UNIVERSITÀ DEGLI STUDI
DI MILANO

Android Security Overview

prof. Carlo Bellettini

carlo.bellettini@unimi.it

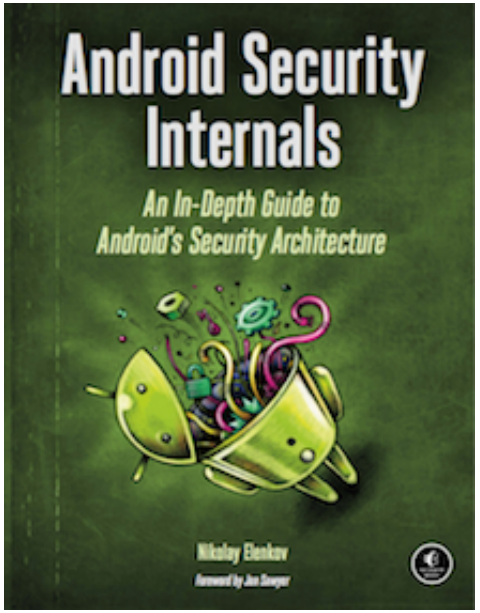
Sondaggio

Quanti hanno un telefonino Android?

Mia esperienza su Android

- ho iniziato a dare alcune tesi su Android quando c'era il G1
- ho avuto vari telefonini: HTC Hero, Sony Experia Play, Samsung Galaxy Nexus, Moto X 2014
- ho un tablet: Samsung Galaxy Tab 10.1

Risorse principali



https://source.android.com/security/enhancements/index.html

Deve essere sicuro?

È un target appetitoso

- contiene dati sensibili (nostre password, email, calendario, foto, ???)
- è sempre connesso (risorsa preziosa per attacchi, per reperibilità..)
- sono molto comuni
- hw con significativa potenza di elaborazione
- nella maggior parte dei casi, gli update (anche di sicurezza) non sono disponibili o installati

È sicuro?

"Scary Android security hole in 99% of phones: PANIC!" – Computerworld

"HTC promises fix for massive Android security flaw" – MobileBeat

"Android users are two and a half times as likely to encounter malware today than 6 months ago..." – Lookout Mobile Threat Report

"Today's mobile devices are a mixed bag when it comes to security... still vulnerable to many traditional attacks...." - Carey Nachenberg, Symantec

"Android Security Will Be Big News in 2011: 10 Reasons Why" - eWeek

"The growth rate in malware within Android is huge; in the future there will definitely be more." - Nikolay Grebennikov, CTO of Kaspersky

"Any time a technology becomes adopted and popular, that technology will be targeted by the bad guys." - Jay Abbott, PricewaterhouseCoopers LLP

\

Cosa vogliamo?

- Proteggere il sistema da applicazioni *pericolose*
- Avere controllo su cosa può fare una applicazione
- Avere garanzie su autore dell'applicazione
- Avere garanzie che l'applicazione non sia stata manomessa
- Bloccare accessi non autorizzati
- Tenere sicuri i dati in caso di furto/perdita

Isolamento applicazioni

Usiamo il sandboxing della JVM?

- in realtà non usiamo proprio la `JavaVM`, ma un `Android RunTime` (*ART*) e prima ancora una `DalvikVM`

ART e Dalvik VM non effettuano sandboxing

- non programmiamo solo in Java...

Basato su Linux

Linux ci fornisce:

- Modello dei permessi user-based
- Isolamento dei processi
- Meccanismi di IPC sicuri

Application Sandboxing

La sicurezza viene forzata a livello di OS

- UID e GUID univoci (localmente) per ogni applicazione
  - fissati al momento della installazione
    - si parte da uid = 10000

Application Sandboxing

La sicurezza viene forzata a livello di OS

- UID e GUID univoci (localmente) per ogni applicazione
  - fissati al momento della installazione
    - si parte da uid = 10000
  - permette chiara separazione tra le applicazioni (da parte di OS)
    - identifica le risorse accessibili dalle singole applicazioni

