



Toward the integration of Grsecurity in Embedded Android Operating System

Philippe THIERRY & Sylvain LEROY

October 28th 2011
Presentation for ELCE

- ◆ **Introduction**
- ◆ **What is the purpose of our work ?**
- ◆ **About Android software architecture**
 - General software architecture
 - Zygote & the application management
 - The Binder ICC integration
 - Existing security considerations
- ◆ **Why we chose Grsecurity**
- ◆ **Why we didn't choose SELinux**
- ◆ **What we have enabled from GRSecurity**
 - What is currently supported
 - Matching Android activities
 - Basic security enhancement
 - Specific Android security add-ons
- ◆ **Currently working features**
- ◆ **Other works on Android security**
- ◆ **Conclusion**

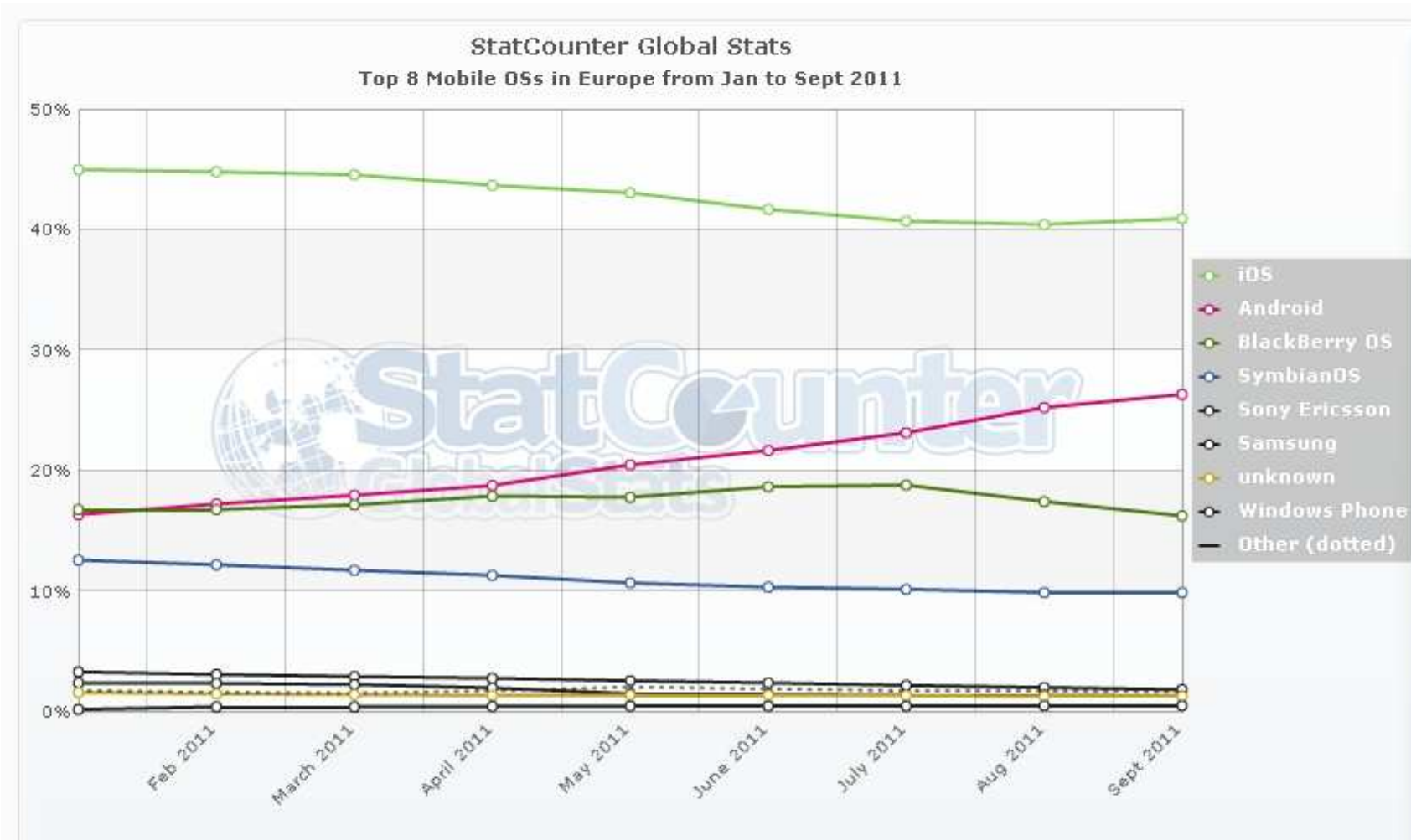
In 2011 :

