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)

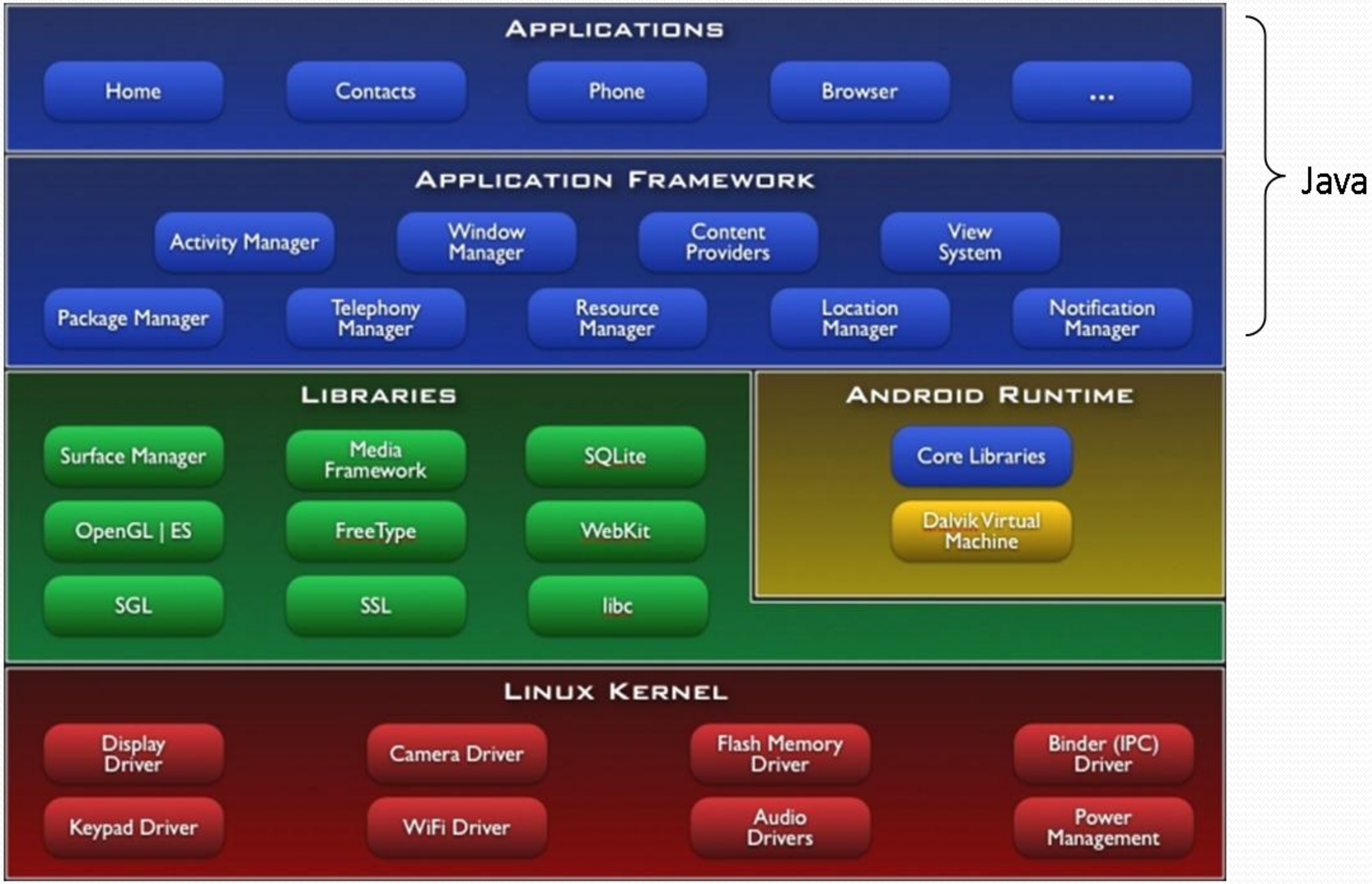
MAC(Mandatory Access Control)

- 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



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

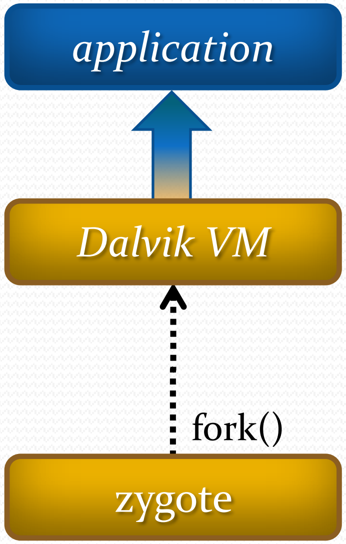
  (Biology concept of "zygote": duplicate, specialize and differentiate)

Dalvik and Zygote

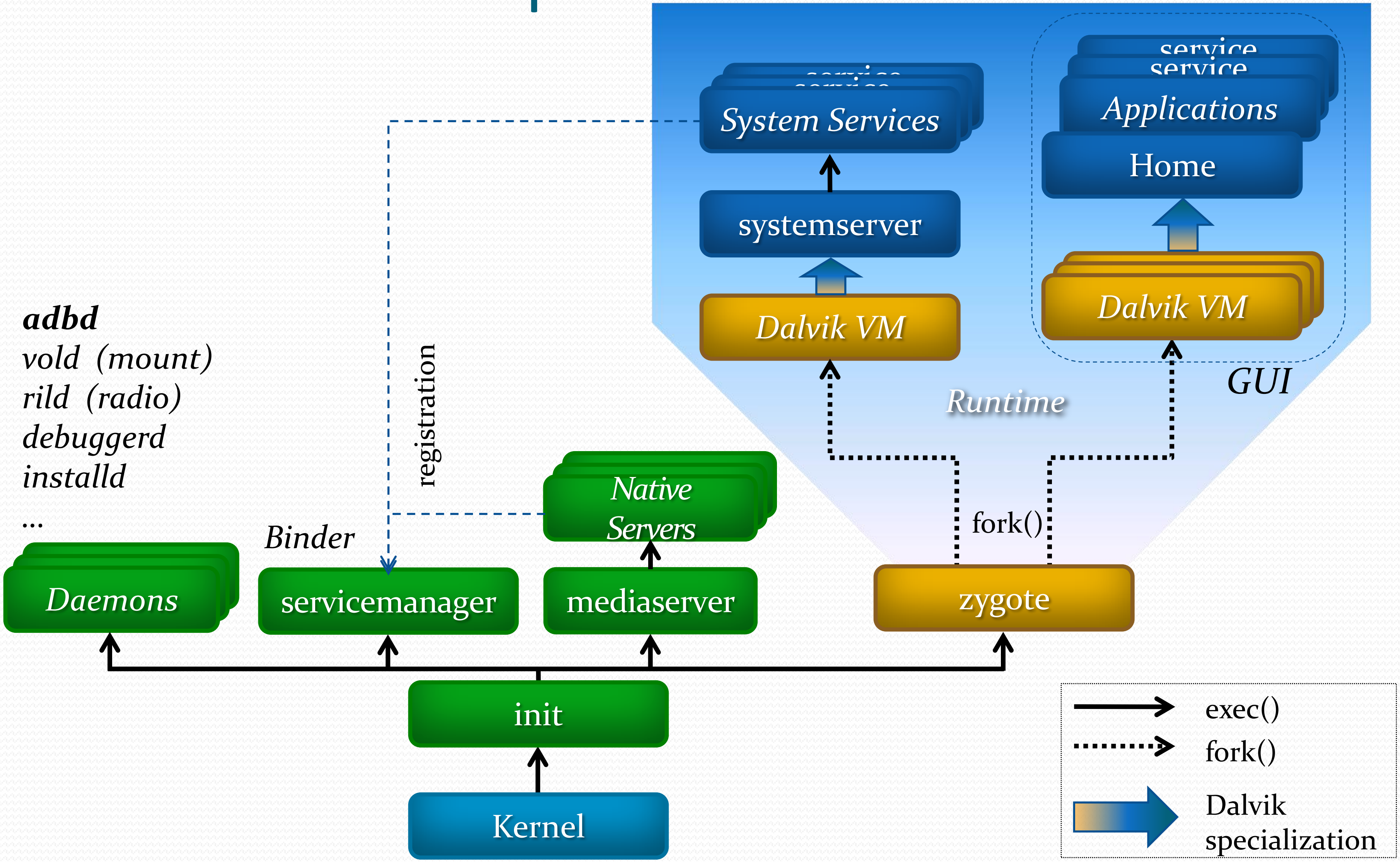
- Runtime is made by **Java programs running in *Dalvik***: Virtual Machine for mobile devices
  - slow CPU, small RAM, no swap space, battery
  - Not a JVM, no JIT: only interpreter of DEX (optimized bytecode obtained from Java .class)
  - Multiple VM instances can run efficiently.
- ***Zygote* process**: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

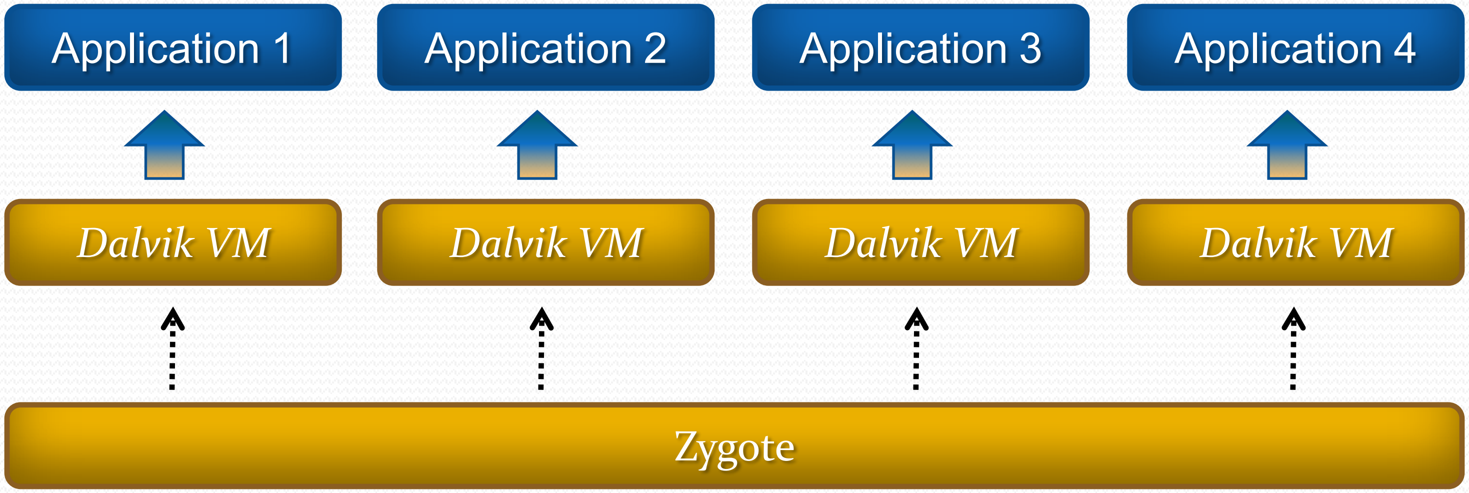


Android boot sequence



Android security model (1/2)

