Learning, Analyzing and Protecting Android with TOMOYO Linux

Japan Linux Symposium 2009
2009.10.23
Daisuke Numaguchi
Tetsuo Handa
Giuseppe La Tona
NTT DATA CORPORATION
1. INTRODUCTIONS
TOMOYO overview

- MAC implementation for Linux
  - Behavior oriented system analyzer and protector
  - Pathname-based MAC tools
- It consists of:
  - a kernel patch (ccspatch)
  - a set of utilities (ccstools) for managing access control settings (a.k.a. policy)
MAC (Mandatory Access Control)

- Restrict access according to policy.
- No exception, no bypass
  - Performed inside kernel space
- SELinux, Smack, TOMOYO, AppArmor, LIDS, grsecurity, etc.
How to use TOMOYO?

- Protect
  - System administrator's operations
- Learning
  - Know system behaviors
- Analyze
  - Debug
Android overview

Java
Android Kernel

- Linux Kernel 2.6 with some changes
  - Reduced set of standard Linux utilities -> toolbox
  - No support glibc -> Bionic libraries
  - No standard IPC -> Binder, specific IPC driver
  - No native windowing system
  - Optimized Power Management
  - Low memory killer, Alarm etc.
Dalvik and Zygote

- Runtime is made by **Java programs running in Dalvik**: Virtual Machine for mobile devices
  - slow CPU, small RAM, no swap space, battery
  - Not a JVM, no JIT: only interpreter of DEX (optimized bytecode obtained from Java .class)
  - Multiple VM instances can run efficiently.
- **Zygote process**:
  - first instance of Dalvik VM, partially initialized
  - load *preload* classes and resources
  - is kept always alive in idle state

When an *application execution* request occurs:
- zygote `fork()`s to a new process...
  - ...which loads the requested package
(Biology concept of “zygote”: duplicate, specialize and differentiate)
Dalvik and Zygote

• Runtime is made by **Java programs running in Dalvik**: Virtual Machine for mobile devices
  • slow CPU, small RAM, no swap space, battery
  • Not a JVM, no JIT: only interpreter of DEX (optimized bytecode obtained from Java .class)
  • Multiple VM instances can run efficiently.

• **Zygote process**:  
  • first instance of Dalvik VM, partially initialized  
  • load *preload* classes and resources  
  • is kept always alive in idle state  

When an **application execution** request occurs:  
• **zygote** fork()s to a new process...  
• ...which loads the requested package  
  (Biology concept of “zygote”: duplicate, specialize and differentiate)
Android boot sequence

*adb*  
vold (mount)  
rild (radio)  
debugged installd

**Daemons**  
Binder  
**Native Servers**  
svcmanager  
mediaserver

**System Services**  
systemserver  
Dalvik VM

**Runtime**  
zygote

**Applications**  
Home  
Dalvik VM  
GUI

Copyright (C) 2009 NTT Data Corporation 2009/10/23
Android security model (1/2)

- Each application runs in its own process
- Runtime in separate instances of Dalvik virtual machine
Each process is a “secure sandbox”

- Linux Discretionary Access Control (DAC) for file access: all applications are assigned a unique UID (constant)
  - UID for system services are hard-coded
  - UID for user packages are progressively assigned at install-time, starting from uid 10000 (and mapped to app_0, app_1, ...); they are saved in a file and are maintained constant during the life of the package on the device.
  - Application specific files are saved in /data/data in separate folders owned by specific UID users
2. TOMOYO ON ANDROID
TOMOYO Linux versions

- There are 2 development lines:
  - Fully equipped version (1.x series)
    - Provides full functionalities of pathname-based MAC (MAC for files, network, capabilities...)
  - Mainlined version (2.x series)
    - Uses Linux Security Modules (LSM)
    - Subset of MAC functionalities (only for files, so far)
      - Missing functionalities will be added in the future
    - Supports only kernels 2.6.30 and later
Android kernel

- Android SDK 1.6 ("donut") comes with kernel 2.6.29.
Android kernel

- Android SDK 1.6 ("donut") comes with kernel 2.6.29.
- TOMOYO 2.x is available since kernel 2.6.30.
- TOMOYO 2.2 function is only file access control.
Android kernel

- Android SDK 1.6 ("donut") comes with kernel 2.6.29.
- TOMOYO 2.x is available since kernel 2.6.30.
- TOMOYO 2.2 function is only file access control.