- ◆ Android is in the top 3 mobile OSs used on smartphones world wide
- ◆ It became number 2 in Europe
 - Before Backberry OS (RIM)
 - After iOS (Apple)

Some figures :

- ◆ 500,000 activations per day world wide¹
- ◆ FIXME

¹Study realized by ComScore (<http://www.comscore.com>)



Source : <http://gs.statcounter.com>

Let's talk a little about GRSecurity

- ◆ Chroot advanced enhancements
- ◆ RBAC policy support with advanced learning mode
- ◆ Modules auto-loading hardening
- ◆ kernel threads and behavior hiding
- ◆ advanced RLIMIT NPROC support
- ◆ advanced logging
- ◆ Trusted Path Execution (TPE) mode
- ◆ Advanced uid & gid filtering for network access
- ◆ Logging overload protections
- ◆ kernel memory protection Null pointer exception protection
- ◆ NX bit software emulation various ELF protections and randomization

GRSecurity web site: <http://www.grsecurity.net>

Our test components :

- ◆ Android froyo (2.2)



- ◆ Nexus One/G1 (dev phone)



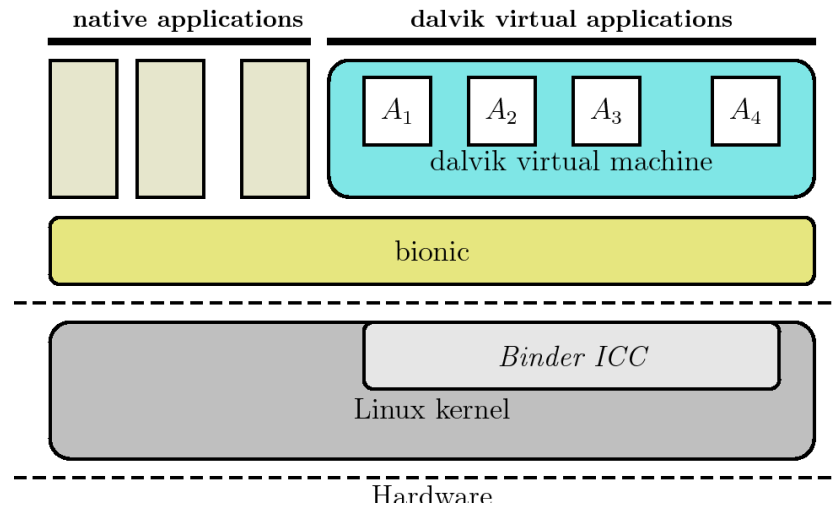
- ◆ Android/Linux kernel 2.6.32.9



- ◆ Grsecurity 2.1.14-2.6.32.9-201003112025



- ◆ **GRSecurity is a well known solution mostly used in x86 GNU/Linux architecture**
 - This solution is able to harden standard GNU/Linux operating system
 - It is able to limit various security vulnerabilities (e.g. some rootkits)
- ◆ **The purpose of this work is to validate the capacity to deploy the GRSecurity solution into Android devices**
- ◆ **The capacity of GRSecurity to handle current Android security threats remains to be done**

