DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico

Malware de Android

25 de julio de 2015

Seguridad Informática
1er Cuatrimestre de 2015

Integrante	LU	Correo electrónico
Castro, Alan		alancastro90@gmail.com
Kujawski, Kevin	459/10	kevinkuja@gmail.com
Ortíz de Zárate, Juan Manuel		jmanuoz@gmail.com
Vanecek, Juan	169/10	juann.vanecek@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción

Se nos presentó el desafío de crear un Malware para Android que simulando ser una aplicación normal por atrás le robe información al usuario del dispositivo. Entre una serie de varias opciones para robar decidimos 2, robar los contactos y la ubicación del dispositivo, para luego armar el historial de ubicaciones.

2. Herramientas utilizadas

Para trabajar con Android vamos a necesitar el SDK, un emulador para celulares, y un entorno de programación. Todo esto viene incluido en un paquete fácil de utilizar.

Comenzamos entonces descargando el ADT Bundle (http://developer.android.com/sdk/index.html) que incluye los componentes esenciales del SKD de Android y una versión del Eclipse que incorpora ADT (Android Development Tools) para agilizar el desarrollo de aplicaciones Android. La herramienta incluye el emulador que utilizamos para simular que estamos trabajando sobre un celular y poder hacer las verificaciones más rápidamente. Para correrlo hay que seleccionar un dispositivo, nosotros utilizamos el Nexus One que es un celular básico.

Por otro lado para el servicio no hace falta utilizar algo tan pesado como otro Eclipse, así que preferimos utilizar una aplicación en PHP. Para ello necesitamos un servidor.

El WAMP (Windows Apache MySql PHP) monta un servidor completo en Windows de manera muy sencilla, simplemente instalándolo y corriéndolo

Ahora sí, podemos levantar el servidor y tener la aplicación funcionando. Para esto solo necesitamos copiar la carpeta del servidor en la carpeta www que se encuentra en el directorio en donde se instaló el WAMP

3. Aplicación

3.1. Cliente

El servicio que roba los datos propiamente dichos es `HijackerService` que hereda de `IntentService` e implementa dos métodos de su protocolo:

- `public int onStartCommand(Intent intent, int flags, int startId)`

  Para que cada vez que se ejecuta se muestre un Toast con el texto *robando* para verificar que se está ejecutando. Claramente si se quiere utilizar esta aplicación con mala intención hay que comentar esta línea.

- `protected void onHandleIntent(Intent intent)`

  Que es el método invocado cuando al thread se le solicita un pedido para procesar. Es dentro de este método que se llama a dos clases `ContactsHijacker` y `LocationHijacker`, las cuales se encargan de leer la información correspondiente y dársela a `ServerWrapper` que es el que se encargará de enviarla al servidor.

  Cabe mencionar que estas dos clases trabajan independientemente una de otra y mandan la información al server por separado para evitar que los contactos tengan que esperar a los datos del GPS.

El servidor también lo implementamos nosotros, para probar nuestro malware. Aunque bastante simple, cumple con el propósito de almacenar en dos archivos de texto los contactos y las ubicaciones recibidas, para eventualmente luego mostrarlas. `ServerWrapper` podría enviar la información a cualquier otro lado si se quisiera.

Se decidió implementar `HijackerService` como un servicio para que pudiera ejecutarse independientemente y de fondo al thread principal. Pero como todo Intent se ejecuta una sola vez y luego muere, por lo que necesitábamos una clase que funcione de lanzador del mismo.

Para ello diseñamos la clase `HijackerServiceLauncher`, quien se encarga de ejecutar periódicamente a `HijackerService`. El tiempo está dado en segundos por la variable privada `FREQ_HIJACKER`.

Por último queríamos que el launcher del malware se ejecutara o bien cuando se iniciaba el juego o bien cuando se iniciara el teléfono, por lo que creamos la clase `HijackerServiceLauncherAtBoot` que extendiera a `BroadcastReceiver` para poder recibir el evento de booteo completo, momento en el cual ejecuta a `HijackerServiceLauncher`.