So, choose TOMOYO 1.x !!
Porting TOMOYO to Android

- Patching Android Kernel with TOMOYO patch
- Adapting ccstools
- Cross-compiling for Android
- Adding TOMOYO Policy Loader to Android boot
- Creating policy
Patching Android Kernel

- TOMOYO 1.7.x (Fully equipped version)
- Emulator (no real Android device needed)
  → Linux kernel version: Goldfish v2.6.29
  "Goldfish" is the name given to the ARM architecture emulated by Android SDK Emulator
- ccspatch 1.7.1-pre for Goldfish v2.6.29
Adapting ccstools

- Ccstools is for managing TOMOYO’s policy.
- Ccstools was intended for use on PC
- Ccstools has been enhanced with Network mode for embedded systems
- More convenient for developing policies and debugging
- Two utilities are needed for the device: ccs-init, ccs-editpolicy-agent
Modifying Android boot (1/2)

- Put "ccs-init (program for activating TOMOYO)" inside /sbin/
  - the kernel will call /sbin/ccs-init before /init starts.
- Copy below files needed by /sbin/ccs-init
  - /system/bin/linker
    - /system/ partition is not mounted yet when /sbin/ccs-init starts.
  - /lib/libc.so
  - /lib/libm.so
    - Environment variable LD_LIBRARY_PATH="/system/lib" is not set yet when /sbin/ccs-init starts.
Modifying Android boot (2/2)

- Put "ccs-editpolicy-agent (program for managing TOMOYO remotely)" inside /sbin/
- Append

```bash
service ccs_agent /sbin/ccs-editpolicy-agent 0.0.0.0:7000 oneshot
```

to /init.rc

- ccs-editpolicy-agent will listen to tcp port 7000
- We can issue "adb forward tcp:10000 tcp:7000" to connect from host environment.
Creating policy

- Put access control settings (a.k.a. policy) in /etc/ccs
  - /sbin/ccs-init will load them

Details:
http://tomoyo.sourceforge.jp/1.7/android-arm.html
TOMOYO on Android overview
TOMOYO on Android overview
EDITING POLICY (VIA AGENT)
Environment

Android emulator (Goldfish)

Application framework

Libraries

Android runtime

TOMOYO Linux (kernel patch)

Ubuntu 8.04

Policy editor
ccs-editpolicy

TOMOYO Agent
ccs-editpolicy-agent

TCP/IP

app app app app

Copyright (C) 2009 NTT Data Corporation
<kernel> /init /system/bin/app_process
 0: 1  <kernel>
 1: 1  /init
 2: 1  /sbin/adbd
 3: 1  /sbin/ccs-editpolicy-agent
 4: 1  /system/bin/app_process
 5: 1  /system/bin/bootanimation
 6: 1  /system/bin/dbus-daemon
 7: 1  /system/bin/debugged
 8: 1  /system/bin/installd
 9: 1  /system/bin/keystore
 10: 1  /system/bin/logcat
 11: 1  /system/bin/mediaserver
 12: 1  /system/bin/qemud
 13: 1  /system/bin/rild
 14: 1  /system/bin/servicegemanager
 15: 1  /system/bin/sh
 16: 1  /system/bin/vold
 17: 1  /system/etc/init.goldfish.sh
          /system/bin/getprop
 18: 1  /system/bin/ifconfig
 19: 1  /system/bin/qemu-config
 20: 1  /system/bin/qemu-props
 21: 1  /system/bin/route
Domain transition tree

```
<kernel> /init /system/bin/app_process

0:  1  <kernel>
   1:
   2:  1  /sbin/adbd
   3:  1  /sbin/ccs-editpolicy-agent
   4:  1  /system/bin/app_process
   5:  1  /system/bin/bootanimation
   6:  1  /system/bin/dbus-daemon
   7:  1  /system/bin/debuggerd
   8:  1  /system/bin/installd
   9:  1  /system/bin/keystore
  10:  1  /system/bin/logcat
  11:  1  /system/bin/mediaserver
  12:  1  /system/bin/qemud
  13:  1  /system/bin/rild
  14:  1  /system/bin/servicemanager
  15:  1  /system/bin/sh
  16:  1  /system/bin/vold
  17:  1  /system/etc/init.goldfish.sh
  18:  1  /system/bin/getprop
  19:  1  /system/bin/ifconfig
  20:  1  /system/bin/qemu-props
  21:  1  /system/bin/route
```
Profile