- Each application runs in its own process
  - Runtime in **separate instances** of Dalvik virtual machine



Android security model (2/2)

- 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

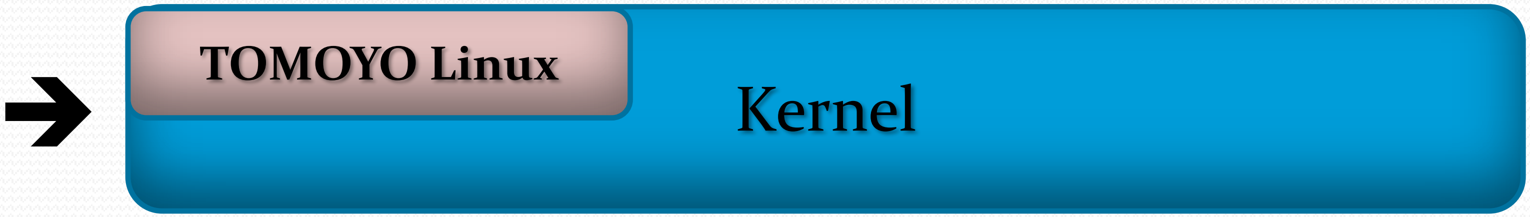
- Android SDK 1.6 ("donut") comes with kernel 2.6.29 .
- TOMOYO 2.x is available since kernel 2.6.30
- TOMOYO 2.2 function is only file access control
- ***So, choose TOMOYO 1.x !!***

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

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

to /init.rc

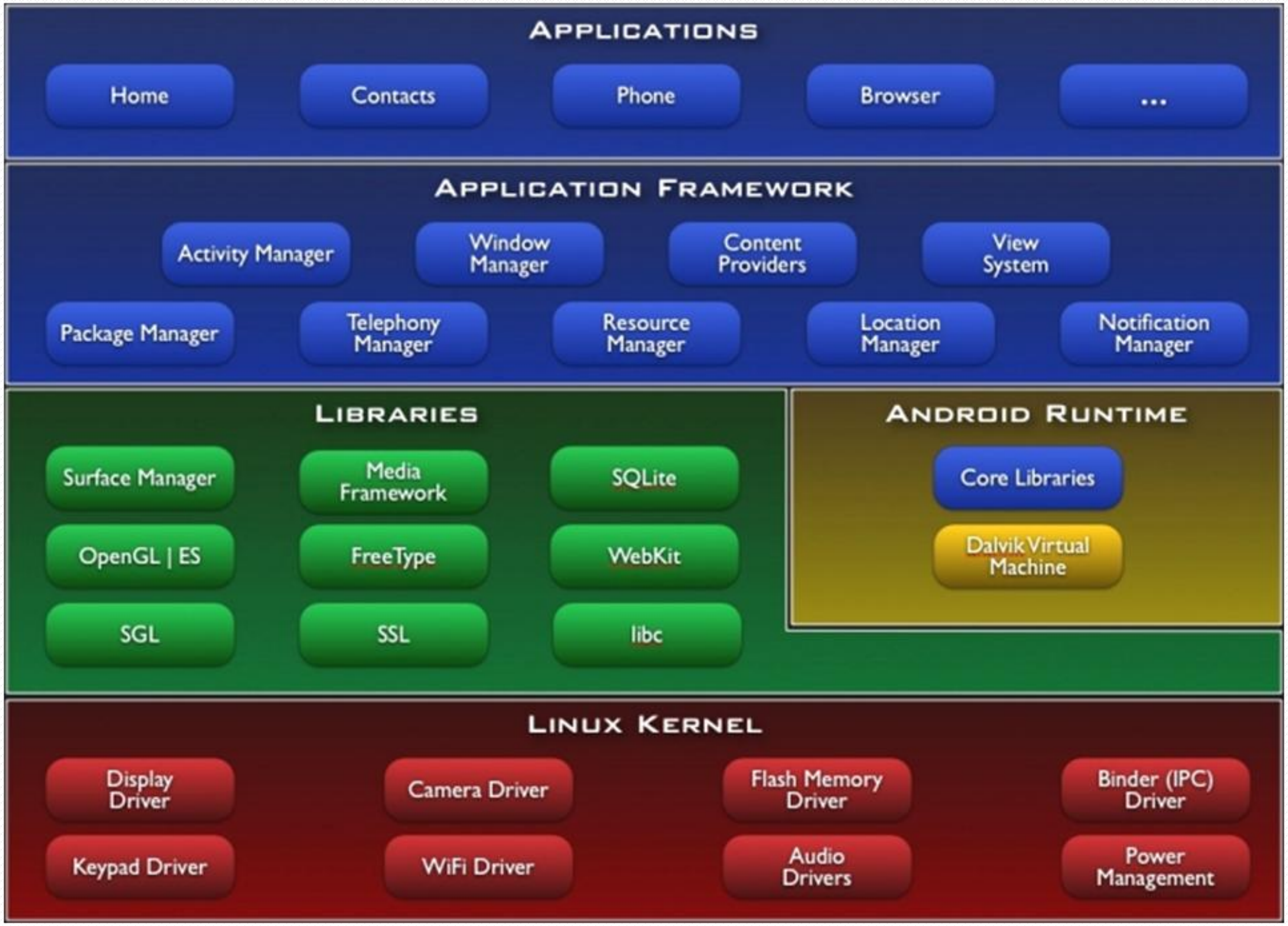
  - ccs-editpolicy-agent will listen to tcp port 7000
  - We can issue "adb forward tcp:10000 tcp:7000" to connect from host environment.

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



APPLICATIONS

Home
Contacts
Phone
Browser
...