Si el launcher no estaba corriendo, se ejecuta cuando el usuario abre el juego 2048 y el MainActivity lo lanza. Como `HijackerServiceLauncher` está diseñado como un singleton, nos garantizamos que solo haya una instancia que lanza el malware, aminorando las posibilidades de sospecha del usuario quién notaría su celular con memoria y cpu ocupada con muchos procesos innecesarios.

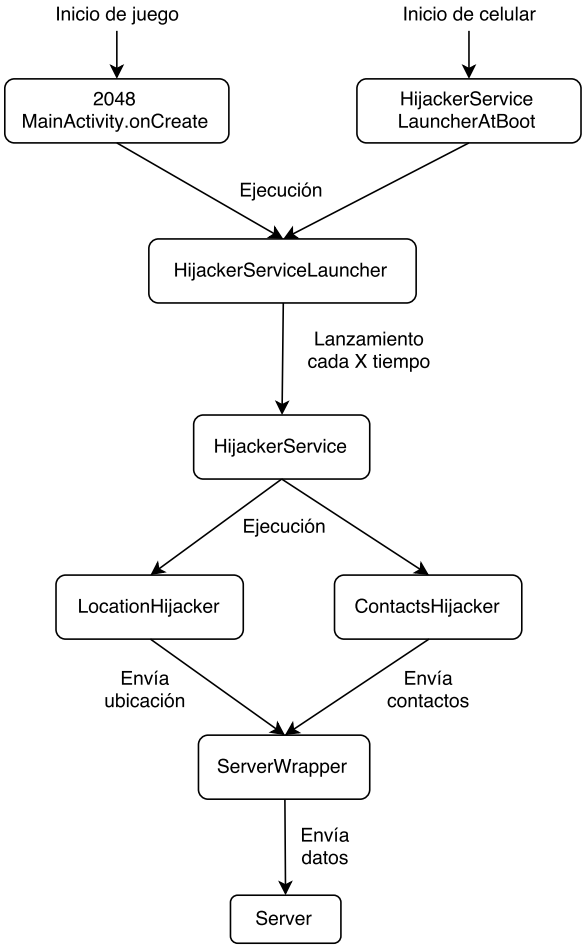

Figura 1: Diagrama de ejecución

3.2. Servidor

El servidor es el que se encarga de recibir los datos que le envia la aplicacion, guardarlos y posteriormente mostrarlo. El mismo esta desarrollado en PHP, utilizando Bootstrap como framework de diseño y GoogleMaps para la funcionalidad del mapa, la información es guardada en archivos de texto.

El servidor tiene dos archivos diferentes para recibir los distintos tipos de información, él mismo lee por POST los parámetros que el servidor le envia en tipo JSON, lo decodifica y lo guarda.

Si se accede al index.php del sitio se verán los datos de la siguiente manera:

