exe o .dll) para la plataforma Symbian OS, este cuenta con un conjunto de capacidades declaradas internamente. Así, un binario puede ser incorporado en un teléfono móvil, sea porque fue incluido previamente por el fabricante en la memoria del dispositivo o porque fue instalado como una aplicación adicional. De cualquier manera, se toma la decisión de que si aquel binario es lo suficientemente confiable para otorgársele las capacidades que ha solicitado. En el caso en que la aplicación haya sido precargada en la memoria del dispositivo, se requiere una verificación manual del proceso de concesión de las capacidades. Cuando una aplicación no propietaria requiera instalación, el proceso de concesión de capacidades se vuelca a un proceso automático manejado directamente por el instalador del software a instalarse.

Una vez que un binario se ha integrado en la memoria del teléfono móvil o un complemento se ha instalado satisfactoriamente, el Sistema Operativo Symbian OS puede asumir que ese binario es lo suficientemente confiable y procede a concederle las capacidades que este las requiera. Sin embargo la concesión de dichas capacidades dependen del tipo de aplicación que las solicite; por ejemplo si es un archivo EXE, este debe ejecutarse con cierto nivel de permisos, con el objetivo de poder llevar a cabo ciertas operaciones privilegiadas. Si se trata de un archivo DLL, las capacidades ya se encuentran declaradas con anterioridad a nivel interno; estas indican el grado de confianza de dicho binario que puede ser utilizado por procesos que corren con

menos privilegio. Resumiendo, un archivo del tipo DLL no siempre se ejecutará con el nivel de capacidades que fue declarado; al contrario, los del tipo EXE necesariamente deben ejecutarse con el nivel de capacidades que se les fue asignado. Para un mejor entendimiento, existe una serie de reglas que un binario debe seguir al momento de su ejecución.

Regla 1: Cada proceso posee un conjunto de capacidades, las cuales no pueden ser cambiadas durante su secuencia de ejecución.

Regla 2: Un proceso puede cargar una DLL, solo si dicha DLL es de confianza y posee por lo menos las mismas capacidades que el proceso requiere.

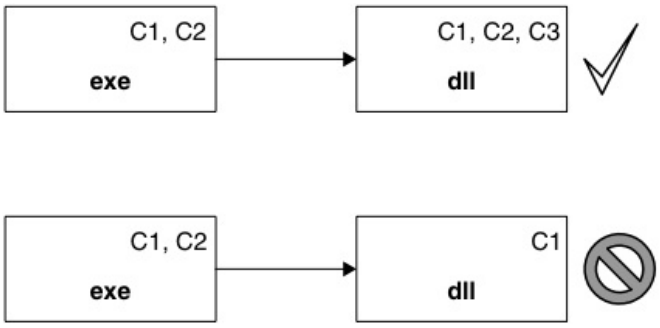


Figura 3.2 Carga directa de DLLs[8]

Un caso especial de carga de DLLs, son los llamados Links Estáticos, estos son utilizados por aplicaciones que necesitan de complementos (plug-ins) que pueden o no estar presentes al momento de su ejecución.

Los Links Estáticos resuelven el problema de referenciamiento de DLLs en tiempo de desarrollo, lo cual permite que su tiempo de carga sea más eficiente. Con respecto a este caso en particular, si una DLL mantiene un Link Estático con una segunda, al momento en que un proceso la requiere, la segunda DLL es cargada en ese

[8] HEATH, *Craig*, Symbian OS Platform Security

simultáneamente. Esto considerando que el primer archivo DLL cuenta con una capacidad que el segundo no la tiene.

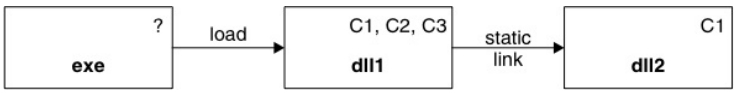


Figura 3.3 Liks Estáticos de DLLs[9]

En este caso como el archivo dll1 mantiene un Link Estático (satatic link) con dll2, dll1 nunca puede ser cargado por un proceso que tenga capacidades del tipo C2, porque este proceso sería incapaz de poder cargar el archivo dll2 (esto por no contar con las mismas capacidades).

Regla 2b: Solo se podrá cargar un archivo DLL que mantenga un Link Estático con un segundo, si y solo si, la segunda DLL es de confianza y cuenta con las mismas capacidades que el primero.

[9] HEATH, *Craig*, Symbian OS Platform Security

Capitulo 4

4. Virología Móvil

4.1. Introducción

La creciente evolución de los teléfonos móviles se encuentra estrechamente relacionada a las necesidades funcionales del usuario final. Debido a la integración funcional con la que hoy cuentan dichos dispositivos, es fácil pensar en el incremento de equipos en el mercado. La necesidad del usuario por mantenerse continuamente comunicado, ha convertido a este dispositivo en un elemento que necesita una alta disponibilidad y calidad en los servicios que brinda. Sin embargo, esta revolución no sólo aporta comodidad y ventajas para el usuario, sino que también lleva consigo una responsabilidad absoluta, al momento de solventar baches de programación de aplicaciones débiles y afrontar problemáticas que deben ser tenidas en cuenta para garantizar óptimos niveles de seguridad.

Este tipo de Virología posee una infraestructura compuesta por la morfología y la taxonomía vírica, y que son a su vez utilizadas en la virología aplicada a los ordenadores personales. Sin embargo, antes de iniciar en el siguiente estudio, es necesario definir las principales familias de virus que componen el panorama actual: [10]

- Virus basado en propagación

  Este tipo de virus utiliza los protocolos de comunicación más conocidos como Bluetooth y Wi-Fi[14] para clonar un teléfono móvil, sin ocasionar daños económicos al usuario ni capturar o alterar su información personal. Caso característico Cabir.

- Virus basado en la modificación del sistema operativo

  Este tipo de virus modifica los archivos del sistema operativo con el

[10] GOSTEV, A, Introducción a la virología móvil. Hakin9.

objetivo de inutilizar funcionalidades específicas o cambiar el aspecto de la interfaz del usuario. Caso característico Skulls

- Virus basado en los daños económicos

  Este tipo de virus utiliza la información personal del usuario, como por ejemplo su lista de contactos, para propagarse a través de MMS, Bluetooth o Wi-Fi, ocasionando así un daño económico al usuario. Caso característico Commwar.