APPLICATION FRAMEWORK

Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager

LIBRARIES

Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc

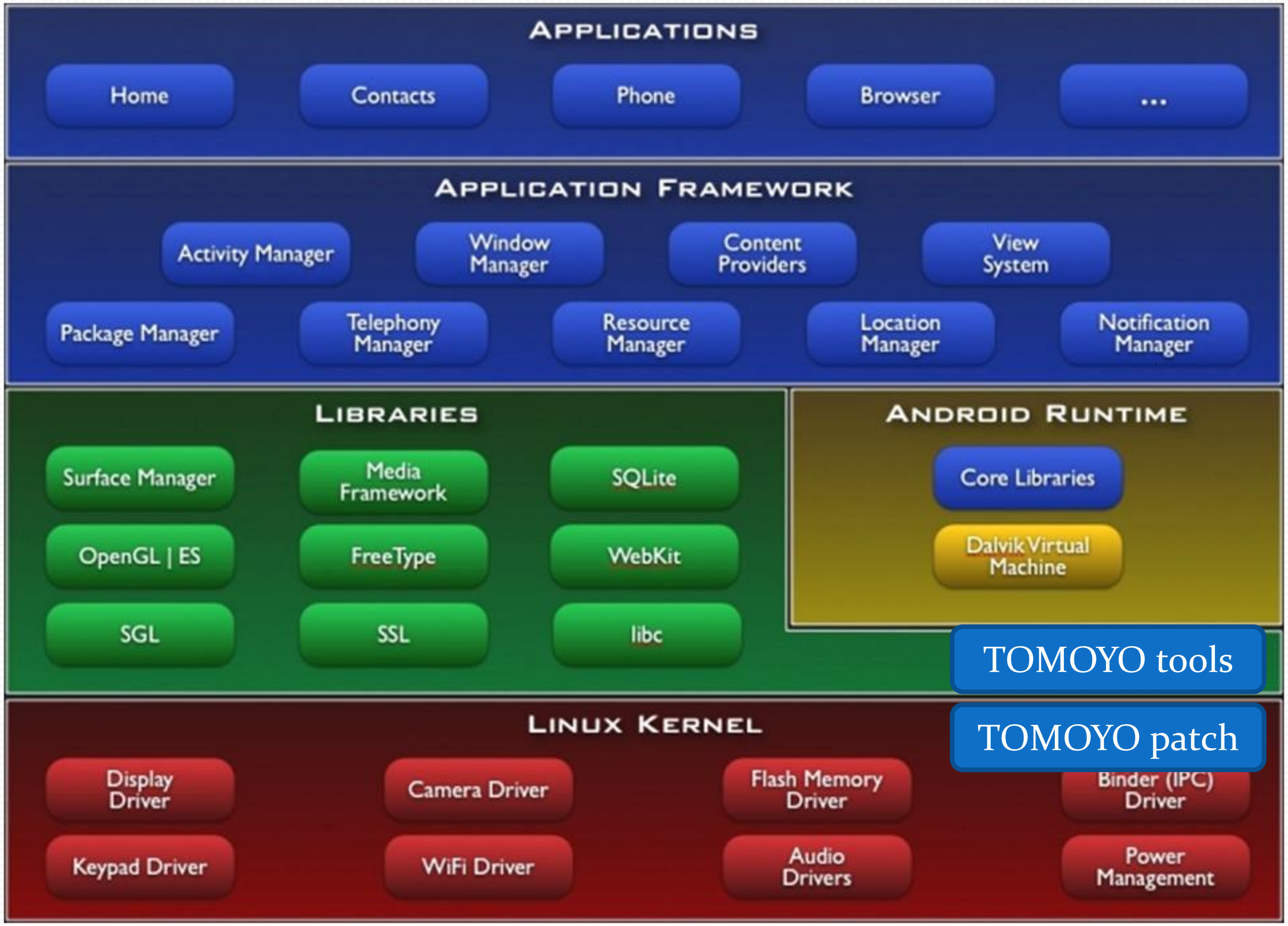
ANDROID RUNTIME

Core Libraries
Dalvik Virtual Machine

LINUX KERNEL

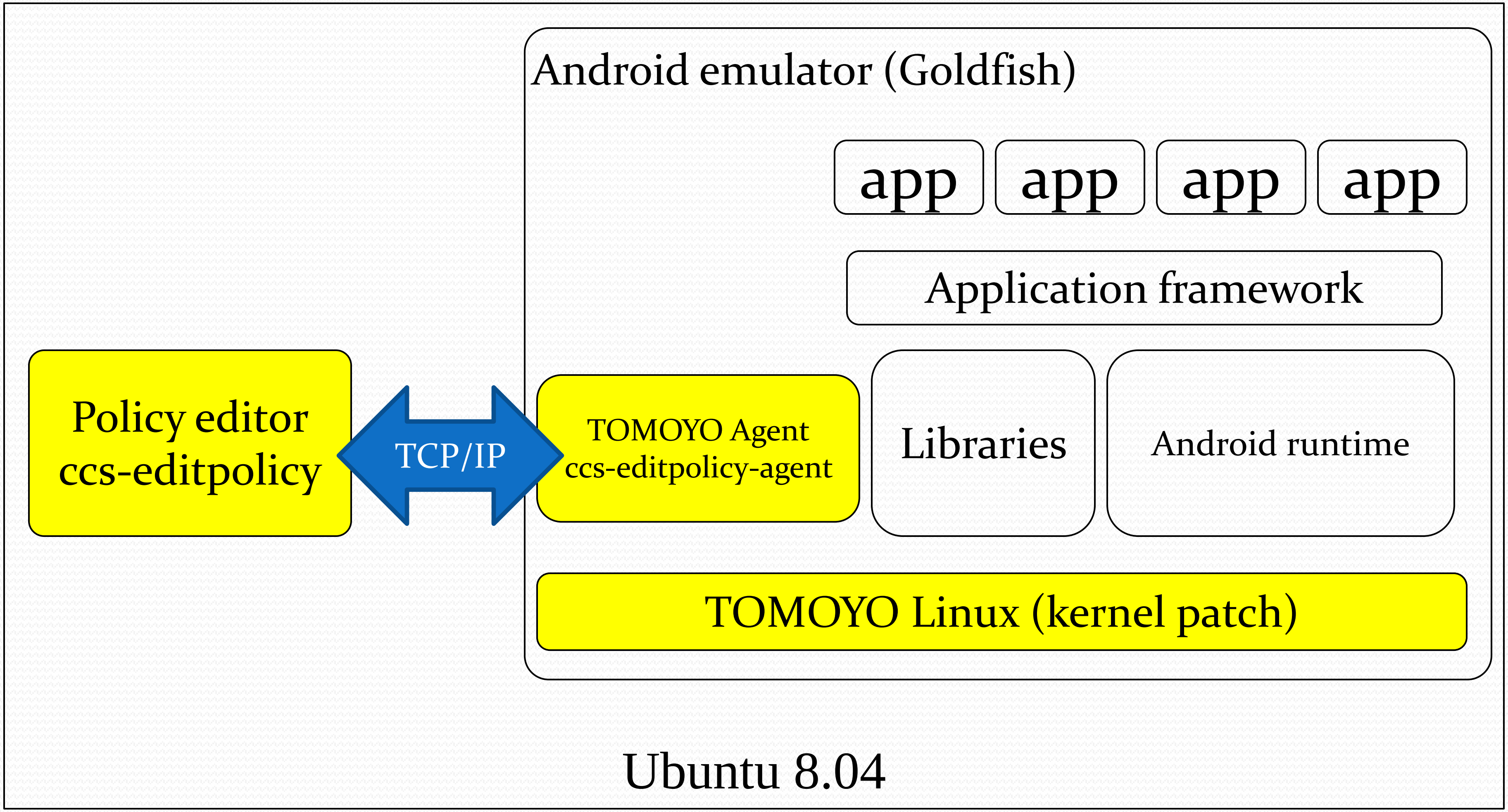
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

TOMOYO on Android overview



EDITING POLICY (VIA AGENT)

Environment



Android emulator (Goldfish)

app

app

app

app

Application framework

Policy editor
ccs-editpolicy

TCP/IP

TOMOYO Agent
ccs-editpolicy-agent

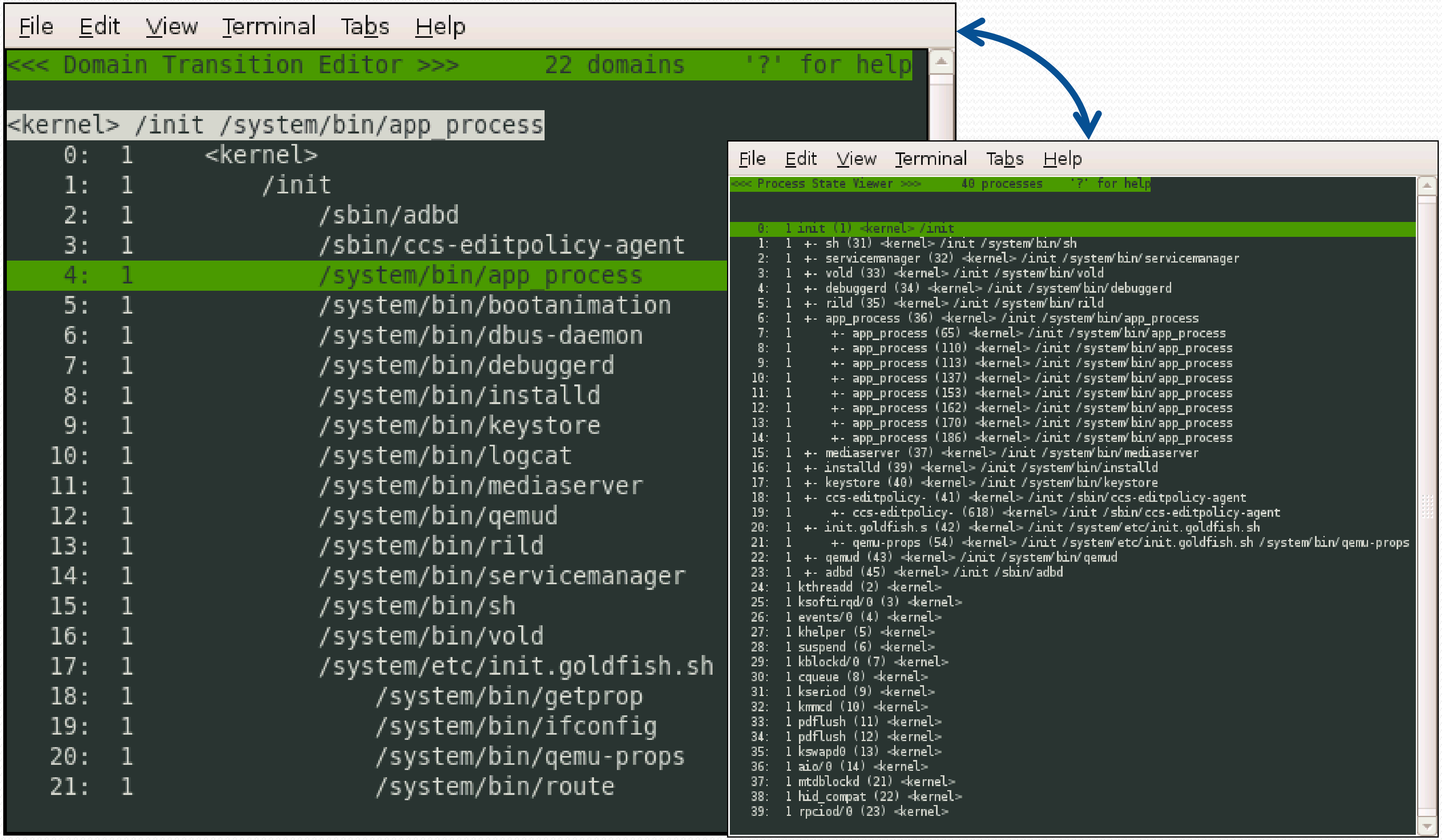
Libraries

Android runtime

TOMOYO Linux (kernel patch)