FileSystem Protection

- Partizionamento del file system
  - / e /system sono *read-only*
  - /data e /cache sono *read-write*
- I file sono assegnati agli Utenti/Applicazioni... quindi gli accessi sono protetti l'uno dall'altra
  - È possibile criptare l'intero file system a livello kernel con dmcrypt

Isolamento di default

Inizialmente una applicazione ad esempio non può:

- Leggere o scrivere fuori dalla propria directory
- (Dis)Installare/Modificare altre applicazioni
- Usare componenti di altre applicazioni
- Usare la rete
- Accedere ai dati dell'utente (Contatti, SMS, email)
- Usare API potenzialmente *non gratuite* (effettuare chiamate, mandare SMS,...)
- Tenere attivo lo schermo, riavviarsi al boot

Permessi

- Vengono richiesti per potere eseguire operazioni *sensibili*
- Le applicazioni dichiarano le loro richieste prima della installazione

Sono caratterizzati da:

- nome e descrizione
- gruppo
- livello (normal, dangerous, signature, signatureOrSystem)

Lista permessi

Lista permessi da: `Manifest.permission`

ACCESS_CHECKIN_PROPERTIES
: Allows read/write access to the "properties" table in the checkin database, to change values that get uploaded.

ACCESS_COARSE_LOCATION
: Allows an application to access coarse (e.g., Cell-ID, WiFi) location

ACCESS_FINE_LOCATION
: Allows an application to access fine (e.g., GPS) location

ACCESS_LOCATION_EXTRA_COMMANDS
: Allows an application to access extra location provider commands

ACCESS_MOCK_LOCATION
: Allows an application to create mock location providers for testing

ACCESS_NETWORK_STATE
: Allows applications to access information about networks

ACCESS_SURFACE_FLINGER
: Allows an application to use SurfaceFlinger's low level features

ACCESS_WIFI_STATE
: Allows applications to access information about Wi-Fi networks

ACCOUNT_MANAGER
: Allows applications to call into AccountAuthenticators.

ADD_VOICEMAIL
: Allows an application to add voicemails into the system.

Controllo dei permessi

All'interno del file `Manifest.xml` dichiaro i permessi di cui ho bisogno

```
    <uses-permission android:name="string" />
```

Controllo statico

Dichiarazione permessi necessari

```
    <activity android:name=".GetPasswordActivity"
      android:permission="com.yourapp.GET_PASSWORD_FROM_USER" … >
    </activity>
    <service android:name=".UserAuthenticatorService"
      android:permission="com.yourapp.AUTHENTICATE_USER" … >
    </service>
    <provider android:name=".EnterpriseDataProvider"
      android:readPermission="com.yourapp.READ_ENTERPRISE_DATA"
      android:writePermission="com.yourapp.WRITE_ENTERPRISE_DATA" … >
    </provider>
```

Definizione nuovo permesso

```
<permission android:name="com.yourapp.PERMISSION"
    android:protectionLevel="signature"
    android:label="@string/permission_label"
    android:description="@string/permission_desc">
</permission>
```

Enforcing dinamico: esempio

Vogliamo scrivere una applicazione che fa vibrare il telefono quando un conoscente è a me vicino.

Abbiamo bisogno dei permessi:

- ACCESS_FINE_LOCATION: per accedere alla posizione GPS
- INTERNET: per trovare e comunicare mia posizione
- VIBRATE: per poter fare vibrare il telefono

```
<uses-permission android:name="android.permission.VIBRATE" />
```

Classe Vibrator Doc e Src

Enforcing dinamico... (cont)

