Android y el Malware

Juan Heguiabehere Joaquín Rinaudo

22º Escuela de Verano de Ciencias Informáticas
RIO 2015

Qué es el Malware en general

Malware es el nombre genérico que se le da al software cuyo propósito resulta en causar un perjuicio al usuario (o a terceros):

- Robo de información
- Falta de disponibilidad de los equipos o la información
- Gastos no autorizados

Tipos de malware:

- Virus
- Troyanos (*)
- Gusanos

Riesgos asociados a la telefonía móvil

- Robo de información del usuario
  - Número de teléfono, listas de contactos, llamadas hechas/recibidas, agenda
  - Ubicación geográfica actual/histórica
  - Interacción con el teléfono (UserID, clave)
  - Activación y transmisión del micrófono/cámara
- Costos agregados para el usuario
  - Envío de SMS/llamadas a servicios premium
- Vector de ataque por comunicaciones inalámbricas
  - Access point falsos
  - Captura de paquetes WiFi
  - Acceso a redes internas, eludiendo firewalls
- Ataques a la disponibilidad del dispositivo
  - Gasto excesivo de batería
  - Ransomware

Amenazas al modelo de seguridad de Android

O cómo logra el malware sus propósitos

- Vulnerabilidades del kernel de Linux (Escalada de privilegios)
  - Permite hacerse con el control del aparato (rooteo)
  - Son relativamente escasas
  - Pueden hacerse obsoletas con una actualización, pero las actualizaciones son raras
- Abuso de confianza
  - La aproximación 'inocente': la aplicación solicita los permisos que necesita en tiempo de instalación
- Restricciones por compromisos de diseño
  - La tarjeta SD tiene formato FAT, que no implementa permisos
- Aplicaciones vulnerables
  - Algunas aplicaciones pueden ser 'engañadas' para divulgar información sensitiva

Malware en Android

- Por desarrollo:
  - Aplicación simple (por ejemplo una linterna)
  - Aplicación legítima reempaquetada
- Acceso a los recursos protegidos:
  - Abuso de confianza
    - No requiere conocimientos avanzados
    - Mayor parte de las muestras observadas
  - Vulnerabilidades del sistema
    - Explotan vulnerabilidades del kernel de Android
    - Requieren un análisis detallado
    - Menos comunes
    - Pueden dejar de funcionar si se actualiza el kernel

Defensas contra el malware - Análisis estático

- Análisis de AndroidManifest.xml
- Análisis del código binario
- Ejemplos:
  - Androguard (`http://code.google.com/p/androguard/`)
  - APKtool (`http://code.google.com/p/android-apktool/`)
  - JD (`http://jd.benow.ca/`)
  - Smali/Baksmali (`https://code.google.com/p/smali/`)
  - APK2gold (`https://github.com/lxdvs/apk2gold`)
  - Dexter(online) (`http://dexter.dexlabs.org/`)

Análisis estadístico de permisos

- Repackaging
  - Permisos app portadora + permisos específicos del MW
- Hipótesis: Podemos distinguir el malware de una aplicación legítima a través de los permisos que solicita
  - Muestras de malware: VirusShare + AMGP
  - Muestras de goodware: Google Play Store
- Métodos de discriminación:
  - Discriminador lineal
  - Método de Bayes
  - Machine Learning (x ej Weka)