4.2. Morfología Vírica

La morfología vírica forma parte de la virología móvil la cual se ocupa de indagar sobre la estructura de los virus; esta a su vez se encuentra dividida en dos grandes bloques fundamentales que son la morfología externa que es la encargada de definir el comportamiento de aplicaciones maliciosas que se encuentran dentro del teléfono móvil infectado, y la morfología interna la cual describe los detalles de implementación característicos en aplicaciones de este tipo.

En la Figura 4.1 se muestra la estructura principal de los virus en la telefonía móvil, misma que fundamenta los sistemas de clasificación de virus que componen el panorama actual.

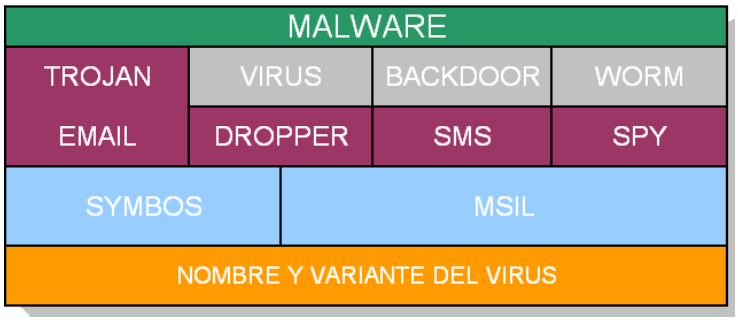


Figura 4.1 Morfología Vírica en la Telefonía Móvil

Este modelo morfológico organizativo muestra a leves rasgos los principales elementos útiles que conforman el nombre de los virus para dispositivos móviles. Primero, se identifica el malware correspondiente a un tipo de virus. Segundo, se debe indicar el sistema operativo afectado

por el virus anteriormente seleccionado. Y finalmente, se especifica una variante distinta si el comportamiento o el código fuente del virus es particular con respecto a sus antecesores.

Además, este modelo basado en capas posee una serie de ventajas e inconvenientes que se detallan a continuación:

- Facilidad de Visualización.- Esta estructura favorece la visualización de los virus dentro de las listas de clasificación de los antivirus actuales.
- Escalabilidad.- Esta organización favorece la escalabilidad de la arquitectura, permitiendo la inclusión de nuevos módulos sin alterar el comportamiento global del sistema de clasificación.

Como se comentó anteriormente la morfología externa estudia el comportamiento seguido por los virus tras su instalación en el teléfono móvil infectado. Dicho análisis sigue ciertos patrones de clasificación y categorización de factores como el conocer la familia del virus, el protocolo o esquema de propagación, y las variantes del virus. Detallando más a fondo esta categorización, la familia del virus identifica la tipología o el comportamiento externo seguido por un virus que pertenezca a esta categorización. El protocolo y esquema de propagación definen respectivamente el medio de comunicación utilizado para su propagación y el árbol de directorio del virus. Por último, la variante determina la alteración, mutación o variación del virus respecto al modelo original identificado por la familia a la cual pertenecía. Por ejemplo:

Esquema de Propagación del virus Cabir.
Nombre: *Cabir*
Familia de Virus: *Propagación*
Protocolo de Propagación: *Bluetooth*
Variante: *B*
Esquema de Propagación: + *System* → +*apps* → *oidi500*

La morfología interna define los detalles de implementación necesarios para la creación de un virus en una plataforma móvil. Entre las características esenciales de este tipo de morfología se encuentran el lenguaje de programación y el protocolo de comunicación. Debido al problema de la fragmentación móvil, cada lenguaje de programación se encuentra fuertemente unido a un sistema operativo concreto, lo que limita, pero no elimina, las posibilidades de propagación entre plataformas. En este caso en particular, Symbian OS fue programado en C++.

El protocolo de comunicación es uno de los principales factores que influyen en la morfología interna de un virus. Este componente depende directamente de las interfaces de comunicación proporcionadas por el sistema operativo, por lo cual cada caso de propagación varía en función de la plataforma utilizada.

4.3. Taxonomía Vírica

Quien define los principios, métodos y fines de clasificación de virus es la taxonomía vírica, concepto que es ampliamente utilizado por los sistemas de antivirus para detectar nuevas o existentes amenazas para los usuarios de dispositivos móviles. Uno de los principales retos de las compañías de antivirus es la creación y mantenimiento de sus sistemas de detección de intrusos, el cual potencialmente debe realizar una robusta clasificación con la cual se permita determinar la familia del nuevo virus con cierta eficiencia y eficacia. En el 2004, apareció la primera clasificación mediante las siguientes tres tendencias[11]:

- Amenazas basadas en pérdidas financieras.
- Amenazas basadas en la modificación del sistema operativo.
- Amenazas basadas en la propagación.

Una notable evolución en las metodologías de clasificación se dio

[11] GOSTEV, A., Introducción a la virología móvil. Hakin9.

gracias a sus potentes sistemas, los cuales se basaron en los siguientes principios para su desarrollo:

- Conducta o comportamiento del virus.
- Entorno de ejecución.
- Nombre de familia y letra de la variante.

Sin embargo, y a pesar de esta gigantesca evolución, no se llegó a resolver de manera eficaz el problema de la hibridación, la cual consistía en la mezcla entre familias de virus ya existentes, lo cual dificultaba el proceso de detección, documentación y tratamiento de cada nueva variante de un virus.

La Figura 4.2 muestra el modelo de representación taxonómica en una forma más detallada, dicha participación cubre la mayoría de las etapas necesarias para efectuar ataques estudiados en la virología existentes de la seguridad informática.

Figura 4.2 Taxonomía Vírica en la Telefonía Móvil

El presente gráfico está compuesto por cinco tipos de características, cada una con sus correspondientes variantes, “descubrimiento de dispositivos”, “transmisión al dispositivo”, “activación”, “payloads” y “motivación del atacante”. El principal objetivo de este nuevo esquema de representación y clasificación es la reducción del impacto producido

por factores como la fragmentación móvil (multiplataformas) y la hibridación (mezcla de familias).

