CRePE and Miscellaneous Android Hacking

Earlence Fernandes

vrije Universiteit amsterdam

Talk at University of Padova, Italy
I'll be talking about:

- CRePE Introduction
- Android: SDK and System View
- Interesting Implementation Bits
CRePE: Context Related Policy Enforcement for Android

- Supports Security policies which activate based on environmental conditions

- Supports runtime permission revocation

- Has a flexible and expressive policy language and on-device lexer and parser

- Uses “channels” to receive policies. Examples are QR Codes, Internet/WiFi, Bluetooth, SMS etc

- Uses secure digital signatures and PKI to identify valid policies

- Reasonable Energy and Time Overhead

- Pluggable Context Detector – CONTRIBUTIONS WELCOME!
CRePE-like Features on Ice Cream Sandwich


Android Device Administration – They support the ability to disable the android Camera, based on “certain conditions” which are time based.

CRePE can do that and a lot more, and we had implemented this long before ICS
Android Architecture
### Activity (and NativeActivity)
1. UI Component
2. Almost all apps have one

### Services
1. No UI
2. Executes in background
3. NOT a new process**
4. Useful for long running tasks.
   Music.

### Content Providers
1. Uniform Data Storage
2. “Backend” can be anything, including network socket
3. Designed to prevent common SQL injection attacks

### Broadcast Receivers
1. Receive broadcasted events and intents
2. Mostly system broadcasts
Android: A Systems Perspective (1/2)

User Space Application Programs

Android System Framework

JNI Glue

Native Code

Linux Kernel + Android Extensions
Interesting CRePE Implementation Issues

1. Process based internet usage regulation

2. Process based bluetooth usage regulation

3. Bluetooth auto-pairing for CRePE Policies
What?

We must be able to selectively control which process can use the socket API to create a TCP/UDP connection.

Why?

This is an essential security policy template

Why is it interesting?

Current Android Code has implemented access control via a patch to the linux socket API. Only if a process is a member of the INET group, it can use those APIs. If a process is granted android.permissionINTERNET, it becomes a member of INET
Possible Solutions

1. Patch Apache Harmony

2. Patch socket calls like Google did, but only for process IDs

3. ...

An elegant solution exists
Process Based Internet Usage Regulation (3/3)

Netfilter and IPTables

1. Supports UID based conditional access – WHAT WE NEED!

2. Additional Features, IP address based restrictions

3. We can go further. Complex Internet access rules

IPTables and creperoot.c

1. We have a setuid binary that executes iptables as root for us.

2. Only user id 1000 (i.e System) can use this binary.

3. The Java Component “CrepeIPTablesManager” builds rules and manages it.

Look at ANDROID_SRC/frameworks/base/services/java/com/android/crepe/IPTables.java
1. Android uses BlueZ

2. No single point (atleast I couldn't find one) where I could place a hook which checked the calling UID

3. Java based code for BT is Android-custom. We placed a hook at BluetoothSocket.connect() and BluetoothServerSocket.connect()

4. Disadvantage is that only Java apps will be controlled. Not native code
Bluetooth Auto-pairing for CRePE policies

1. We need to auto-pair with a policy sender for transparent policy transfer.

2. BluetoothEventLoop.java contains the code that will decide to display a PIN prompt box.

3. We created a special device class which is advertised by the CRePE policy sender.

4. Patched the file to auto-pair with a predefined PIN.