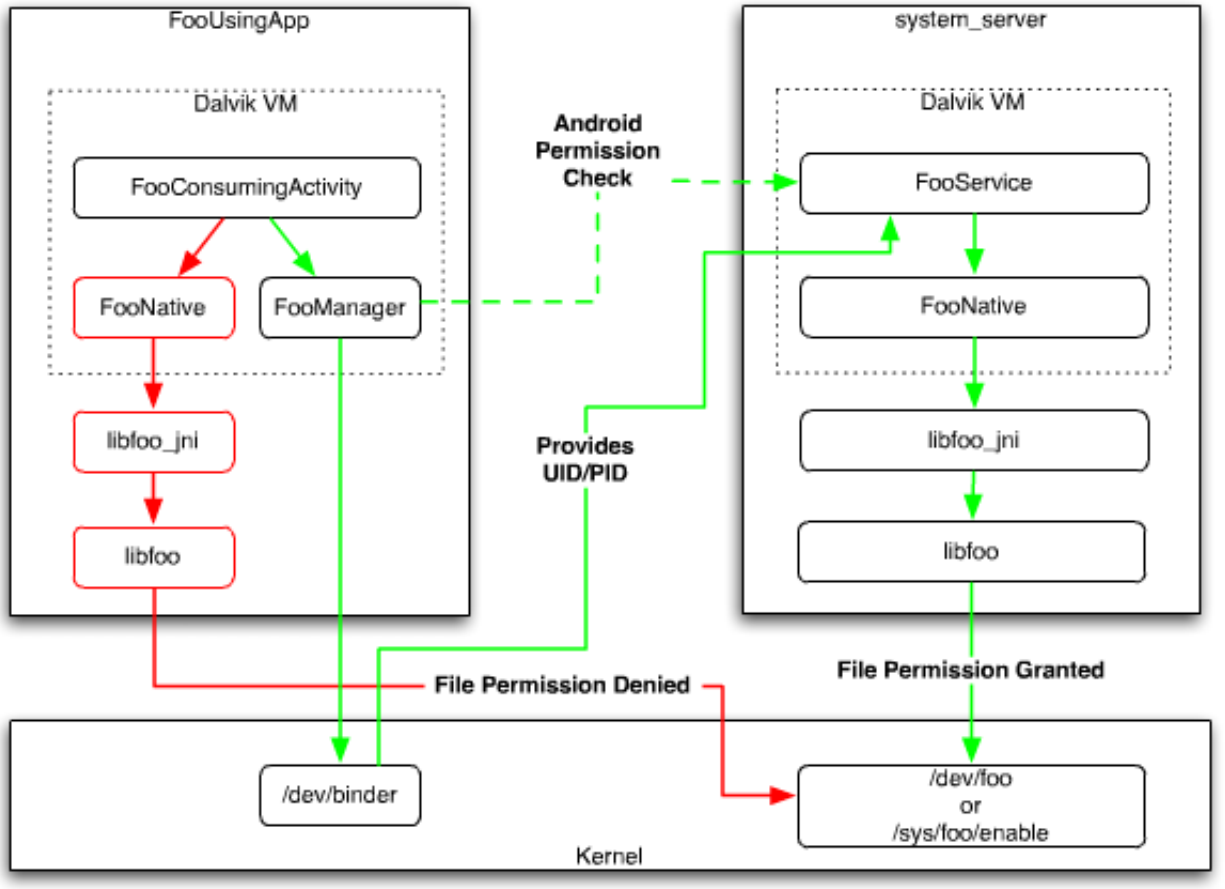
Nel nostro programma useremo

```
mVibrator = (Vibrator)getSystemService(Context.VIBRATOR_SERVICE);
...
mVibrator.vibrate(new long[]{ 0, 200, 0 }, 0);
```

Nel codice del servizio

```
public void vibrate(long milliseconds, IBinder token) {
  if (ctx.checkCallingOrSelfPermission(android.Manifest.permission.VIBRATE)
      != PackageManager.PERMISSION_GRANTED)
    throw new SecurityException( "Requires VIBRATE permission");
}
```

Controllo dinamico dei permessi



Permessi... file sul target

Dal file `packages.xml` (in `/data/system/`)

```
<package name="com.readability"
         codePath="/data/app/com.readability-1.apk"
         nativeLibraryPath="/data/data/com.readability/lib"
         flags="0" ft="136de965d50">
    <sigs count="1">
      <cert index="6" key="308201bd30820126a003020102020044ea974a6300d06092a86
    </sigs>
    <perms>
      <item name="android.permission.INTERNET" />
      <item name="android.permission.ACCESS_NETWORK_STATE" />
      <item name="android.permission.WRITE_EXTERNAL_STORAGE" />
    </perms>
</package>
```