Profile 0 for disabled, 1 for learning, 2 for permissive, 3 for enforcing
Process tree
Process tree

0: 1 init (1) <kernel> /init
1: 1 + sh (31) <kernel> /init /system/bin/sh
2: 1 + servicemanager (32) <kernel> /init /system/bin/servicemanager
3: 1 + vold (33) <kernel> /init /system/bin/vold
4: 1 + debuggerd (34) <kernel> /init /system/bin/debuggerd
5: 1 + rild (35) <kernel> /init /system/bin/rild
6: 1 + app_process (36) <kernel> /init /system/bin/app_process
7: 1 + app_process (65) <kernel> /init /system/bin/app_process
8: 1 + app_process (110) <kernel> /init /system/bin/app_process
9: 1 + app_process (113) <kernel> /init /system/bin/app_process
10: 1 + app_process (137) <kernel> /init /system/bin/app_process
11: 1 + app_process (153) <kernel> /init /system/bin/app_process
12: 1 + app_process (162) <kernel> /init /system/bin/app_process
13: 1 + app_process (170) <kernel> /init /system/bin/app_process
14: 1 + app_process (186) <kernel> /init /system/bin/app_process
15: 1 + mediaserver (37) <kernel> /init /system/bin/mediaserver
16: 1 + installid (39) <kernel> /init /system/bin/installid
17: 1 + keystore (40) <kernel> /init /system/bin/keystore
18: 1 + ccs-editpolicy: (41) <kernel> /init /sbin/ccs-editpolicy-agent
19: 1 + ccs-editpolicy: (618) <kernel> /init /sbin/ccs-editpolicy-agent
20: 1 + init.goldfish.s (42) <kernel> /init /system/etc/init.goldfish.sh
21: 1 + qemu-props (54) <kernel> /init /system/etc/init.goldfish.sh /system/bin/qemu-props
22: 1 + qemu (43) <kernel> /init /system/bin/qemu
23: 1 + adb (45) <kernel> /init /sbin/adb
24: 1 kthreadadd (2) <kernel>

service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server

servicemanager
Daemons
mediaserver
zygote
Problem with splitting domains

- The applications are executed with different UID (i.e.: root, system, app_, ...) and different process name, but...
Problem with splitting domains

- The applications are executed with different UID (i.e.: root, system, app_, ...) and different process name, but...

- ...they are all fork()ed from app_process!

Copyright (C) 2009 NTT Data Corporation
Problem with splitting domains

- New and unexpected situation for TOMOYO Linux

- In TOMOYO Linux, domain transitions occur after process invocation, that is `execve()`, not `fork()`

→ Splitting domain

```
<kernel> /init /system/bin/app_process
```

in different domains according to each single application is impossible...?
Problem with splitting domains

```
<kernel> /init /system/bin/app_process
```

```
0: 1 <kernel>
1: 1 /init
2: 1 /sbin/adbd
3: 1 /sbin/ccs-editpolicy-agent
4: 1 /system/bin/app_process
5: 1 /system/bin/bootanimation
6: 1 /system/bin/dbus-daemon
7: 1 /system/bin/debuggerd
8: 1 /system/bin/installld
9: 1 /system/bin/keystore
10: 1 /system/bin/logcat
11: 1 /system/bin/mediasever
12: 1 /system/bin/qemud
13: 1 /system/bin/rild
14: 1 /system/bin/servicemanager
15: 1 /system/bin/sh
16: 1 /system/bin/vold
17: 1 /system/etc/init.goldfish.sh
18: 1 /system/bin/getprop
19: 1 /system/bin/ifconfig
20: 1 /system/bin/qemu-props
21: 1 /system/bin/route
```
TOMOYO’s MAC and Android DAC