Ubuntu 8.04

Editpolicy



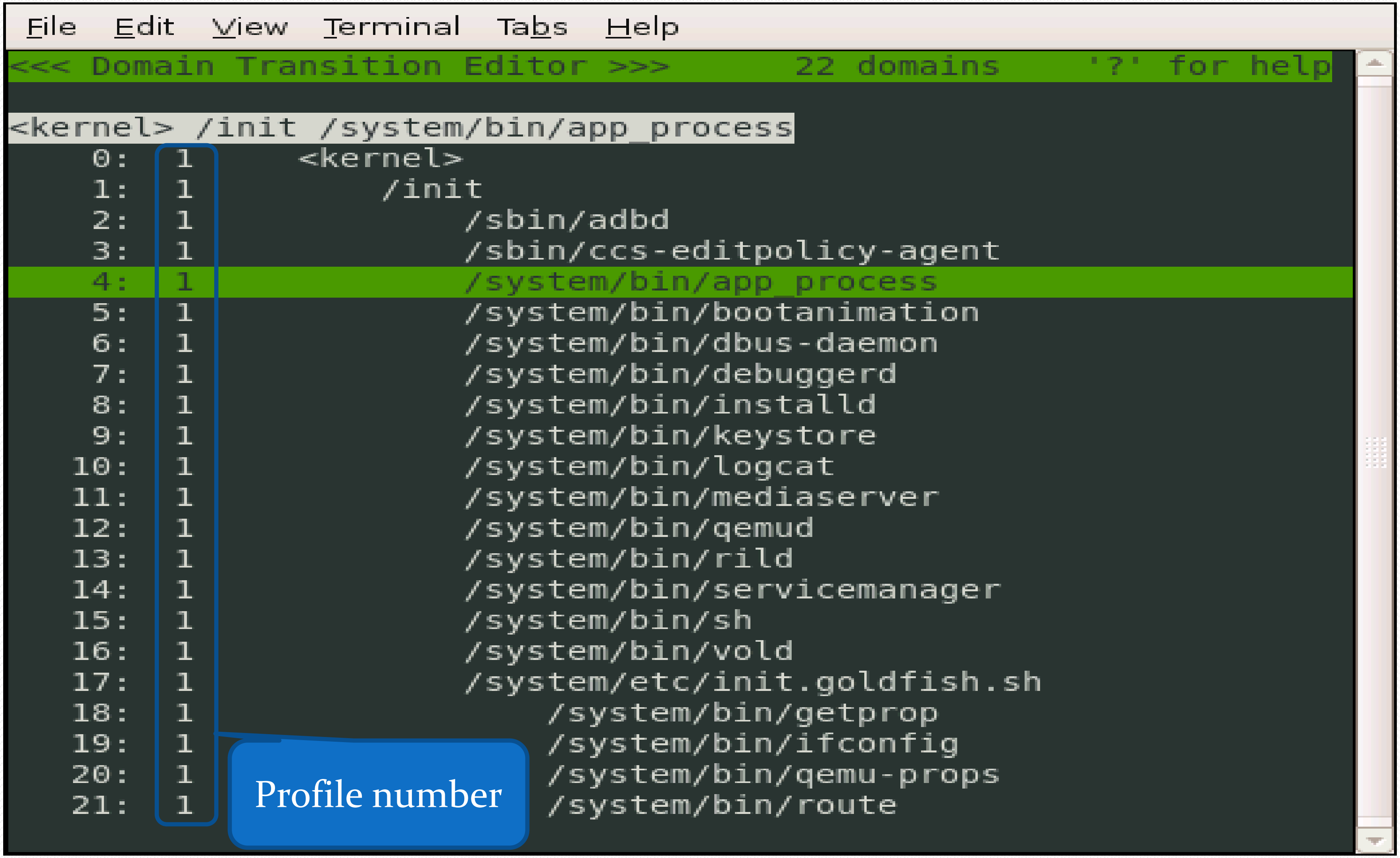
```
<<< Domain Transition Editor >>>      22 domains      '?' for help

<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/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
```

```
<<< Process State Viewer >>>      40 processes    '?' for help


   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  +- installd (39) <kernel> /init /system/bin/installd
  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  +- qemud (43) <kernel> /init /system/bin/qemud
  23:  1  +- adbd (45) <kernel> /init /sbin/adbd
  24:  1 kthreadd (2) <kernel>
  25:  1 ksoftirqd/0 (3) <kernel>
  26:  1 events/0 (4) <kernel>
  27:  1 khelper (5) <kernel>
  28:  1 suspend (6) <kernel>
  29:  1 kblockd/0 (7) <kernel>
  30:  1 cqueue (8) <kernel>
  31:  1 kseriod (9) <kernel>
  32:  1 kmmcd (10) <kernel>
  33:  1 pdflush (11) <kernel>
  34:  1 pdflush (12) <kernel>
  35:  1 kswapd0 (13) <kernel>
  36:  1 aio/0 (14) <kernel>
  37:  1 mtdblockd (21) <kernel>
  38:  1 hid_compat (22) <kernel>
  39:  1 rpciod/0 (23) <kernel>
```

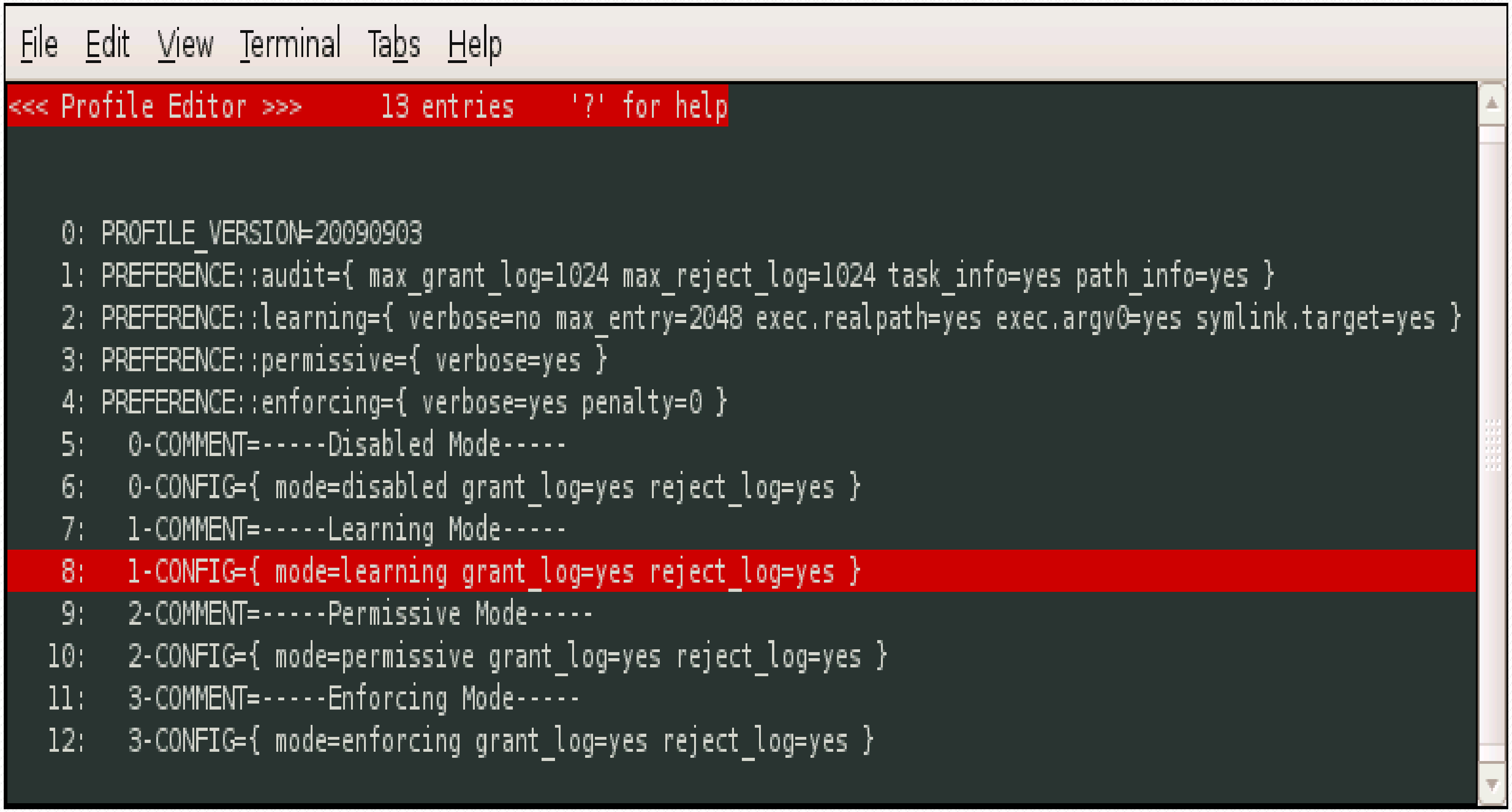
Domain transition tree



```
<<< Domain Transition Editor >>>      22 domains    '?' for help

<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/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



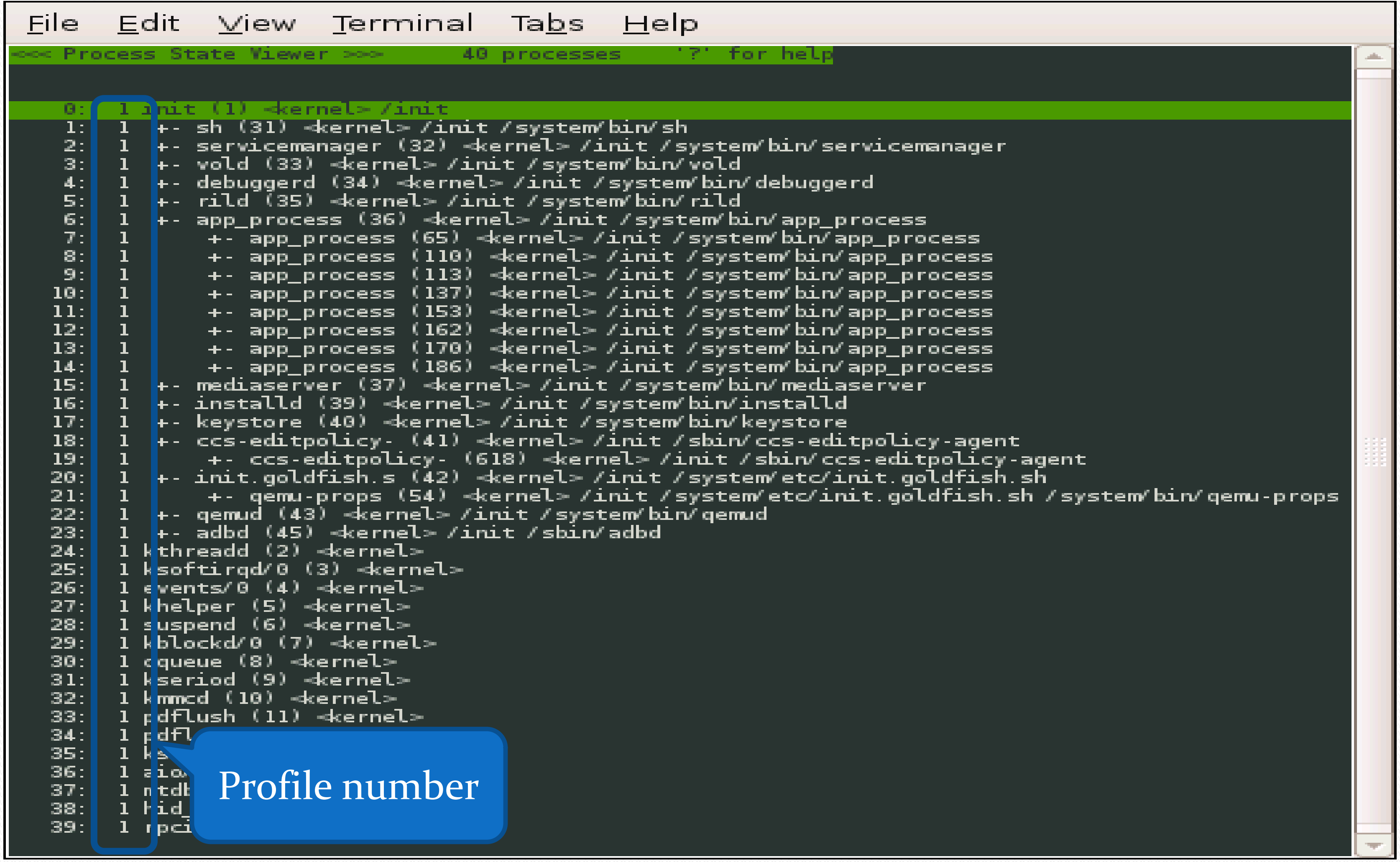
```
<<< Profile Editor >>>      13 entries    '?' for help