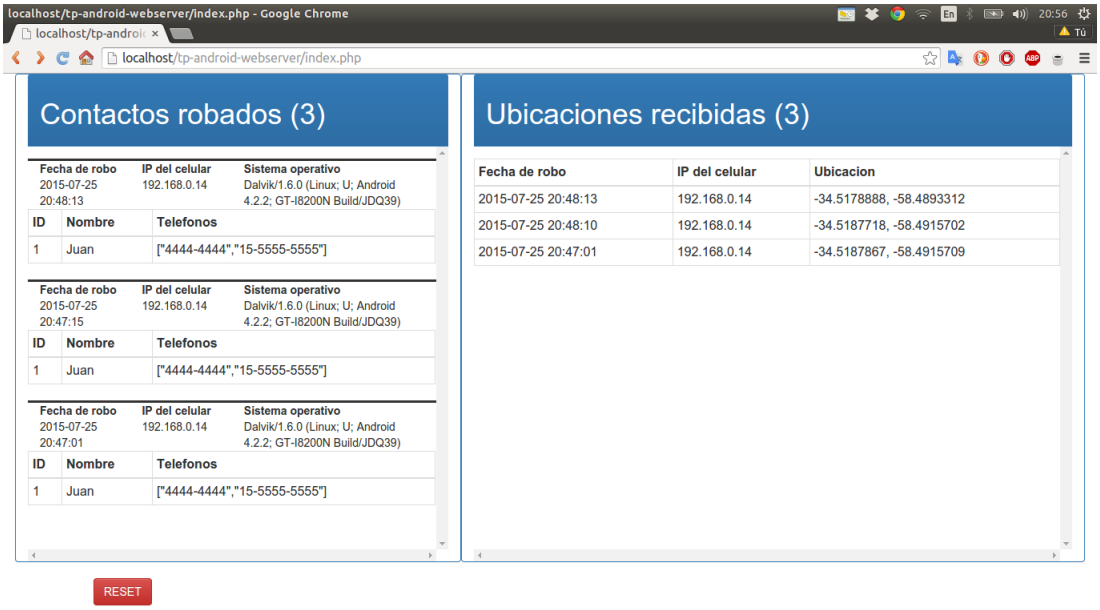

Figura 2: Frontend Server

4. Discusión

Para que funcione el malware tenemos que agregar ciertas variables al Manifest: el registro del service `HijackerService`, y de `HijackerServiceLauncherAtBoot` indicando que recibirá la acción de `BOOT_COMPLETED`, así como los permisos que tendrá la aplicación:

1. `READ_CONTACTS` para leer los contactos.
2. `ACCESS_FINE_LOCATION` para acceder al gps.
3. `ACCESS_COARSE_LOCATION` para acceder a la ubicación a través del proveedor de internet.
4. `INTERNET` para poder usar internet.
5. `ACCESS_NETWORK_STATE` para leer el estado de la red.
6. `RECEIVE_BOOT_COMPLETED` para poder lanzar el malware al iniciar el celular.

Como decidimos hacer el malware en una sola aplicación estos son los mínimos permisos que se requieren para funcionar y para el juego 2048 se podría argumentar que son exagerados y un usuario podría sospechar de instalarlo. Pero como el malware no depende de él para funcionar, se podría poner en cualquier aplicación que los usara realmente ya que son permisos comunes y básicos ciertas apps, y así el usuario no sabría que se están aprovechando de estos permisos para robarle información.

Hay que tener en cuenta que para que el malware funcione este debe estar conectado a internet para poder mandar la información al server, y el GPS debe estar prendido para obtener una ubicación más precisa, aunque si no lo está utilizará la ubicación del proveedor de internet.

Es parametrizable la frecuencia de robo de la información modificando la variable `FREQ_HIJACKER` de la clase `HijackerServiceLauncher` y la dirección del server a donde enviar los datos. Esto hay que setearlo antes de compilar la aplicación.

De los contactos decidimos robar solamente el nombre y los números telefónicos por comodidad, pero se podría agregar cualquier otra información que se quisiera.

A lo largo del desarrollo del malware no tuvimos muchas dificultades, simplemente el testeo de la aplicación en un emulador que traía ciertas limitaciones como puede ser que no reconozca el GPS y haya que enviarle la información por otro lado, y el testeo de que se inicie apenas se prenda el telefono. A pesar de eso, en cuestiones de visualización, lógica y acceso a la información a robar no tuvimos problema, lo que nos concientiza sobre lo fácil que es modificar una aplicación y manipular al usuario para hacer fraude sin que este se diera cuenta.

4.1. Mejoras

- Configuración de la IP del server.
- Configurar el tiempo de envio de la información. Actualmente esta fijo en 10 segundos, factor que podria generar que se note la sobrecarga y por consiguiente que se detecte el malware.
- Mejorar el servidor visualmente.
- Lograr identificar el dispositivo. Ya sea por un login o por la MAC del dispositivo.

5. Bibliografía

1. Xuxian Jiang and Yajin Zhou (2013), Android Malware, ISBN 978-1-4614-7393-0.
2. StackOverflow
3. Google