4.4. Familias y Modificaciones

La popularidad de los teléfonos inteligentes y el continuo crecimiento de novedosas aplicaciones y servicios traen consigo el crecimiento de responsabilidades por parte de los integrantes de la cadena vírica que se mencionó anteriormente. Hacia mediados de agosto de 2009 se registraron 106 familias y 514 modificaciones de programas maliciosos para dispositivos móviles. Como se puede apreciar en la Figura 4.3 para finales de 2010 las cifras cambiaron significativamente a 153 familias y más de 1000 modificaciones. Es decir, en el 2010 se detectaron un 65,12% más programas maliciosos para dispositivos móviles que en 2009 y casi el doble que 17 meses antes. [12]

Basándose en las estadísticas y proyecciones del 2010 con respecto a la virología móvil; la situación en la que se encuentra Symbian es la siguiente:

Plataforma	Cantidad de Familias	Cantidad de Modificaciones
J2ME	45	613
Symbian	74	311
Python	5	60
Windows Móvil	16	54
Android OS	7	15
Sgold	3	4
MSIL	2	4
IphoneOS	1	2

Tabla 4.1 Cantidad de Familias y Modificaciones según su SO

[12] http://www.viruslist.com/sp/analysis?pubid=207271120#2

Cabe destacar que el foco de ataque se encuentra bastante bien sectorizado hacia sistemas basados en la tecnología J2ME[15] y Symbian. Recordemos que las aplicaciones maliciosas representan peligro no solo para los usuarios de teléfonos inteligentes, sino para casi todos los usuarios de teléfonos celulares comunes y corrientes. En su mayoría, estos programas maliciosos tratan de enviar mensajes SMS a un número corto, incurriendo en un costo no contemplado para el usuario del equipo.

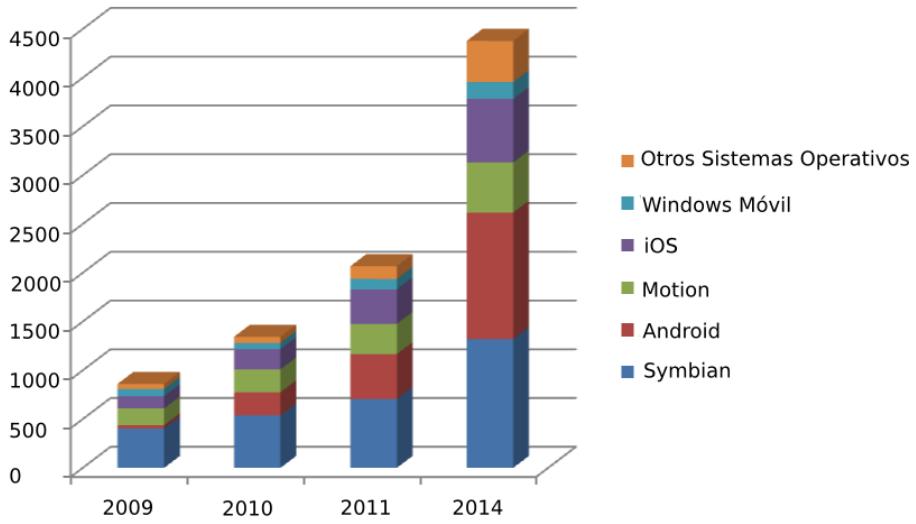


Figura 4.3 Distribución de Familias y Modificaciones según su SO

Aplicaciones maliciosas encontradas para Symbian OS entre el 2009 y el 2010.

Familia	Fecha de detección
Trojan-SMS.Lopsoy	octubre 2009
Trojan-SMS.BadAssist	noviembre 2009
not-a-virus:Monitor.Flesp	diciembre 2009
not-a-virus:Monitor.Dadsey	diciembre 2009
Worm.Megoro	marzo 2010
not-a-virus:Monitor.Bond006	junio 2010
Worm.Sagasi	junio 2010
Trojan-Spy.Reples	junio 2010

Trojan-Spy.Zbot	septiembre 2010
Worm.Nmplug	noviembre 2010
Trojan-Downloader.Minplay	diciembre 2010

Tabla 4.2 Principales aplicaciones maliciosas que atacan a Symbian

4.5. Malware Móvil

Alrededor de los últimos años, se ha generado cierto tipo de especulación al respecto de la proliferación de malware en dispositivos móviles inteligentes. Sin embargo en este lapso de tiempo cierto tipo de desarrollos concernientes a aplicaciones maliciosas han ido apareciendo en una forma desmedida.

4.5.1. Ciclo de vida del Malware Móvil

Hace un par de años atrás eran ya conocidas ciertas aplicaciones maliciosas que se expandían entre usuarios celulares sea vía Bluetooth, MMS o SMS, mismos que infectaban librerías gráficas del sistema operativo, modificaban o reemplazaban iconos, accedían a la información de tarjetas de memoria entre otras. Sin embargo hoy en día los cybercriminales cuentan con nuevas técnicas como la de DoS[16] (del inglés Denial of Service), deshabilitar el sistema operativo, descargar vía Internet archivos corruptos en forma transparente para el usuario final, realizar llamadas silenciosas a números con cargos internacionales, infectar las tarjetas de expansión del dispositivo, e incluso hurtar las credenciales del usuario de su cuenta de banca móvil.

4.5.2. Vectores de Ataque

Quienes se dedican al desarrollo de aplicaciones maliciosas para dispositivos móviles tiene que lidiar con el cómo hacer llegar su código efectivo a los distintos sistemas operativos móviles que existen hoy en día. Al contrario de los sistemas operativos especializados para PC en donde la gran mayoría de usuarios cuentan con Windows como su sistema operativo, en el mundo de los dispositivos móviles

se cuenta con Symbian, Apple[17], BlackBerry[18], Android[19], Windows Móvil 7[20] por nombrar algunos. En la Figura 4.6, se muestra la distribución de amenazas según su sistema operativo. Sin embargo esta distribución tiende a rendir a futuro un brusco cambio, partiendo de la unión estratégica de Microsoft y Nokia, y por otro lado la fuerte aceptación de Android en el mercado actual.