   0: PROFILE_VERSION=20090903
   1: PREFERENCE::audit={ max_grant_log=1024 max_reject_log=1024 task_info=yes path_info=yes }
   2: PREFERENCE::learning={ verbose=no max_entry=2048 exec.realpath=yes exec.argv0=yes symlink.target=yes }
   3: PREFERENCE::permissive={ verbose=yes }
   4: PREFERENCE::enforcing={ verbose=yes penalty=0 }
   5:   0-COMMENT=-----Disabled Mode-----
   6:   0-CONFIG={ mode=disabled grant_log=yes reject_log=yes }
   7:   1-COMMENT=-----Learning Mode-----
   8:   1-CONFIG={ mode=learning grant_log=yes reject_log=yes }
   9:   2-COMMENT=-----Permissive Mode-----
  10:   2-CONFIG={ mode=permissive grant_log=yes reject_log=yes }
  11:   3-COMMENT=-----Enforcing Mode-----
  12:   3-CONFIG={ mode=enforcing grant_log=yes reject_log=yes }
```

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

Process tree

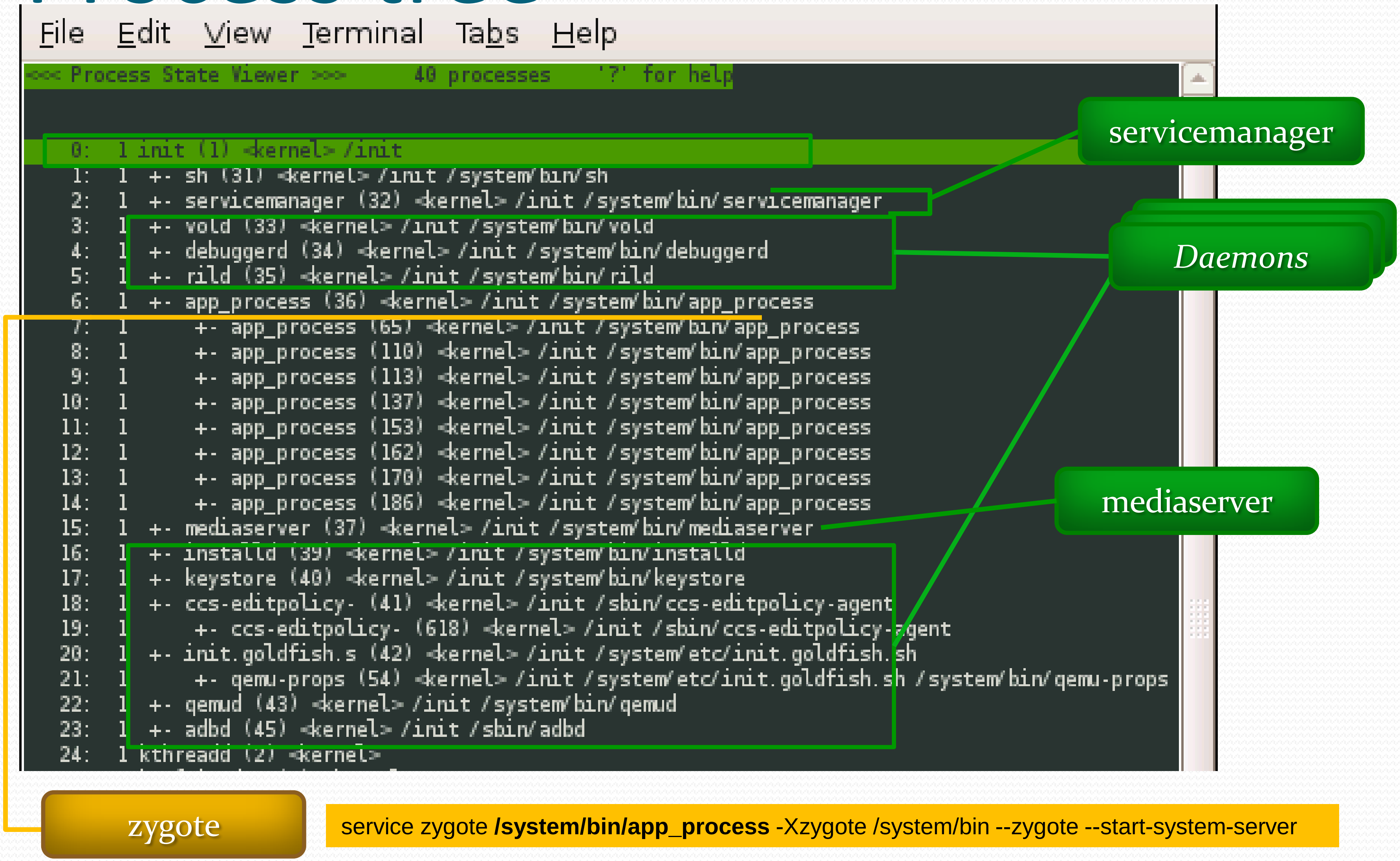


```
<<< Process State Viewer >>>          40 processes      '?' for help


   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  +- installd (39) <kernel> /init /system/bin/installd
  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  +- qemud (43) <kernel> /init /system/bin/qemud
  23:   1  +- adbd (45) <kernel> /init /sbin/adbd
  24:   1 kthreadd (2) <kernel>
  25:   1 ksoftirqd/0 (3) <kernel>
  26:   1 events/0 (4) <kernel>
  27:   1 khelper (5) <kernel>
  28:   1 suspend (6) <kernel>
  29:   1 kblockd/0 (7) <kernel>
  30:   1 cqueue (8) <kernel>
  31:   1 kseriod (9) <kernel>
  32:   1 kmmcd (10) <kernel>
  33:   1 pdflush (11) <kernel>
  34:   1 pdfl
  35:   1 ks
  36:   1 aio
  37:   1 mtdb
  38:   1 hid_
  39:   1 rpci
```

Profile number

Process tree



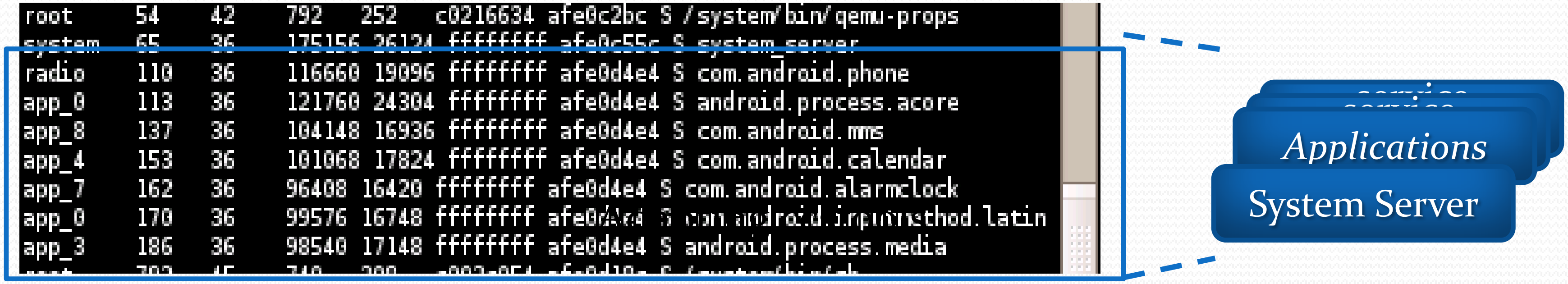
```
<<< Process State Viewer >>>       40 processes     '?' for help

  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  +- installd (39) <kernel> /init /system/bin/installd
 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  +- qemud (43) <kernel> /init /system/bin/qemud
 23:  1  +- adbd (45) <kernel> /init /sbin/adbd
 24:  1 kthreadd (2) <kernel>
```

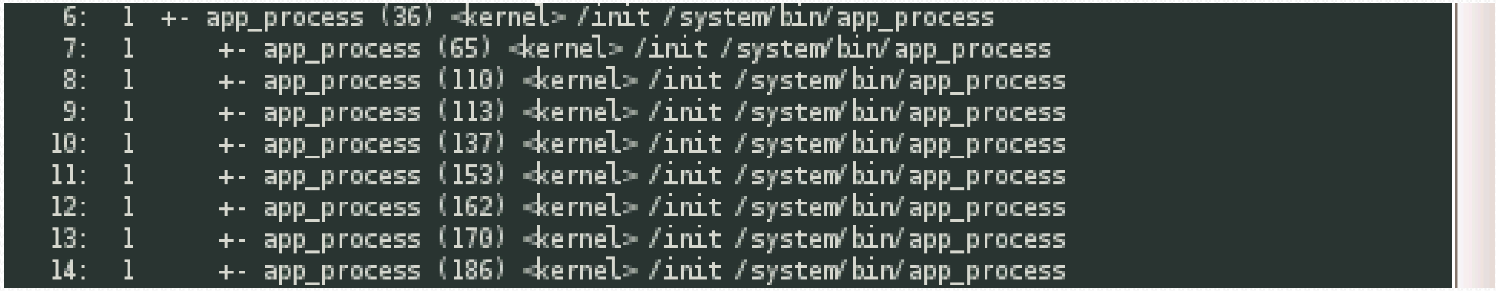
service zygote **/system/bin/app_process** -Xzygote /system/bin --zygote --start-system-server

Problem with splitting domains

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



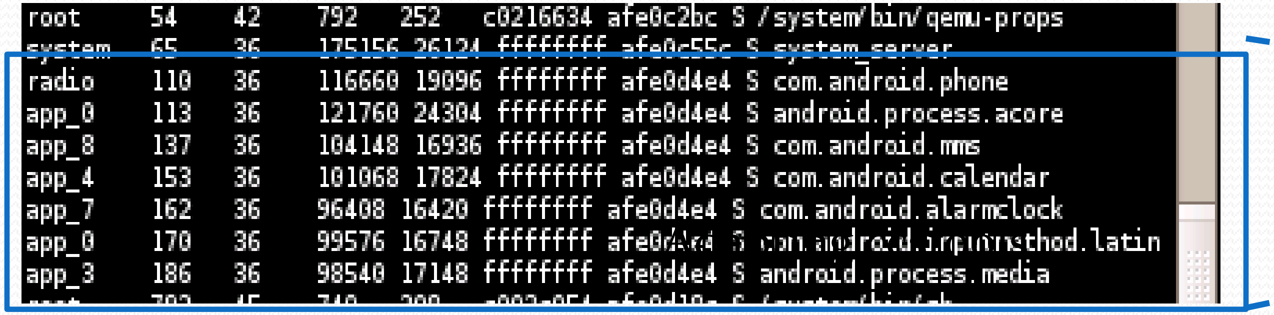
```
root      54    42    792   252   c0216634 afe0c2bc S /system/bin/qemu-props
system    65    36    175156 26124 ffffffff afe0c55c S system_server
radio     110   36    116660 19096 ffffffff afe0d4e4 S com.android.phone
app_0     113   36    121760 24304 ffffffff afe0d4e4 S android.process.acore
app_8     137   36    104148 16936 ffffffff afe0d4e4 S com.android.mms
app_4     153   36    101068 17824 ffffffff afe0d4e4 S com.android.calendar
app_7     162   36    96408 16420 ffffffff afe0d4e4 S com.android.alarmclock
app_0     170   36    99576 16748 ffffffff afe0d4e4 S com.android.inputmethod.latin
app_3     186   36    98540 17148 ffffffff afe0d4e4 S android.process.media
```