- Android security rule: data files of one application should be prevented from being accessed by other applications
- This is performed by using DAC permissions, as said before
- TOMOYO can provide with conditional ACL a further insurance that this rule is respected, especially in cases when:
  - DAC permissions are poorly configured
  - root process (zygote) would be hijacked

```
allow_read/write @APP_DATA_FILE if task.uid=path1.uid
allow_unlink @APP_DATA_FILE if task.uid=path1.uid
allow_mkdir @APP_DATA_DIR if task.uid=path1.parent.uid1
```
TOMOYO’s MAC and Android DAC

- DAC’s ability to restrict by UID has a low granularity: only “owner”, “group”, “others”.
- TOMOYO, on the other hand, allows minimal and customizable permissions to any group of specific UIDs.
- Example: users are app_1, app_2, app_3, app_4; some files owned by app_2 (uid=10002) need to be accessed by app_1 (uid=10001) also, but not by all the “others”.

```allow_read/write  @SOME_FILES  if  task.uid=10001-10002```
An example

We want to allow only the Browser to connect to Internet.

In this way any process running under "<kernel> /init /system/bin/app_process"
domain would be allowed to open TCP connection on any IP, port 80.

→ least-privilege principle violated
Solution

- TOMOYO Linux allows conditional ACL
- **Using task’s UID as a condition**, for access grant.

```
File Edit View Terminal Tabs Help
<<< Domain Policy Editor >>> 201 entries '? for help

<kernal> /init /system/bin/app_process
186: allow capability SYS_TIME
187: allow capability SYS_UNLINK
188: allow network TCP bind 0.0.0.0 0
189: allow network TCP connect 0.0.0.0-255.255.255.255 80 **if task.uid=@HTTP_USERS**
190: allow network TCP connect 74.125.153.113 443
191: allow network UDP bind 0.0.0.0 30372
```

In this way only the process with UID in HTTP_USERS group will be able to connect
Solution

- Add UID of browser application to HTTP_USERS group

In this way only browser will be able to connect
DEMO: Make policy for Web browser

- Web browser access to restrict the location
Saving access logs

- You can save access logs by starting ccs-auditd (host computer) as shown below.

```
/usr/sbin/ccs-auditd /tmp/grant_log /tmp/reject_log 127.0.0.1:10000
```

- You can create advanced policy settings from access logs.
Policy error handler

Similar to “page fault handler”

Access request

Permitted by policy?

NO

Access granted

YES

Permitted by handler?

NO

Access rejected

YES
Conclusions

- TOMOYO Linux suits well on Android
  - Will suits on other embedded devices as well
- MAC enforced for system services and user applications
  - Whole system or targeted applications
- Why not to try TOMOYO?
Thank you for your attention

Daisuke Numaguchi <numaguchid@nttddata.co.jp>
Tetsuo Handa <penguin-kernel@i-love.sakura.ne.jp>
Giuseppe La Tona <giuseppelatona@gmail.com>
Information

- Mailing list
  - English: tomoyo-users-en@lists.sourceforge.jp
  - Japanese: tomoyo-users@lists.sourceforge.jp
- Web site
  - http://tomoyo.sourceforge.jp/
- Wiki
  - http://elinux.org/TomoyoLinux
Copyrights

- Linux is a registered trademark of Linus Torvalds in Japan and other countries
- Android is a registered trademark of Google
- TOMOYO is a registered trademark of NTT Data Corporation in Japan
- Other names and trademarks are the property of their respective owners.