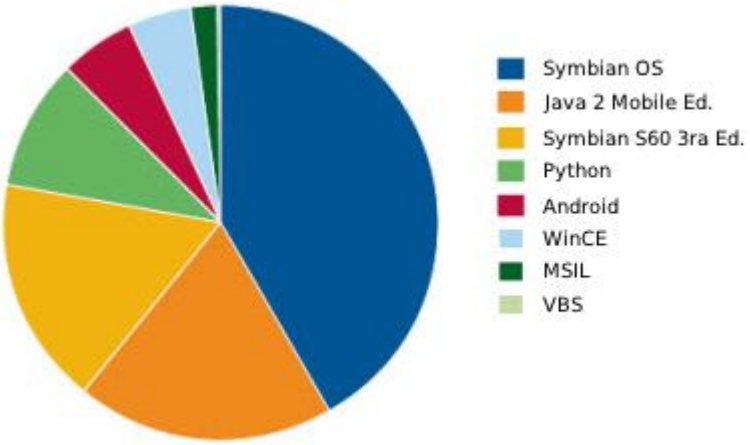


Figura 4.4 Amenazas a Sistemas Operativos Móviles[13]

En años pasados sin lugar a duda el objetivo principal de ataques fue Symbian S60. Se podría decir que su principal debilidad fue su usuario final, esto debido a que aplicaciones maliciosas eran instaladas ingenuamente por los mismos dueños del dispositivo móvil. La técnica utilizada por los atacantes era desarrollar aplicaciones llamativas al usuario final, pero incorporando en sus líneas de código eventos que trabajaban por debajo de la aplicación principal, cosa que era totalmente transparente para el usuario, el cual caía en cuenta de la intromisión al momento de pagar sus cuentas y encontrarse con que se habían hecho llamadas a números por cobrar o de larga distancia sin su consentimiento.

En el 2009 los llamados servidores remotos de Nokia (hoy conocidos como servidores de descarga OVI), fueron vulnerados por hackers, quienes incorporaron aplicaciones maliciosas en dichos

[13] McAfee Threats Report Q4 2010

equipos, dando paso a una proliferación desmedida de troyanos y aplicaciones no confiables. Otro vector de ataque fue conocido como MRM (del inglés Mobile Ready Malware). Este ataque consistía en contar con un Malware residente en el dispositivo móvil, el cual era activado e incluso actualizado vía remota por un servidor, sin que el usuario tenga conocimiento.

MRM tenía un funcionamiento muy similar a una botnet[21], la idea es mantener dispositivos móviles conectados remotamente a un servidor, el cual le envía Malware nuevo para ser distribuido deliberadamente entre los contactos de la víctima vía SMS o MMS, por citar un ejemplo.

Con respecto al ataque del tipo DoS, este lo que hace es saturar la red telefónica inundándola con envío masivo de paquetes.

Como se mencionó anteriormente, la técnica más utilizada por quienes se dedican a desarrollar Malware, es esconder cualquier tipo de aplicación maliciosa dentro de un aplicación legítima. Esta técnica permite a la aplicación maliciosa trabajar silenciosamente y sin problema alguno sin que el usuario se imagine lo que está pasando. Otro truco bastante usual es deshabilitar el certificado de una aplicación confiable para ser utilizado por una no legítima.

4.5.3. Caso de Ataque a Symbian [14]

Nombre: Zitmo (Zeus In The MObile)

Familia: SymbOS/Zitmo.A!tr

Fecha: 07 marzo 2011

Vector de Ataque: Inyección a formularios HTML vía SMS

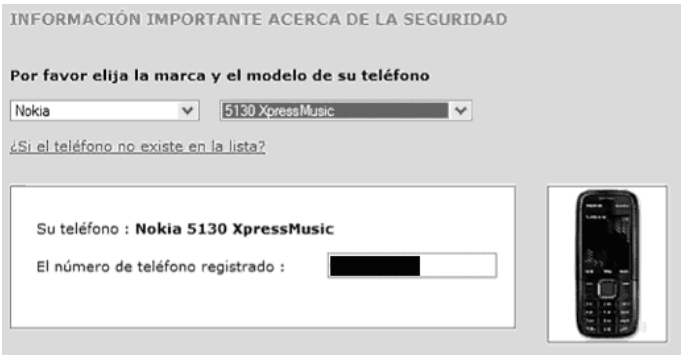
Zitmo es una variante de Zeus, el cual ataca a dispositivos móviles que utilizan Symbian como su sistema operativo; siendo un poco más específico Zitmo ataca aplicaciones del tipo Banca Móvil que se encuentren instaladas, inyectando código HTML en sus formularios.

[14] http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-i.html

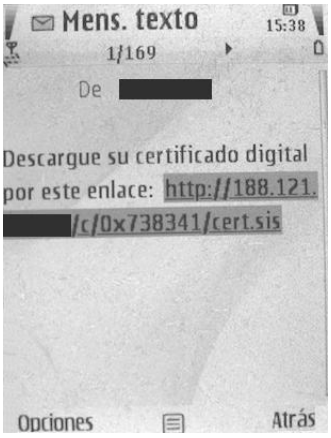
En primer lugar el virus recolecta el número telefónico y el modelo del dispositivo móvil. Este envía a la víctima por medio de un mensaje SMS, una URL en donde se encuentra la aplicación maliciosa. La idea de funcionamiento es simple, muchas de las entidades bancarias se manejan con factores de autenticación, como por ejemplo claves que funcionan por una única vez (OTP, por sus siglas en inglés), las cuales son enviadas al usuario celular vía mensaje SMS con un tiempo de expiración. Zitmo intercepta la clave de ingreso reenviando todos los mensajes SMS a un servidor remoto.

4.5.3.1. Esquema del Ataque

1. Zitmo inyecta código HTML en la aplicación banca movil, en la cual le solicita al usuario ingrese ciertos datos sobre el equipo desde el cual esta accediendo.



2. Una vez que averigua el número celular de la víctima, el delincuente envía un mensaje SMS indicando un link en donde supuestamente se encuentra el certificado digital que debe ser instalado.



3. Si el usuario sigue el enlace, se le propone instalar una aplicación. Este puede instalarla ejecutando así el Malware o simplemente no aceptarla, siendo así fallido el intento de intromisión por parte del virus.
4. Si es efectiva la instalación de la aplicación mencionada anteriormente, el delincuente trata de hacer una operación online en el banco que requiere enviar una confirmación por SMS.
5. El banco envía un mensaje SMS con el código de autenticación al número de teléfono de la víctima.
6. El programa malicioso envía dicho código por mensaje SMS al número de teléfono del delincuente.
7. El delincuente recibe el código de autenticación y completa la operación online en el banco.

"La complejidad del ataque confirma que los intereses de los delincuentes están en constante expansión. Hasta el descubrimiento de este programa malicioso, la autenticación por SMS era uno de los métodos más seguros de protección de las operaciones bancarias online. Ahora los delincuentes han aprendido a burlar este nivel de defensa"[15]

[15] Kaspersky Lab ZAO. 2011

Capitulo 5

5. Análisis de Robustez, Seguridad y Estabilidad [16]

5.1. Introducción

La seguridad en dispositivos móviles es un tema relativamente nuevo, el cual requiere de una atención especial tanto por los diseñadores del dispositivo, desarrolladores de aplicaciones, y por quienes se encargan de gestionar la funcionalidad de la red celular.