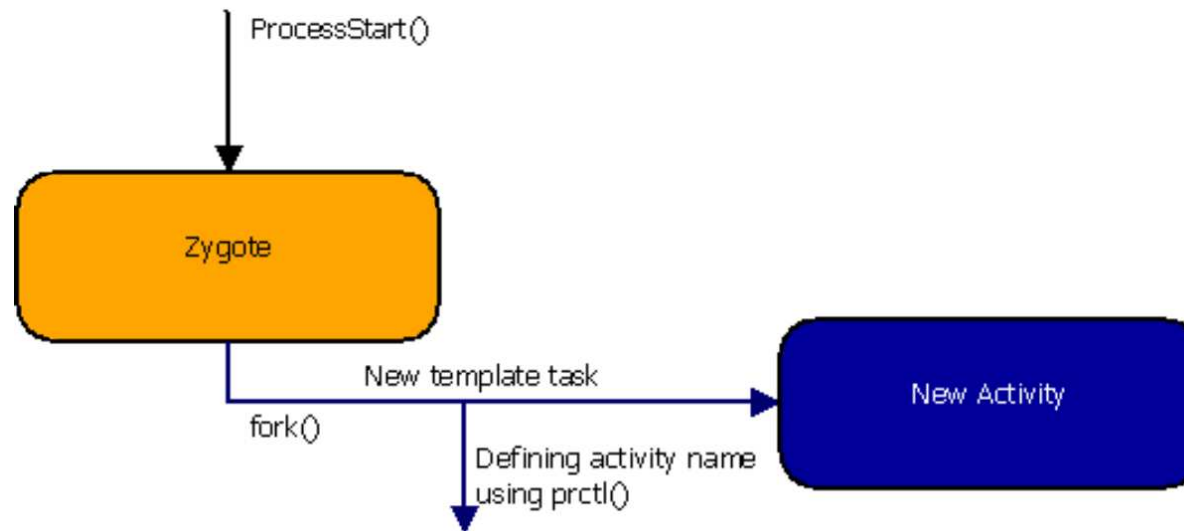


- Android is a Linux-based operating system, with some specificities
- Android architecture is based on a specific libc (i.e. Bionic) and a Java virtual machine (i.e. Dalvik) based applications management
- Pthread implementation is Android specific, optimized for Dalvik

- Android is a Linux-based operating system, with some specificities
- Bionic is not compliant with a standard POSIX libc and does not implement some usual functions like `getprotobyname()`
- Most of the Google added value has been implemented in the Dalvik virtual machine
- A new inter-process communication paradigm has been integrated in the Linux-kernel, using a Corba-like communication system (Binder)

Zygote, « the creator »

- ◆ Runs as root
- ◆ Forks to create new applications on demand
- ◆ Preloads Dalvik's libraries
- ◆ Sometimes « forgets » to change its uid from root to application's uid
 - Zimperlich's exploit from Sebastian Kraemer (efficient on Froyo)



Packages management

◆ User management : uid/gid

- User management : uid/gid
- is chosen at installation time in respect with existing application on the smartphone as in GNU/Linux systems

◆ Signed packages

- Every package is signed with the private key of the developer. Only used to identify the author of the application
- Not for security purposes

◆ Permissions

- Permissions are setup on package's installation for its whole installed life

Let's talk about the Android Binder...

- ◆ Inter-application communication is done using an Android-specific (not POSIX-compliant) kernel component named Binder (based on OpenBinder)
- ◆ The Binder manages pre-defined capabilities defined originally for smartphones :
 - BATTERY_STATS
 - CALL_PHONE
 - CHANGE_WIFI_STATE
 - ...
- ◆ These permissions are used through the Dalvik virtual machine, depending on the permissions declared by the developer in its activity's Manifest.xml file
- ◆ The usage of the permissions is defined by the developer and validated by the user during the application installation
- ◆ Even if these permissions are used by Dalvik applications, standard IPC are still accessible

◆ Process separation

- ◆ « *The kernel is solely responsible for sandboxing applications from each other. In particular the Dalvik VM is not a security boundary, and any app can run native code (see the Android NDK). All types of applications — Java, native, and hybrid — are sandboxed in the same way and have the same degree of security from each other.* » **Source :**

<http://developer.android.com/guide/topics/security/security.html>

◆ ASLR (Address Space Layout Randomization)

- ◆ A « random seed » is chosen at boot time and is unique for every application until reboot
- ◆ This limitation is introduced by prelinking optimization

Just In Time impact

- ◆ The Dalvik virtual machine uses JIT (Just In Time) capacity. The goal of JIT is to compile the code when it is about to be executed for the first time by Dalvik

- ◆ JIT is not compatible with Write^eXecute security enhancement because it needs to :
 1. write (compile) code during runtime
 2. execute it