Permessi e GID

Guardiamo il file `/etc/permissions/platform.xml`

```
<permissions>

    <permission name="android.permission.INTERNET" >
        <group gid="inet" />
    </permission>

    <permission name="android.permission.WRITE_EXTERNAL_STORAGE" >
        <group gid="sdcard_r" />
        <group gid="sdcard_rw" />
    </permission>

    <assign-permission name="android.permission.MODIFY_AUDIO_SETTINGS" uid="m
    <assign-permission name="android.permission.ACCESS_SURFACE_FLINGER" uid="

</permissions>
```

Sotto Android non ci sono file `/etc/passwd` o `/etc/groups` e quindi gli ID sono hardcoded [src]

```
#define AID_INET      3003  /* can create AF_INET and AF_INET6 sockets */

 ...

static const struct android_id_info android_ids[] = {
    { "root",      AID_ROOT, },
    ...
    { "inet",      AID_INET, },
    ...
};
```

MultiUser

In Android 4.2 è stato aggiunto supporto per multiutenza.

Ad ogni utente viene assegnato un identificativo: `00,10,11,12`

I processi assumono dei PID ottenuti per concatenazione:

- u0_a3 → 10003
- u0_a4 → 10004
- …
- u10_a3 → 100003
- u10_a4 → 100004
- …
- u11_a3 → 110003
- u11_a4 → 110004

Garantire l'origine di una applicazione

Le applicazioni devono essere firmate per potere essere installate

- ma non viene richiesto l'uso di una CA: chiavi self signed

A cosa servono?

- per garantire update

Attacchi

- root exploit
- social engineering
- privilege escalation
- *combo* privileges
  - shared uid

Root exploit

Non è previsto che una applicazione (non di sistema) abbia diritti di root

- bypasserebbe qualunque controllo permessi

Esistono diversi *exploit*

- exploid
  - baco in `udev` (init/ueventd)
- rageagainstthecage
  - race condition in `adbd`
- softbreak/gingerbreak
  - buffer overflow in `vold`