En este análisis se efectuaron varias pruebas bajo factores estándar de desempeño de dispositivos que trabajan con Symbian como Sistema Operativo. La idea es evaluar posibles rupturas en la arquitectura de seguridad del sistema operativo en cuestión.

Al momento de evaluar la seguridad de un dispositivo, no solo se deben estimar ataques directos, sino también considerar que este puede ser utilizado como puente para conectarse a un sistema que se encuentra en otra red. Es decir, se podría obtener acceso a redes corporativas desde un dispositivo que se encuentra comprometido.

En la siguiente sección se mostraran los resultados de dicho análisis, basado en el escaneo de vulnerabilidades, de red, y un test de penetración a un dispositivo que utiliza Symbian versión 9.2 como Sistema Operativo (Nokia E71).

5.1.1. Test de Penetración

Con respecto a los sistemas operativos de equipos de escritorio es usual el realizar pruebas de robustez o simplemente testearlos en busca de posibles vulnerabilidades. Por otro lado, a manera de comparación, los teléfonos inteligentes al ser una tecnología algo nueva, están relativamente libres de experiencias sobre ataques o vulnerabilidades que los precedan; sin embargo si cuentan con fallos

[16] Sheikh Mahbub Habib, Cyril Jacob y Tomas Olovsson
An Analysis of the Robustness and Stability of the Network Stack in Symbian-based Smartphones

de operatividad, limitaciones en espacio y memoria, comportamientos que están casi mitigados en su totalidad en el mundo de las PC´s.

Al tratarse de un test que busca encontrar vulnerabilidades en Symbian, pero al no contar con un relevamiento anterior, se toma como base el realizar pruebas que afectan a varios sistemas operativos convencionales y a sus protocolos de comunicación. Dicho test se lo realiza con un cliente que corre Linux Ubuntu 11.04 Unity y como víctima un Nokia E71 con Symbian OS 9.2

A. Escaneo de Red

Gracias a las prestaciones del utilitario para exploración de redes y auditoria de seguridad Nmap[22] (¨Network Mapper¨), fue posible el obtener tanto la dirección MAC e IP del dispositivo celular. Así aplicando técnicas de descubrimiento (fingerprinting), se pudo encontrar puertos abiertos, filtrados y cerrados en el equipo. Esta técnica es utilizada para revelar los servicios que se encuentran habilitados en el dispositivo, burlando incluso mecanismos de seguridad del tipo firewall. Puntualmente, Nmap obtuvo como resultado información sensible relacionada a la arquitectura interna del dispositivo, muchos de los servicios que se encontraban ejecutándose en el sistema, por ejemplo *Symbian service broker* (symbian-sb port 3389), el cual se utiliza para determinar la versión del sistema operativo instalado en el dispositivo móvil, el número IMEI y los principales servicios con sus puertos correspondientes.
Un extracto de los resultados obtenidos se muestran en la Tabla 5.1

Prueba	Protocolo	Comando	Resultado
Descubrimiento del Host Escaneo por ping	ICMP	# nmap -sP 10.10.0.1-254	Ya que el Router (Cisco- Linksys) se encuentra en modo DHCP, se puede obtener una dirección IP en el mismo nodo de la víctima. **Se encontró (Nokia Danmark A/S), su dirección MAC e IP**.
Escaneo completo a	TCP	# nmap -PE -v -p1-65535 -	Todos los puertos escaneados se encontraban cerrados.

todos los puertos TCP		PA21,23,80,3389 -A -T4 10.10.0.10	
Escaneo completo avanzado UDP	UDP	# nmap -PE -v -PA21,23,80,3389 -sU -A -T4 10.10.0.10	Todos los puertos escaneados se encontraban cerrados.
Escaneo Descubrimiento Sistema Operativo (fingerprinting)	TCP, UDP	# nmap -A -O -v -d 10.10.0.10	**Mac Address: 00:21:FE:36:75:0A (Nokia Danmark A/S)** Device type: switch general; purpose phone Running: Bay Networks embedded, IBM i5/OS V5|V6, **Nokia Symbian OS 9.X|10.X**, Sony Ericsson embedded, Sony Ericsson Symbian OS 9.X OS details: Bay Networks BayStack 450 switch (software version 3.1.0.22), Bay Networks BayStack 450 switch (software version 4.2.0.16), IBM i5/OS V5R4, IBM i5/OS V5R4 on an IBM iSeries (PPC), IBM i5/OS V6R1, **Nokia E51, E90 Communicator, or N95 mobile phone (Symbian OS 9.2 - 10.0), Nokia E60, E61, E65, or E70 mobile phone (Symbian OS), Nokia E70 mobile phone (Symbian OS)**, Sony Ericsson G900 mobile phone, Sony Ericsson M600i mobile phone (Symbian OS 9.1), Sony Ericsson P1i mobile phone (Symbian OS 9.1).

Tabla 5.1 Resultados del escaneo de red*

B. Escaneo de Vulnerabilidades

Para propiciar un escaneo de las potenciales vulnerabilidades que presenta el dispositivo, fue necesaria la incursión de Nessus[23], una potente aplicación que permite realizar un exhaustivo escaneo de vulnerabilidades con el objetivo de encontrar que puertos y servicios

* Para mayor visualización de los resultados obtenidos verificar tabla de Anexos (Escaneo Red).

están ejecutándose en el dispositivo. Una pequeña reseña de los resultados se los puede observar en la Tabla 5.2.

En resumen Nessus no reportó vulnerabilidades significativas debiéndose a que el dispositivo bajo su configuración por defecto trae todos sus puertos de red cerrados, aunque si aporto con información que podría ser utilizada para propiciar un ataque de forma mas dirigida y efectiva.

Puerto / Protocolo	**Prueba**	**Descripción**	**Riesgo/ID**
General/TCP	Detección tarjeta de red del fabricante.	Se llega a deducir el fabricante del dispositivo a partir del OUI (Organizationally Unique Identifier)	**Bajo 35716**
General/TCP	Escaneo de información Nessus	Se genera información con respecto a: - La versión del plugin - El tipo de plugin - La versión del motor de Nessus - El rango de puertos escaneados - La fecha del escaneo - El tiempo de duración del escaneo - El numero de hosts escaneados	**Bajo 19506**

Tabla 5.2 Resultados del escaneo de vulnerabilidades*

C. Test de Penetración

Para esta fase de pruebas de penetración, fue primario la ejecución de ciertos ataques dirigidos al objetivo (Nokia E71) haciendo foco en el protocolo TCP/IP[24]. Los resultados se muestran a continuación.