What we have noticed

- ◆ **The Android userspace is a complex and dynamic world**
 - Dalvik is a large living software
 - Total Physical Source Lines of Code (SLOC) = 214,160
 - Native and Java applications are usually updated and varies from an Android version to another
 - Depending on the phone, some add-on applications may be deployed
- ◆ **As user-space source code access is not yet available for Android 3.x :**
 - Adding or upgrading security features to a newer version might lead to fork Android because user-space add-ons depend on a minimal source code access
 - Security is easier to manage when only one component is impacted
 - Furthermore, the kernel source code is under GPL license, which guarantee its availability

About the impact of SELinux on the user-space

- ◆ SELinux depends on the libselinux, a user-space library which needs to be cross-compiled and deployed in the system
- ◆ User-space applications have to be linked against the libselinux












On SELinux policy management

- ◆ SELinux security contexts management is harder to configure and some new security policies need to be defined for Android Activities

Most important features of the GRSecurity/PaX patch

- ◆ Chroot advanced enhancements
- ◆ RBAC policy support with advanced learning mode
- ◆ Modules auto-loading hardening
- ◆ kernel threads and behavior hiding
- ◆ advanced RLIMIT NPROC support
- ◆ advanced logging
- ◆ Trusted Path Execution (TPE) mode
- ◆ Advanced uid & gid filtering for network access
- ◆ Logging overload protections
- ◆ kernel memory protection Null pointer exception protection
- ◆ N^X bit software emulation various ELF protections and randomization

Enabled official GRSecurity features

- ◆ Chroot advanced enhancements  (need to be tested)
- ◆ RBAC policy support with advanced learning mode  (Binder integration need to be implemented)
- ◆ Modules auto-loading hardening 
- ◆ kernel threads and behavior hiding 
- ◆ advanced RLIMIT NPROC support  (Not tested. Android needs have to be measured first)
- ◆ advanced logging 
- ◆ Trusted Path Execution (TPE) mode  (Not tested at all)
- ◆ Advanced uid & gid filtering for network access  (Need fonctionnal RBAC policy)
- ◆ Logging overload protections 
- ◆ kernel memory protection Null pointer exception protection 
- ◆ N^X bit software emulation various ELF protections and randomization  (JIT & prelinking incompatibilities)

- ◆ **How did we select which security feature to enable**
 - ◆ **Some are specially difficult to manage**
 - PaX memory management
 - ◆ **Some will need Android-specific modifications**
 - Efficient RBAC based network access filtering
 - ◆ **Some are being deployed on new up-to-date hardware and software targets**

Detecting Activity creation

- ◆ Detecting Zygote behaviour when creating a new task

Integrating basics

- ◆ Kernel threads and memory protections
- ◆ chroot enhancement
 - for native apps, not yet tested
- ◆ Simple white listing for Android Activities
 - execution authorization, depending on the application name defined by Zygote

Advanced Android-specific

- ◆ Sandboxing Dalvik
 - scope limitation to Dalvik processes only
- ◆ Layout randomization for non-dalvik processes using glibc
 - currently in progress

- ◆ **Since Dalvik processes can be detected efficiently, other security add-ons (like white-listing) have been easier to deploy**
- ◆ **Nevertheless, defining correct security policy for some activities/providers/managers is not easy :**
 - ◆ **The *com.android.settings* activity configures the system and needs to access every part of it and is, because of that, difficult to manage**

Taintdroid

- ◆ **Detecting any access to sensitive informations by installed applications**
 - Dalvik, libcore and vold have been patched
 - Not tested by us

Other existing solutions

- ◆ **Application allowing the user to reduce permissions asked during application installation**
 - User friendly permissions management per application
 - The user select how restrictive permissions should be
 - Unkown impact on the dalvik and bionic source code
 - Not tested by us

Status of work

- ◆ Grsecurity is a kernel security patch, which need to be ported to various Android kernel version
- ◆ This means that the kernel should be recompiled and flashed on the targets, which is not always easy
- ◆ GRSecurity-based devices are efficiently protected when associated with a controlled infrastructure (i.e. market, VPN...)
- ◆ GRSecurity-based devices security needs to be validated against existing known vulnerabilities



Picture source: web.ac-reims.fr

- ◆ <https://grsecurity.net>
- ◆ <https://developer.android.com>