```
  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
```

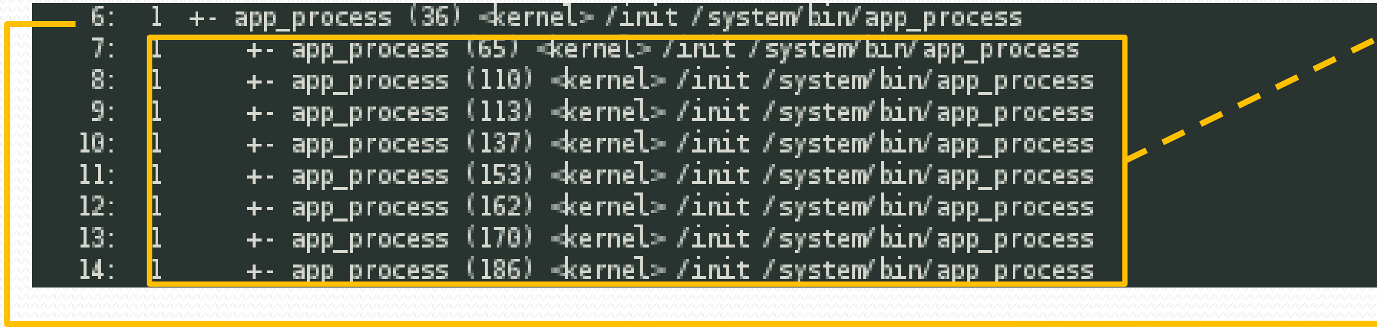
Problem with splitting domains

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

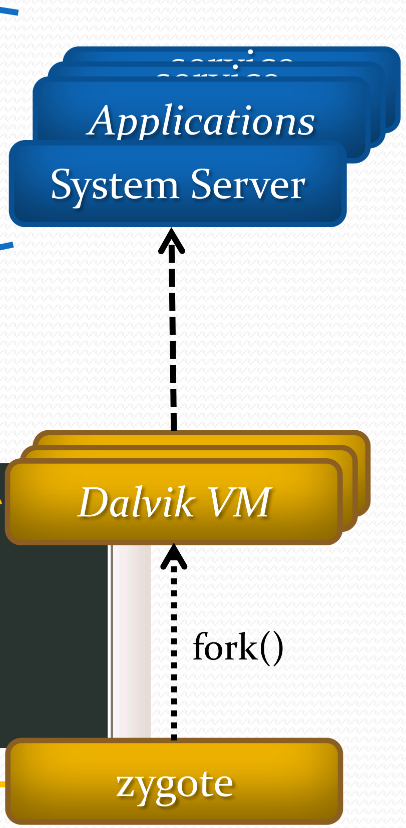


```
root      54    42     792    252    c0216634 afe0c2bc S /system/bin/qemu-props
system    65    36     175156 26124  ffffffff afe0c55c S system_server
radio     110   36     116660 19096 ffffffff afe0d4e4 S com.android.phone
app_0     113   36     121760 24304 ffffffff afe0d4e4 S android.process.acore
app_8     137   36     104148 16936 ffffffff afe0d4e4 S com.android.mms
app_4     153   36     101068 17824 ffffffff afe0d4e4 S com.android.calendar
app_7     162   36     96408  16420 ffffffff afe0d4e4 S com.android.alarmclock
app_0     170   36     99576  16748 ffffffff afe0d4e4 S com.android.inputmethod.latin
app_3     186   36     98540  17148 ffffffff afe0d4e4 S android.process.media
```

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



```
 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
```



Problem with splitting domains

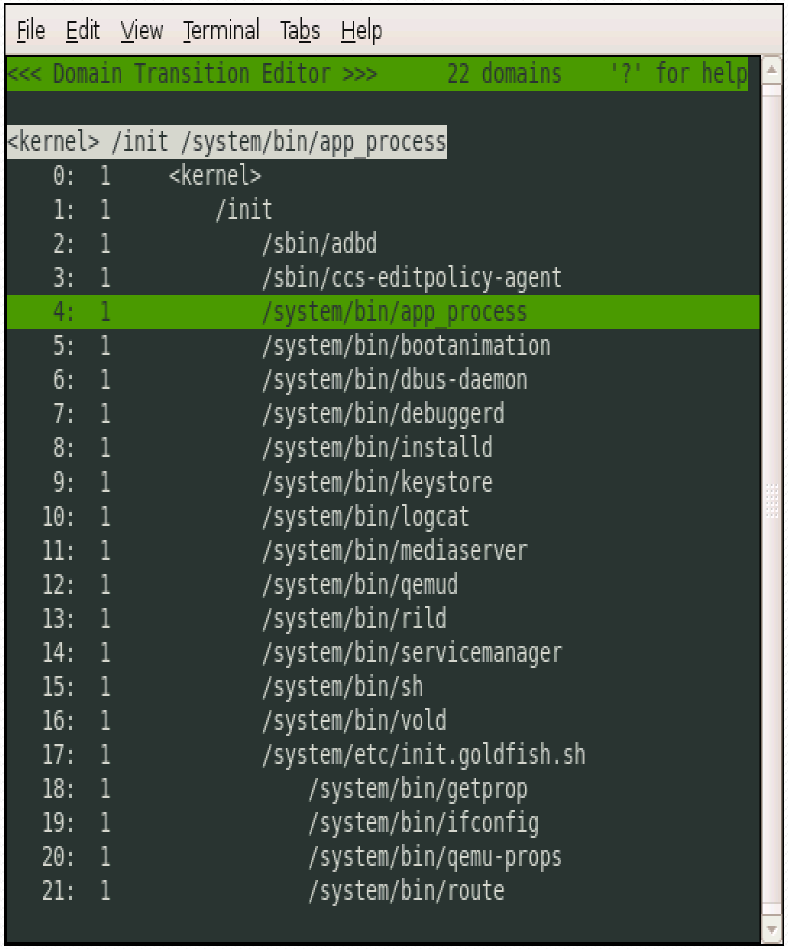
- New and unexpected situation for TOMOYO Linux

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

→Splitting domain

<kernel> /init /system/bin/app_process

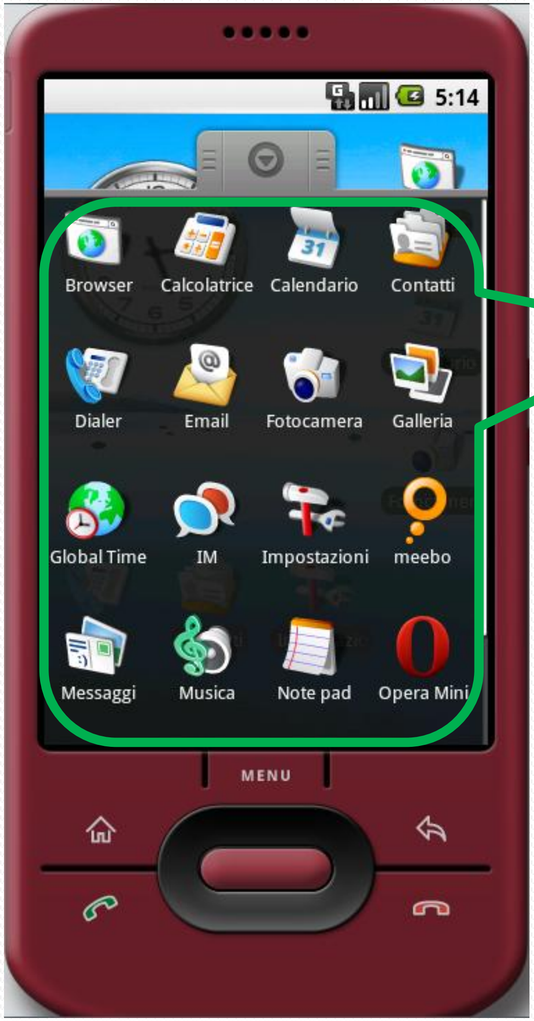
in different domains according to each single application is impossible. . . ?

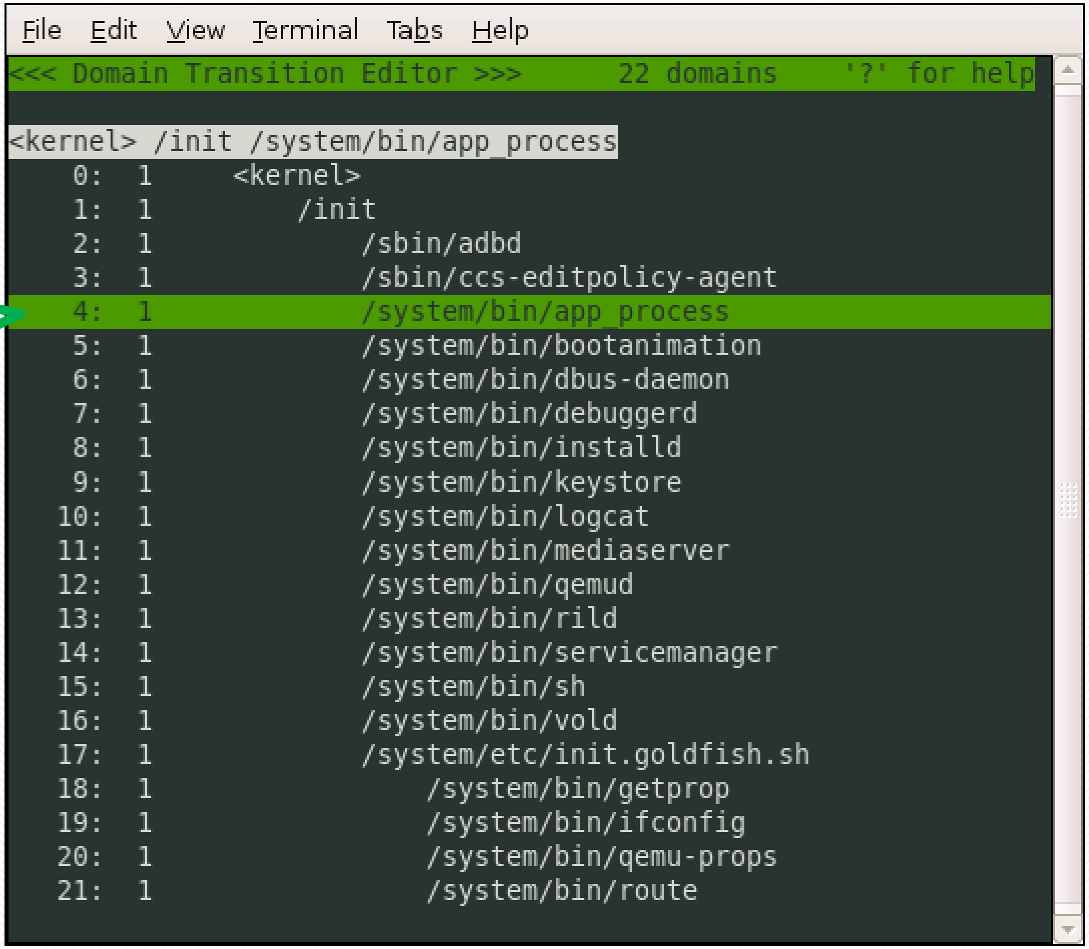