Social Engineering

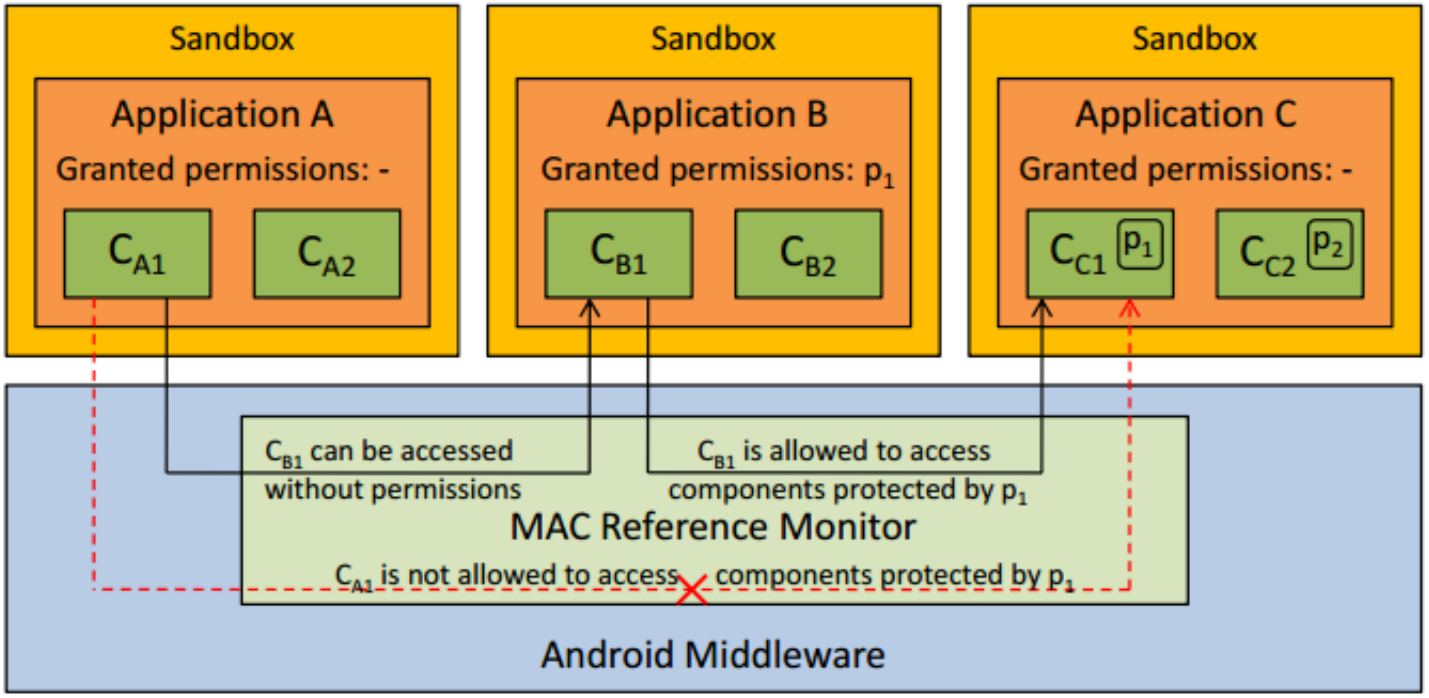
- Prendi una applicazione (popolare) o inventa qualcosa di accattivante
- Inserisci del codice malizioso (trojan, root exploit!?!)
- Reimpacchetta e rifirma con un tuo nuovo certificato
- Rimetti sul market (o su sito web, o market alternativo)
- Aspetta :-D

Esempi (Symantec):

rootcager, Pjapps, Bgserv

Privilege Escalation

Un componente riesce (indirettamente) a chiamare un componente protetto da un permesso che non ha



Shared UID

Non è vero che per forza ogni applicazione ha UID univoco

È *possibile* condividerlo **per applicazioni firmate con la stessa chiave**

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  ...
  android:sharedUserId="string"
  android:sharedUserLabel="string resource"
  ... >
...
</manifest>
```

Possibile scenario: shared ID

Il developer *CB* sviluppa due applicazioni:

- gestore UnimiSpaces
  - richiede tra gli altri permessi `full internet`

Possibile scenario: shared ID

Il developer *CB* sviluppa due applicazioni:

- gestore UnimiSpaces
  - richiede tra gli altri permessi `full internet`
- generatore di password (SuperGen?)
  - .red[NON] richiede permessi di rete e quindi penso di potermi fidare

Singolarmente le due applicazioni mi sembrano ragionevoli, ma *combinate* insieme?

Altro?

- sblocco del telefono (PIN?, PASSWORD?, Path?, Sorriso?)
- se qualcuno compromette il vostro account google, può installare remotamente applicazioni
- Side loading e google app verification service

SELinux

- Supporto iniziale in 4.3
- Permette di stabilire delle policy
  - forzate a livello di kernel (quindi è possibile forzare vincoli anche per processi eseguiti da root)
- Scenari:
  - Symlinks
  - AppData
  - System Files

Risorse

- Google
- Libri
  - *Nikolay Elenkov* - Android Security Internals: An In-Depth Guide to Android's Security Architecture
  - *Joshua J. Drake et al.* - Android Hacker's Handbook
- Privilege Escalation Attacks on Android
  - http://link.springer.com/content/pdf/10.1007%2F978-3-642-18178-8_30.pdf
- MultiUser Security Problems
  - https://arsenb.wordpress.com/2014/09/30/android-multiuser-model-architecture-and-related-security-threats/
  - http://www.cis.syr.edu/~wedu/Research/paper/multi_user_most2014.pdf