Método de Bayes

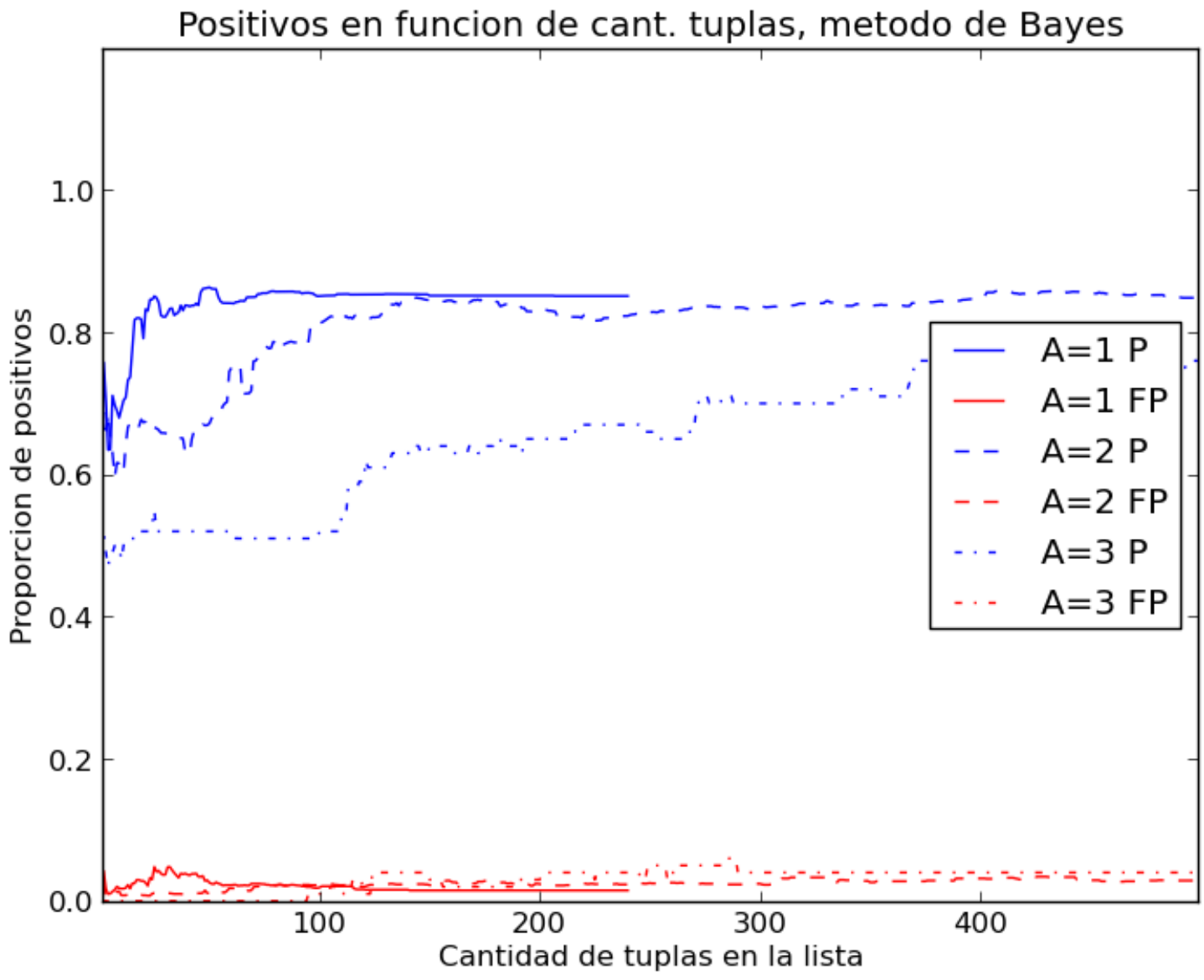
$$P(y \mid x, \mathcal{H}) = \frac{P(x \mid y, \mathcal{H})P(y \mid \mathcal{H})}{\sum_{y'} P(x \mid y', \mathcal{H})P(y' \mid \mathcal{H})}$$

donde:
y es el tipo de aplicación (malware o aplicación legítima),
x es una tupla de permisos de la aplicación a evaluar,
$\mathcal{H}$ es nuestra estimación sobre el balance entre malware y aplicaciones legítimas,
$P(y \mid x, \mathcal{H})$ es la probabilidad de que la tupla x esté tomada de una muestra de tipo y,
$P(x \mid y, \mathcal{H})$ es la frecuencia relativa de la tupla x en el total de muestras de tipo y,
$P(y \mid \mathcal{H})$ es la probabilidad *a priori* de que una muestra sea de tipo y,
$P(x \mid \mathcal{H})$ es la frecuencia relativa de la tupla x en nuestra muestra.

Criterios

- Verdaderos positivos (TP)
- Verdaderos negativos (TN)
- Falsos positivos (FP)
- Falsos negativos (FN)
- Precisión: cuántos de los positivos detectados son verdaderos
- Exhaustividad: cuántos de los positivos verdaderos son detectados
- Prec $=$ TP/(TP+FP)
- Exh $=$ TP/(TP+FN)

Método de Bayes - Resultados



Detección de posibles riesgos de privacidad

- Algunos permisos permiten obtener información sensitiva
  - Leer contactos
  - Leer SMS
  - Leer agenda
  - Leer ubicación
- Otros permiten enviar información a otro lado
  - Internet
  - Enviar SMS
  - Llamar por teléfono (más complicado)
- Cualquier combinación entre permisos del primer grupo y permisos del segundo es un riesgo de privacidad

Análisis estático - El código

Las aplicaciones de Android, en mayor o menor medida, se pueden decompilar para acceder al código fuente. Esto nos permite:

- Buscar llamadas sospechosas al sistema (envío de SMS, lectura de la agenda en una linterna)
- Buscar strings específicos (0-610-algo, nombres de archivos del sistema)
- Buscar usos de Reflection (puede estar levantando código que no viene en el DEX)
- Buscar intentos de explotación del kernel
- Dificultades posibles:
  - El código puede estar ofuscado
  - El contenido extra puede estar encriptado
  - El contenido extra puede no estar entre los recursos de la aplicación, sino que se descarga de la red
  - Canales encubiertos

Defensas contra el malware - Análisis dinámico

- Ejecución de la aplicación en un ambiente controlado
  - Registro de llamadas al sistema
  - Seguimiento de información sensible (taint tracking)
  - Intentos de explotar bugs del kernel
- Ejemplos:
  - Taintdroid (`http://appanalysis.org`)
  - Droidbox (`http://code.google.com/p/droidbox`)
  - DroidScope (`http://code.google.com/p/decaf-platform/wiki/DroidScope`)
  - APK Analyzer (online) (`http://www.apk-analyzer.net/`)
  - Andrubis (online) (`http://anubis.iseclab.org/`)

Seguimiento de información (taint tracking)

- VM modificada para incluir información de sensitividad de los datos.
- Cuando una variable influye en el valor de otra, se transfiere el nivel de sensitividad
- También se siguen datos en su paso por archivos y métodos
- Se ejecuta la aplicación, pero se alerta si información sensible es transmitida fuera del teléfono
- Limitaciones: Canales encubiertos - detección de ambientes virtuales

Análisis dinámico: Interacción

Se interactúa con la aplicación instalada

- Drozer (interactivo)
- CobraDroid (preparado para usar con proxies)
- Burpsuite/mitmproxy (Proxies)
- Cydia Substrate (Inyección de código)

Defensas contra el malware - Mitigaciones

- Antivirus específicos de Android (AVG, Dr. Web, Avast...)
- PatchDroid: Aplica parches para las vulnerabilidades de Android (necesita root)
- En Google:
  - Mejoras al kernel (bug fixes, fortificaciones)
  - Google Bouncer
    - Análisis estático y dinámico
    - Se puede eludir
    - No se recomienda jugar con el Bouncer
  - Verify Apps
    - Consulta a Google cuando queremos instalar una app
    - Avisa si es sospechosa o directamente maliciosa
    - Monitorea el comportamiento de apps ya instaladas

Otras iniciativas

- Android Malware Genome Project (Zhou y Jiang, NCSU)
  - Más de mil piezas de malware analizadas y clasificadas
  - Corpus a disposición de otros investigadores
- VirusShare
  - Centro de acopio de malware, todas las arquitecturas
  - Sección de Android ($\sim$11000 muestras)
- VirusTotal
  - Análisis en línea de archivos (52 antivirus distintos)
  - Análisis estático automatizado

Tópicos avanzados

- Malware
  - Ofuscación de bytecode
  - Detección de ambientes virtuales
- Android:
  - ASLR
  - SELinux
  - NX, PIE, `FORTIFY_SOURCE`
- Antivirus:
  - Mejoras en los ambientes virtuales
  - Análisis dinámico de comportamiento
- Aplicaciones legítimas:
  - Medidas anti-reempaquetado (modificación de la máquina Dalvik)

Redondeando

- Las defensas contra malware están poco a poco alcanzando el nivel que tienen en SOs de PC.
- El malware también está de a poco siendo más sofisticado... pero sigue necesitando que el usuario lo instale voluntariamente.
- Sólo instalar aplicaciones de origen confiable, y aún así tener cuidado.
- A veces se cuela malware en el PlayStore: ver que el autor sea confiable.
- La primera y más importante línea de defensa es **el usuario**.
- Incluso con aplicaciones legítimas, hay riesgos, porque las aplicaciones en sí pueden ser vulnerables.

Referencias

- Malware
  - Métodos de detección de malware
  - Análisis forense de malware de Android
  - Android Malware Genome Project
- Análisis estático
  - Androguard
  - APKInspector
  - Detección de aplicaciones "piggybacked"
- Análisis dinámico
  - Drozer
  - CobraDroid
  - Burpsuite
  - Cydia Substrate
  - PREC

Referencias - Ofuscación de bytecode

- Droid Chameleon
- Proguard
- DashO
- DexProtector
- ApkProtect
- Shield4j
- Stringer
- Allatori
- O-LLVM
- Divilar

Referencias

- Vulnerabilidades del kernel
  - Lista de vulnerabilidades de Android
  - Explicación de TowelRoot
  - Análisis de vulnerabilidades y técnicas de explotación
  - Exploits varios (Justin Case)
- Defensas del kernel
  - ASLR (Address Space Layout Randomization)
  - SELinux (Security Enhanced Linux)
  - Lista de mejoras de la seguridad de Android