```
<<< Domain Transition Editor >>>      22 domains     '?' for help

<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/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
```

Problem with splitting domains





```
<<< Domain Transition Editor >>>         22 domains     '?' for help

<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/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
```

<kernel> /init /system/bin/app_process

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
```

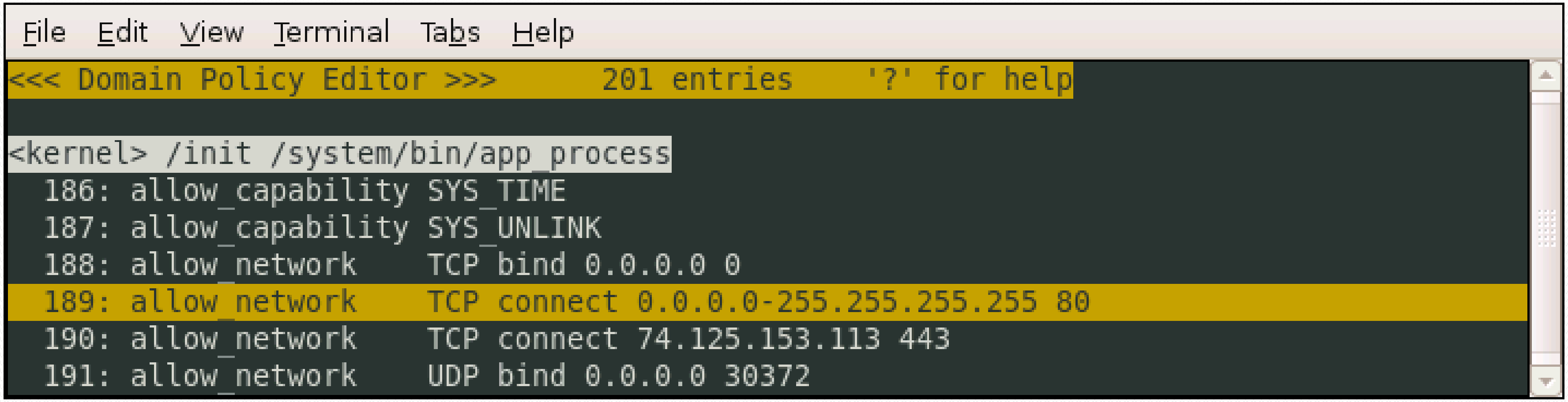
TOMOYO’s MAC and Android DAC

- 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.



```
<<< Domain Policy Editor >>>      201 entries     '?' for help

<kernel> /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
  190: allow_network    TCP connect 74.125.153.113 443
  191: allow_network    UDP bind 0.0.0.0 30372
```

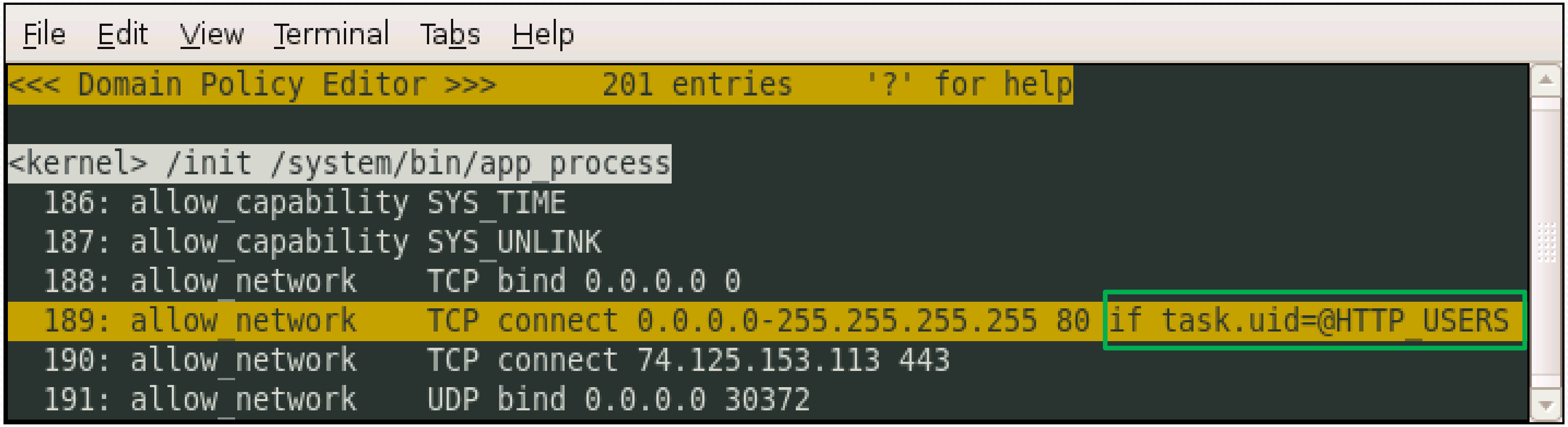
In this way **any** process running under
   “<kernel> /init /system/app/process”

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.



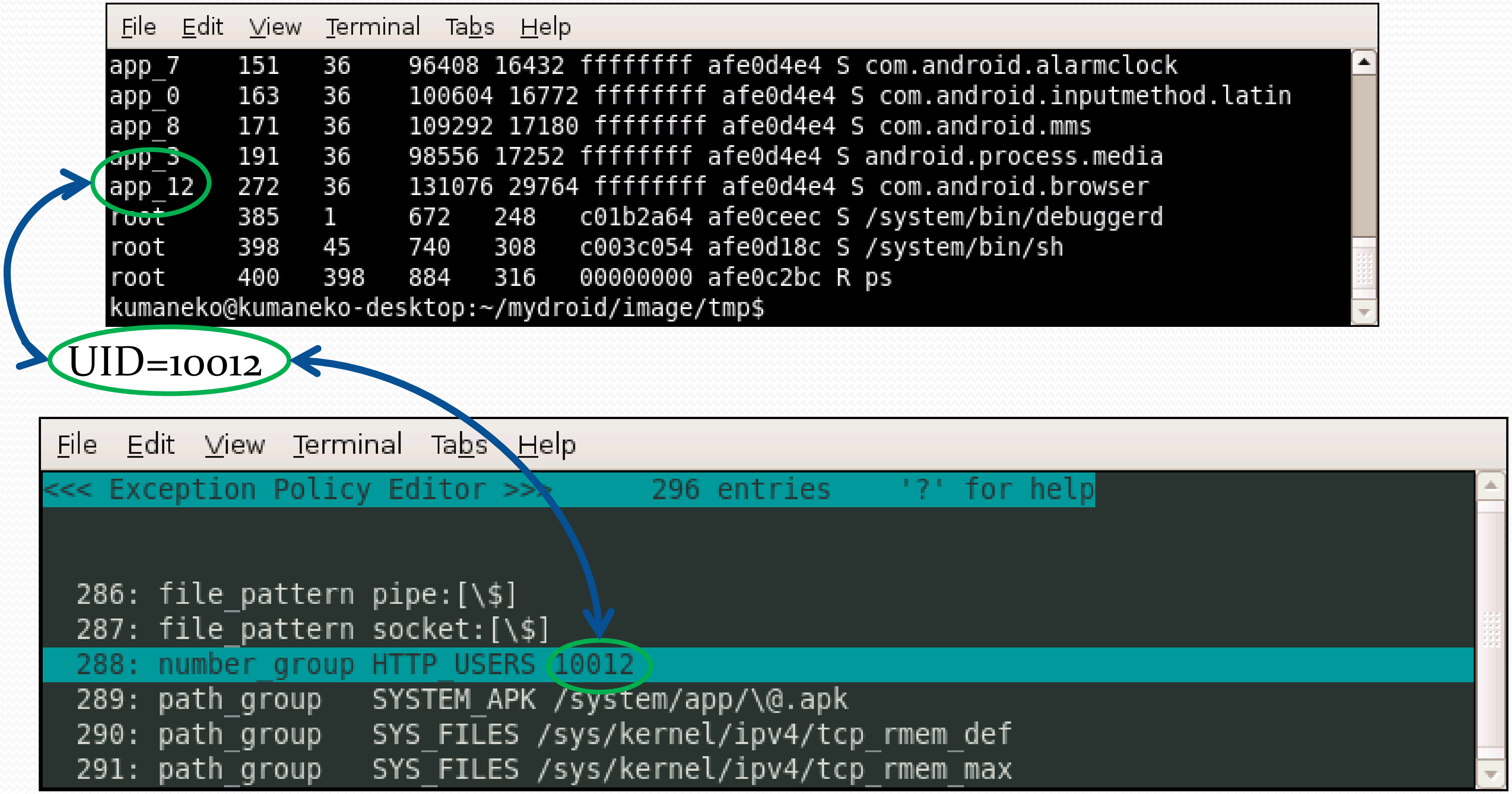
```
<<< Domain Policy Editor >>>      201 entries     '?' for help

<kernel> /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**



File Edit View Terminal Tabs Help
UID=10012
File Edit View Terminal Tabs Help