C.1. ARP spoofing

Se pudo observar que el dispositivo es vulnerable a un ataque del tipo DoS, al momento que recibe sin petición previa paquetes ARP de su dirección IP.

* Para mayor visualización de los resultados obtenidos verificar tabla de Anexos (Escaneo Vulnerabilidades).

Su ejecución se la puede realizar de dos formas. La primera es utilizando la IP del dispositivo pero vinculada a otra MAC, provocando así un conflicto de direcciones de red. La segunda es utilizando tanto la IP como la MAC del dispositivo, con la intención de verificar que tan probable es generar confusión al dispositivo objetivo. Durante la realización del test también se pudo ejecutar un sniffing del trafico de red al momento en que se comunican el dispositivo objetivo y el router al cual está conectado. Valiéndose de las capturas del trafico anteriores fue factible el reenvío de paquetes y peticiones ARP ficticias simulando un ataque MiTM[25] (del ingles Man in The Middle).

El procedimiento se efectuó utilizando la herramienta arpspoof de dsniff. El test se lo realizó bajo una plataforma Linux Ubuntu 11.04 *Unity* (atacante) 10.10.0.106 y un Nokia E71 equipado con Symbian 9.2 (víctima) 10.10.0.10. los resultados se muestran en la Tabla 5.3.

Prueba	**Capa de red**	**Comando del ataque**	**Resultado**
TCP SYN flood	Transporte	# hping3 --flood --syn --rand-source 10.10.0.10 # hping3 --flood --syn --destport 3923 --rand-source 10.10.0.10	Durante la inundación de paquetes dirigidos al equipo (víctima), este logra establecer una precaria comunicación, aunque destacando el hecho de que no llego a presentar posibles estados de denegación de servicio DoS.
ARP spoofing	Enlace	# arpspoof 10.10.0.10	Se observa un ataque del tipo DoS. En este caso el dispositivo víctima detecta un conflicto de red, por lo cual solicita por repetidas ocasiones se le asigne una nueva dirección IP.
Petición ARP	Enlace	# packit -t arp -A 1 -x 10.10.0.106 -X	La intercepción de tráfico entre el router (Cisco) y la

spoofing		00:1F:3C:6F:3E:8E -y 10.10.0.10 -Y 00:21:FE:36:75:0A -e 00:1D:72:5E:FD:6F -E 00:21:FE:36:75:0A -i wlan0	víctima, da como resultado un ataque del tipo DoS. En el dispositivo víctima no se muestra ningún mensaje de advertencia de lo que sucede.

Tabla 5.3 Resultados del test de penetración*

C.2. Ataque TCP SYN flooding

Para gestionar la ejecución del ataque TCP SYN flooding fue necesario utilizar la herramienta Hping3 a manera de ensamblador de paquetes; dicho ataque se basa en el uso de entornos WLAN a través de plataformas Linux. Sus resultados se muestran en la Tabla 5.3.

A manera de experiencia, a pesar de que el dispositivo móvil nunca dejo de comunicarse con el entorno durante la ejecución del ataque, el equipo se volvió muy lento y no responsivo; resaltando así su apropiada auto defensa contra el ataque.

* Para mayor visualización de los resultados obtenidos verificar tabla de Anexos (Test Penetración).

Conclusiones

Muchos de los problemas con los cuales nos encontramos habitualmente con nuestras computadoras de escritorio, han empezado a aparecer en los dispositivos móviles. Sin embargo, la arquitectura tanto de un equipo como de otro, cambian enormemente (por ejemplo cantidad de memoria, velocidad de procesamiento), por lo cual a efectos de implementación, la seguridad se convierte en todo un reto. La infraestructura funcional que hoy en día ha llegado también a compartirse con los dispositivos móviles (IEEE 802.11), se convierte en uno de los principales focos a *securizar* no solo por los desarrolladores del sistema operativo como tal, si no armar una cadena de valor de la seguridad en donde se involucren tanto a los fabricantes de los dispositivos, desarrolladores de aplicaciones de terceros, y los usuarios finales. Esto con el objetivo de enfrentar una problemática que con el tiempo llegara a ser más que común en el medio.

El trabajo presentado en esta tesina se enfoca especialmente en estudiar los fallos de seguridad, el malware, y las vulnerabilidades que presenta el Sistema Operativo Symbian para dispositivos móviles. La investigación de esta problemática, permitió demostrar la existencia de malware operativo en la plataforma, además de que efectivamente, es posible valerse de las vulnerabilidades presentes en Symbian para generar estados de denegación de servicio DoS, hurtar credenciales o información confidencial almacenada en el dispositivo por medio de malware o simplemente provocar cierta inestabilidad en el equipo.

Al momento en que se realizaron las distintas pruebas específicas en los test de penetración, se pudo observar que efectivamente las vulnerabilidades que presenta esta plataforma sumados a los conocidos riesgos que conlleva el uso del protocolo TCP/IP sin cifrar, son un blanco fácil para perpetrar ataques, los cuales son usualmente efectuados en medios no seguros como Internet, en donde es común que equipos móviles sean continuamente escaneados y a posteriori presenten fallos del sistema.

Se espera que este trabajo sirva principalmente para entender cuál es el riesgo de utilizar o no un teléfono inteligente en un entorno potencialmente inseguro y en segundo plano como una ayuda tanto para vendedores, organizaciones, o usuarios interesados, que necesiten conocer el nivel de seguridad y estabilidad con el que cuenta un dispositivo equipado con Symbian como su Sistema Operativo.

Anexos

A. Escaneo Red

Descubrimiento del Host Escaneo por ping

luchitss@dv2000:~$ sudo nmap -sP 10.10.0.1-254

Starting Nmap 5.21 (http://nmap.org) at DATE TIME ART

Nmap scan report for 10.10.0.1

Host is up (0.057s latency).

MAC Address: 00:21:29:7D:39:34 (Cisco-Linksys)

Nmap scan report for 10.10.0.10

Host is up (0.050s latency).

MAC Address: 00:21:FE:36:75:0A (Nokia Danmark A/S)

Nmap scan report for 10.10.0.101

Host is up (0.065s latency).