```
app_7     151   36    96408 16432 ffffffff afe0d4e4 S com.android.alarmclock
app_0     163   36    100604 16772 ffffffff afe0d4e4 S com.android.inputmethod.latin
app_8     171   36    109292 17180 ffffffff afe0d4e4 S com.android.mms
app_3     191   36    98556 17252 ffffffff afe0d4e4 S android.process.media
app_12    272   36    131076 29764 ffffffff afe0d4e4 S com.android.browser
root      385   1     672    248   c01b2a64 afe0ceec S /system/bin/debuggerd
root      398   45    740    308   c003c054 afe0d18c S /system/bin/sh
root      400   398   884    316   00000000 afe0c2bc R ps
kumaneko@kumaneko-desktop:~/mydroid/image/tmp$
```

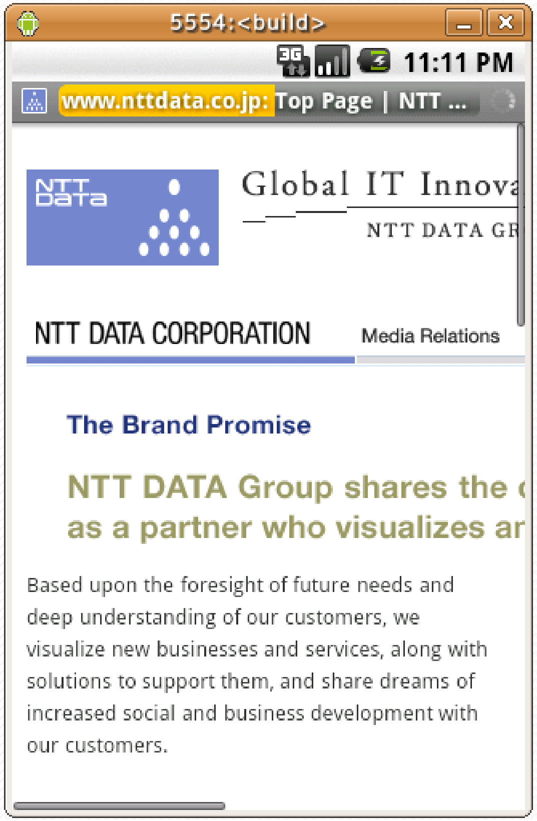
```
<<< Exception Policy Editor >>>        296 entries     '?' for help


  286: file_pattern pipe:[\$]
  287: file_pattern socket:[\$]
  288: number_group HTTP_USERS 10012
  289: path_group   SYSTEM_APK /system/app/\@.apk
  290: path_group   SYS_FILES /sys/kernel/ipv4/tcp_rmem_def
  291: path_group   SYS_FILES /sys/kernel/ipv4/tcp_rmem_max
```

In this way **only** browser will be able to connect

DEMO: Make policy for Web browser

- Web browser access to restrict the location



5554:<build>

11:11 PM

www.nttdata.co.jp: Top Page | NTT ...

Global IT Innova

NTT DATA GR

NTT DATA CORPORATION

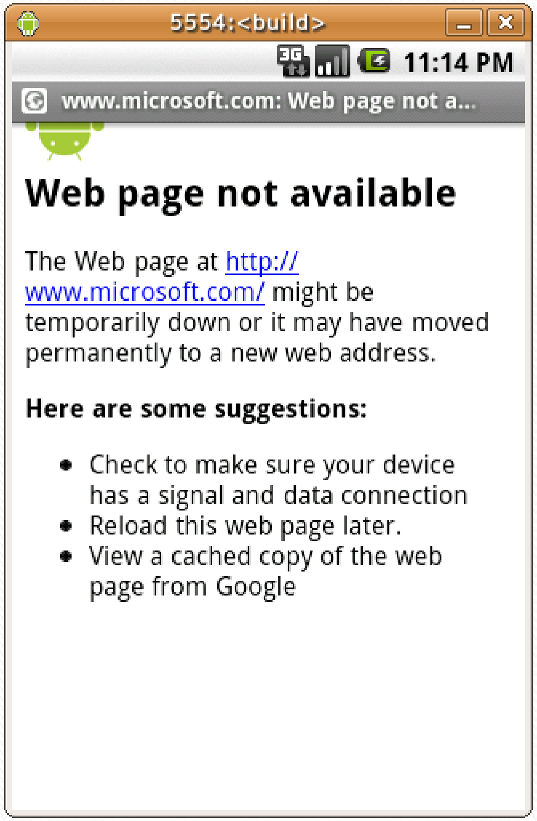
Media Relations

The Brand Promise

NTT DATA Group shares the

as a partner who visualizes ar

Based upon the foresight of future needs and deep understanding of our customers, we visualize new businesses and services, along with solutions to support them, and share dreams of increased social and business development with our customers.



5554:<build>

11:14 PM

www.microsoft.com: Web page not a...

Web page not available

The Web page at http://www.microsoft.com/ might be temporarily down or it may have moved permanently to a new web address.

Here are some suggestions:

- Check to make sure your device has a signal and data connection
- Reload this web page later.
- View a cached copy of the web page from Google

Saving access logs

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

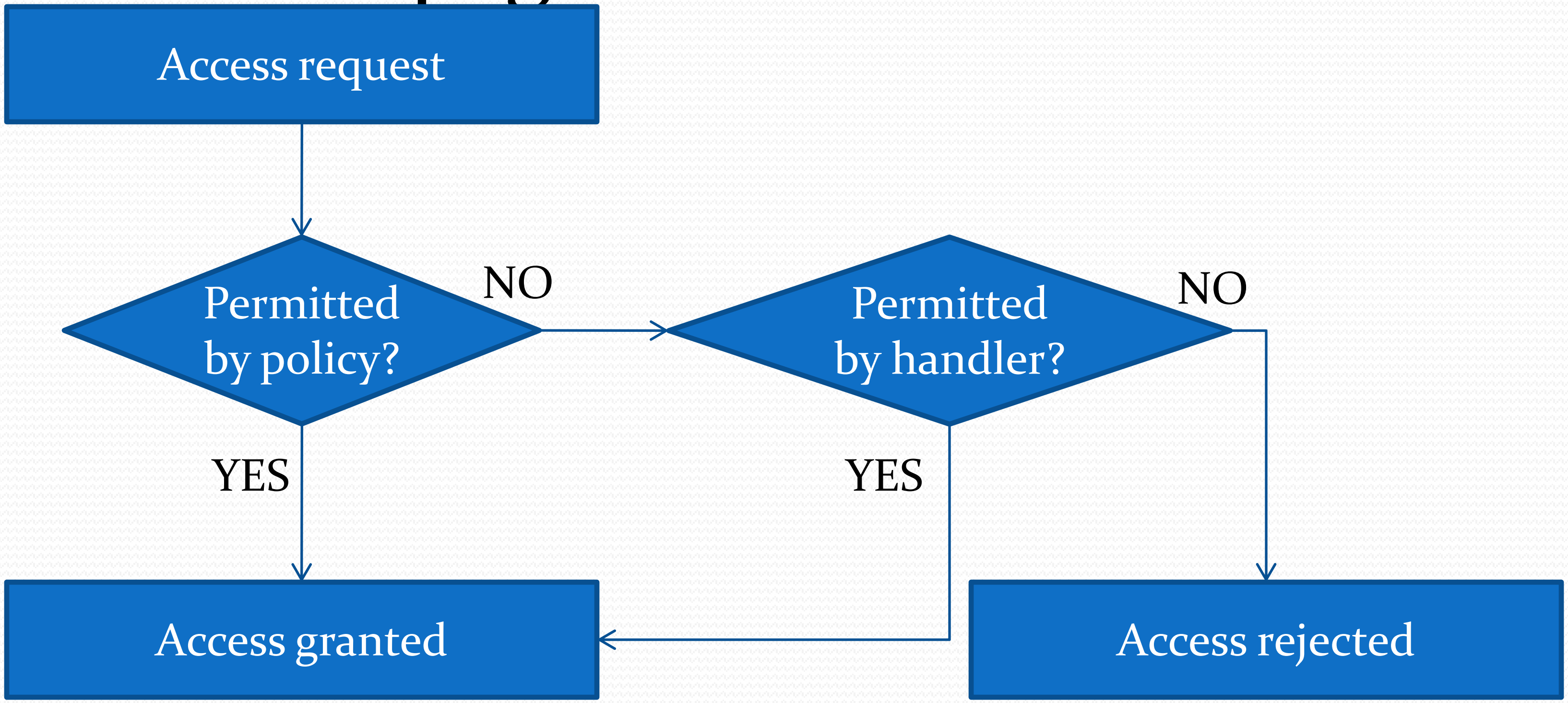
```
/usr/sbin/ccs-auditd   /tmp/grant_log  /tmp/reject_log   127.0.0.1:10000
```

```
#2009-10-19 10:07:15# profile=1 mode=learning (global-pid=36) task={ pid=36 ppid=1 uid=0 gid=0 euid=0 egid=0
suid=0 sgid=0 fsuid=0 fsgid=0 state[0]=0 state[1]=0 state[2]=0 type!=execute_handler } path1={ uid=0 gid=2000 in
o=537 major=31 minor=0 perm=0755 type=file } path1.parent={ uid=0 gid=2000 ino=468 perm=0755 } exec={ real
path="/system/bin/app_process" argc=5 envc=10 argv[]={ "/system/bin/app_process" "-Xzygote" "/system/bin" "--z
ygote" "--start-system-server" } envp[]={ "PATH=/sbin:/system/sbin:/system/bin:/system/xbin" "LD_LIBRARY_PAT
H=/system/lib" "ANDROID_BOOTLOGO=1" "ANDROID_ROOT=/system" "ANDROID_ASSETS=/system/app" "A
NDROID_DATA=/data" "EXTERNAL_STORAGE=/sdcard" "BOOTCLASSPATH=/system/framework/core.jar:/syst
em/framework/ext.jar:/system/framework/framework.jar:/system/framework/android.policy.jar:/system/framework/s
ervices.jar" "ANDROID_PROPERTY_WORKSPACE=9,32768" "ANDROID_SOCKET_zygote=10" } }
<kernel> /init
allow_execute /system/bin/app_process
```

- You can create advanced policy settings from access logs.

Policy error handler

- Similar to "page fault handler"



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@nttdata.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.