MAC Address: 00:25:9C:76:4C:0F (Cisco-Linksys)

Nmap scan report for 10.10.0.104

Host is up (0.065s latency).

MAC Address: 00:26:C7:FB:1E:6E (Intel Corporate)

Nmap scan report for 10.10.0.106

Host is up.

Nmap done: 254 IP addresses (5 hosts up) scanned in 5.94 seconds

Descubrimiento del Sistema Operativo

luchitss@dv2000:~$ sudo nmap -A -O -v -d 10.10.0.10

Starting Nmap 5.21 (http://nmap.org) at 2011-07-11 21:09 ART

PORTS: Using top 1000 ports found open (TCP:1000, UDP:0, SCTP:0)

---------- Timing report ----------

hostgroups: min 1, max 100000

rtt-timeouts: init 1000, min 100, max 10000

max-scan-delay: TCP 1000, UDP 1000, SCTP 1000

parallelism: min 0, max 0

max-retries: 10, host-timeout: 0

min-rate: 0, max-rate: 0

NSE: Loaded 36 scripts for scanning.

Initiating ARP Ping Scan at 21:09

Scanning 10.10.0.10 [1 port]

Packet capture filter (device wlan0): arp and arp[18:4] = 0x001F3C6F and arp[22:2] = 0x3E8E

Completed ARP Ping Scan at 21:09, 0.17s elapsed (1 total hosts)

Overall sending rates: 11.78 packets / s, 494.94 bytes / s.
mass_rdns: Using DNS server 208.67.222.222
Initiating Parallel DNS resolution of 1 host. at 21:09
mass_rdns: 0.57s 0/1 [#: 1, OK: 0, NX: 0, DR: 0, SF: 0, TR: 1]
Completed Parallel DNS resolution of 1 host. at 21:09, 0.57s elapsed
DNS resolution of 1 IPs took 0.57s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 21:09
Scanning 10.10.0.10 [1000 ports]
Packet capture filter (device wlan0): dst host 10.10.0.106 and (icmp or ((tcp or udp or sctp) and (src host 10.10.0.10)))
Increased max_successful_tryno for 10.10.0.10 to 1 (packet drop)
Increased max_successful_tryno for 10.10.0.10 to 2 (packet drop)
Increased max_successful_tryno for 10.10.0.10 to 3 (packet drop)
Completed SYN Stealth Scan at 21:09, 6.01s elapsed (1000 total ports)
Overall sending rates: 176.29 packets / s, 7756.64 bytes / s.
Initiating Service scan at 21:09
Packet capture filter (device wlan0): dst host 10.10.0.106 and (icmp or (tcp and (src host 10.10.0.10)))
Initiating OS detection (try #1) against 10.10.0.10
NSE: Script scanning 10.10.0.10.
NSE: Script Scanning completed.
Nmap scan report for 10.10.0.10
Host is up, received arp-response (0.018s latency).
All 1000 scanned ports on 10.10.0.10 are closed because of 1000 resets

MAC Address: 00:21:FE:36:75:0A (Nokia Danmark A/S)

Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: switch|general purpose|phone

Running: Bay Networks embedded, IBM i5/OS V5|V6, Nokia Symbian OS 9.X|10.X, Sony Ericsson embedded, Sony Ericsson Symbian OS 9.X

OS details: Bay Networks BayStack 450 switch (software version 3.1.0.22), Bay Networks BayStack 450 switch (software version 4.2.0.16), IBM i5/OS V5R4, IBM i5/OS V5R4 on an IBM iSeries (PPC), IBM i5/OS V6R1, Nokia E51, E90 Communicator, or N95 mobile phone (Symbian OS 9.2 - 10.0), Nokia E60, E61, E65, or E70 mobile phone (Symbian OS), Nokia E70 mobile phone (Symbian OS), Sony Ericsson G900 mobile phone, Sony Ericsson M600i mobile phone (Symbian OS 9.1), Sony Ericsson P1i mobile phone (Symbian OS 9.1)

TCP/IP fingerprint:

OS:SCAN(V=5.21%D=7/11%OT=%CT=1%CU=35583%PV=Y%DS=1%DC=D%G=N%M=0021FE%TM=4E1B
OS:90B6%P=i686-pc-linux-gnu)SEQ(CI=I%II=I)T5(R=Y%DF=N%T=45%W=0%S=Z%A=S+%F=A
OS:R%O=%RD=0%Q=)T6(R=Y%DF=N%T=45%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=N%T=4
OS:5%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=45%IPL=38%UN=0%RIPL=G%RID=G
OS:%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=45%CD=S)
Network Distance: 1 hop
HOP RTT ADDRESS
1 17.64 ms 10.10.0.10
Final times for host: srtt: 17638 rttvar: 13698 to: 100000
Read from /usr/share/nmap: nmap-mac-prefixes nmap-os-db nmap-service-probes nmap-services.
OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.21 seconds
Raw packets sent: 1068 (47.558KB) | Rcvd: 1031 (41.504KB)

B. Escaneo Vulnerabilidades

Detección tarjeta de red del fabricante

Plugin ID: 35716 **Port / Service:** general/tcp
Plugin Name: Ethernet Card Manufacturer Detection

Synopsis: The manufacturer can be deduced from the Ethernet OUI.

Description
Each ethernet MAC address starts with a 24-bit 'Organizationally Unique Identifier'.
These OUI are registered by IEEE.

Solution
n/a

See Also
http://standards.ieee.org/faqs/OUI.html
http://standards.ieee.org/regauth/oui/index.shtml

Risk Factor: None

Plugin Output
The following card manufacturers were identified :

00:21:fe:36:75:0a : Nokia Danmark A/S

Plugin Publication Date: 2009/02/19

Plugin Last Modification Date: 2011/03/27

Escaneo de Información

Plugin ID: 19506 **Port / Service:** general/tcp
Plugin Name: Nessus Scan Information

Synopsis: Information about the Nessus scan.

Description
This script displays, for each tested host, information about the scan itself:

- The version of the plugin set
- The type of plugin feed (HomeFeed or ProfessionalFeed)
- The version of the Nessus Engine
- The port scanner(s) used
- The port range scanned
- The date of the scan
- The duration of the scan
- The number of hosts scanned in parallel
- The number of checks done in parallel

Solution
n/a

Risk Factor: None

Plugin Output
Information about this scan :

Nessus version : 4.4.1
Plugin feed version : 201106172235
Type of plugin feed : HomeFeed (Non-commercial use only)
Scanner IP : 10.10.0.106

C. Test Penetración

ARP spoofing

luchitss@dv2000:~$ sudo arpspoof 10.10.0.10

0:1d:72:5e:fd:6f ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.0.10 is-at 0:1d:72:5e:fd:6f

0:1d:72:5e:fd:6f ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.0.10 is-at 0:1d:72:5e:fd:6f

0:1d:72:5e:fd:6f ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.0.10 is-at 0:1d:72:5e:fd:6f

0:1d:72:5e:fd:6f ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.0.10 is-at 0:1d:72:5e:fd:6f

0:1d:72:5e:fd:6f ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.0.10 is-at 0:1d:72:5e:fd:6f

Petición de ARP spoofing

luchitss@dv2000:~$ sudo packit -t arp -A 1 -x 10.10.0.106 -X 00:1F:3C:6F:3E:8E -y 10.10.0.10 -Y 00:21:FE:36:75:0A -e 00:1D:72:5E:FD:6F -E 00:21:FE:36:75:0A -i wlan0

Mode: Packet Injection using device: wlan0

ARP header: Type: Request(1)

Sender: Protocol Address: 10.10.0.106 Hardware Address: 0:1F:3C:6F:3E:8E

Target: Protocol Address: 10.10.0.10 Hardware Address: 0:21:FE:36:75:A

Eth header: Src Address: 0:1D:72:5E:FD:6F Dst Address: 0:21:FE:36:75:A

Writing packet(s) (1): .

-| Packet Injection Statistics |---

Injected: 1 Packets/Sec: 1.0 Bytes/Sec: 42.0 Errors: 0

Bibliografía

- HEATH, *Craig*, Symbian OS Platform Security, Symbian Press, Inglaterra, 2009

- JIPPING, *Michael,* Smartphone Operating System Concepts with Symbian OS, John Wiley & Sons, Inglaterra, 2007

- WILCOX, *Mark*, Porting to the Symbian Platform, John Wiley & Sons, Reino Unido, 2009

- TANENBAUM, *Andrew*, Sistemas Operativos Modernos, 3ra edición, Pearson Educación, México, 2009

- SALES, *Jane*, Symbian OS Internals, John Wiley & Sons, Inglaterra, 2005

- RETAMOSA, *Germán,* Sistema de Detección de Intrusiones Bluetooth para Terminales Móviles, España 2009

- HABBIB, *Sheikh;* JACOB, *Cyril;* OLOVSSON, *Tomas,* An Analysis of the Robustness and Stability of the Network Stack in Symbian-based Smartphones, Departamento de Ingeniería y Ciencias de la Computación, Universidad Tecnológica de Chalmers, Suecia 2009

- http://library.forum.nokia.com/index.jsp?topic=/S60_5th_Edition_Cpp_Developers_Library/GUID-35228542-8C95-4849-A73F-2B4F082F0C44/sdk/doc_source/guide/platsecsdk/SGL.SM0007.013_Rev2.0_Symbian_OS_Security_Architecture.doc.html documentación que abarca los temas de arquitectura y plataforma de seguridad de Symbian OS v9. Accedido en julio 2011

- http://www.viruslist.com/sp/analysis?pubid=207271120 documentación que trata sobre la clasificación y tendencias de los malwere de telefonía celular. Accedido julio 2011

- http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-i.html blog que trata sobre seguridad de la información. Accedido em julio 2011

- http://nmap.org/ , Nmap security scanner v5.50. (Aplicación utilizada para exploración y auditoría de redes). Accedido en julio 2011

- http://www.nessus.org/nessus/ , Nessus Vulnerability Scanner v4.4.1. (Aplicación Web para escanear vulnerabilidades) Accedido en julio 2011

- http://monkey.org/~dugsong/dsniff/ , Dug Song, Dsniff 2.3. (Herramienta para realizar pruebas de penetración de red). Accedido en julio 2011

- http://packetfactory.openwall.net/projects/packit/index.html , Packit (Packet toolkit) (Herramienta de auditoría). Accedido en julio 2011

- http://www.Hping.org/ , Salvatore Sanfilippo, Hping, (Ensamblador y analizar de paquetes TCP/IP). Accedido en julio 2011

Referencias

[Accedido en Julio 2011]

- [1] http://www.gartner.com/technology/about.jsp , Gartner Group Inc.
- [2] http://www.psion.com/es/acerca-de.htm , Psion Inc.
- [3] http://es.wikipedia.org/wiki/C%2B%2B , C++.
- [4] http://www.nokia.com/about-nokia , Nokia Corporation.
- [5] http://www.ericsson.com/thecompany , Telefonaktiebolaget LM Ericsson.
- [6] http://www.motorola.com/Consumers/US-EN/About_Motorola , Motorola Mobility, Inc.
- [7] http://panasonic.net/about/ **,** Panasonic Corporation.
- [8] http://es.wikipedia.org/wiki/GNU/Linux , Linux.
- [9] http://es.wikipedia.org/wiki/Microsoft_Windows , Windows.
- [10] http://es.wikipedia.org/wiki/Bluetooth , Bluetooh.
- [11] http://es.wikipedia.org/wiki/Caballo_de_Troya_%28inform%C3%A1tica%29 Caballo de Troya.
- [12] http://es.wikipedia.org/wiki/Online_Certificate_Status_Protocol , OCSP
- [13] http://es.wikipedia.org/wiki/Hash , Hash Criptográfico.
- [14] http://es.wikipedia.org/wiki/Wi-Fi , Wi-Fi.
- [15] http://www.java.com/es/download/faq/whatis_j2me.xml , J2ME.
- [16] http://es.wikipedia.org/wiki/DOS , DoS.
- [17] http://www.apple.com/contact/ , Apple Enterprise
- [18] http://us.blackberry.com/company.jsp , BlackBerry / Research In Motion
- [19] http://www.android.com/ , Android / Google Inc.
- [20] http://www.microsoft.com/windowsphone/es-es/features/default.aspx , Windows Móvil 7 / Microsoft.

- [21] http://es.wikipedia.org/wiki/Botnet , Botnet
- [22] http://nmap.org/ , Nmap
- [23] http://www.tenable.com/products/nessus , Nessus
- [24] http://es.wikipedia.org/wiki/Familia_de_protocolos_de_Internet , TCP/IP
- [25] http://es.wikipedia.org/wiki/Ataque_Man-in-the-middle , MiTM