Securing Android-Powered Mobile Devices Using SELinux

This paper appears in: Security & Privacy, IEEE **Issue Date:** May-June 2010 **Volume:** 8 Issue:3 **On page(s):** 36 - 44

Asaf Shabtai, Yuval Fledel, and Yuval Elovici
Ben-Gurion University

Present:陳政宇
Date:2011/12/14

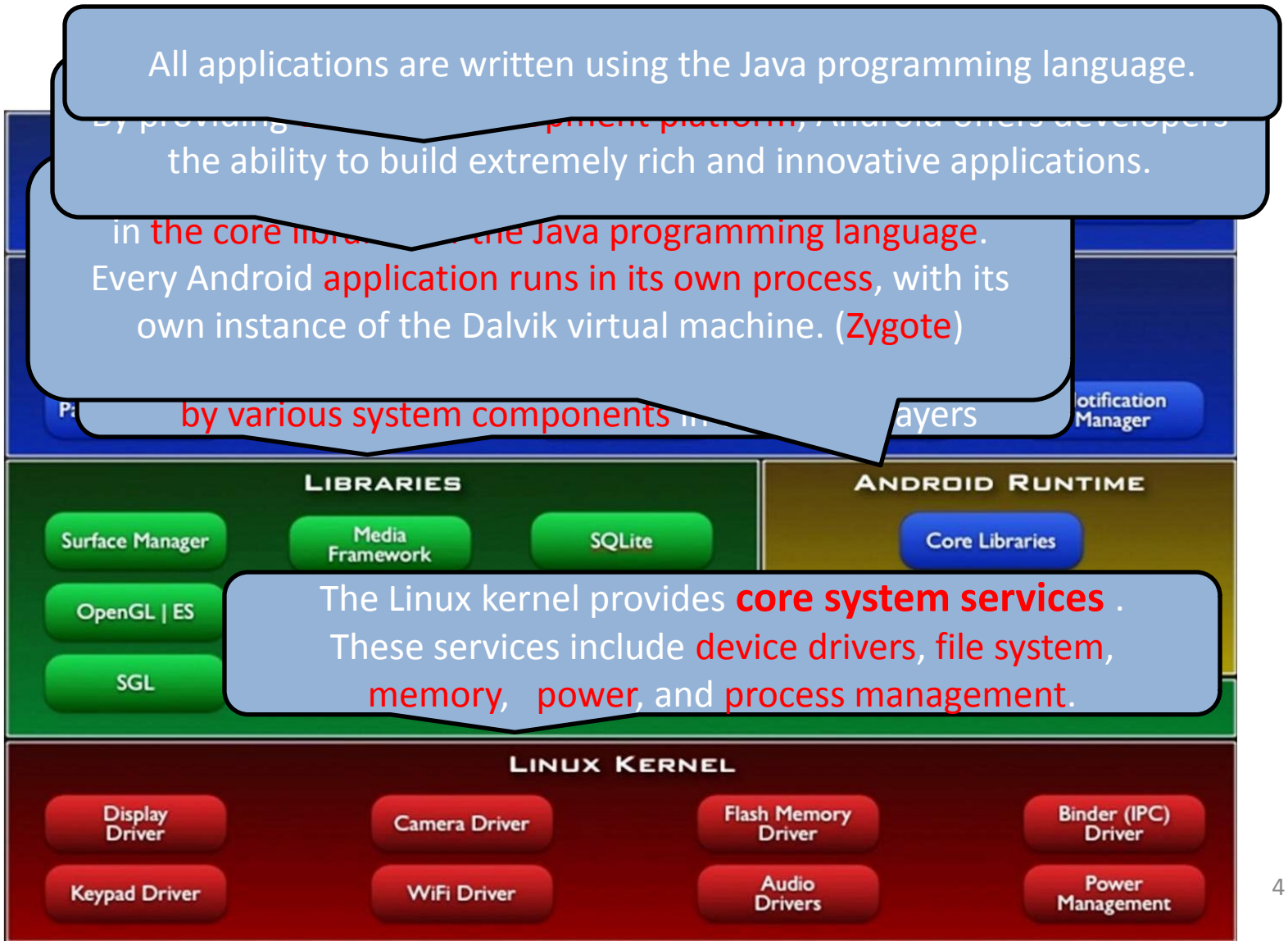
Outline

- 1.Introduction
- 2.The Android Framwork
- 3.Security in Android
- 4.Security-Enhanced Linux
- 5.Using SELinux in Android
- 6.Integrating SELinux in Android
- 7.Benchmarking
- 8.Conclusion

1.Introduction

- Smart phones
  - various PC-like services
- Smart phones are vulnerable to attacks
  - Bluetooth、Wi-Fi、3G、USB etc.
- Security mechanisms for mobile phones
  - Antimalware and antispam、Host-based IDS、fiewall etc.
- cellular phone owners don't count smart phones as regular computers
- Linux-based OS Security Issues
  - privilege escalation attacks
- Linux Security Modules (LSM)
  - low-level, fine-grained access control capabilities
  - specific actions and have limited access to unnecessary resources
- We integrated the Security-Enhanced Linux LSM and evaluated its ability to improve Android's security.

2.The Android Framework



3.Security in Android(1/2)

- Android Access control mechanisms
  - Users mechanism
    - Owner and group
  - file permissions mechanism
    - Execute、Write、Read
    - **system files** are owned by either the system or root user
    - **application files** are owned by an application-specific user
  - application-permission mechanism
    - specific operations that an application can perform.
    - roughly 100 built-in permissions to control operations
      - dialing (**CALL_PHONE**),Internet (**INTERNET**),key strokes (**READ_INPUT_STATE**)
    - Any Android application can declare additional permissions.

3.Security in Android(2/2)

- Android's security mechanisms are insufficient and too coarse-grained to tackle this security issue.
  - EX : INTERNET permissions
  - The file-permission mechanism protects files, but not from a root user.
- LSM is an additional solution. It supports various access control models .

4.Security-Enhanced Linux(1/2)

- SELinux is the best-known LSM.
  - under the GNU GPL on December 22, 2000
  - US National Security Agency
  - Based on principle of least privilege
- Abilities
  - SELinux can limit a process's access to files even if it runs with the root user.
  - SELinux can do so by limiting actions
    - EX : INTERNET can't load and replace kernel

4.Security-Enhanced Linux (2/2)

DEMO

5.Using SELinux in Android

- We present several scenarios demonstrating SELinux's usefulness in Android.

1) Protecting Processes
   - Installd
     - Prevent attacker from gaining root privileges .
2) Protecting Files
   - access control
     - prevent unauthorized users from confidential files.
3) Protecting the System
   - mechanism
     - prevent the security mechanism itself from being overridden.

6.Integrating SELinux in Android (1/4)

```
on init
    loglevel 3
    # SELinux must be initialized very early
    mkdir / selinux
    mount selinuxfs selinuxfs / selinux
    loadpolicy / se-policy
    write /proc/1/attr/current system_u:system_r:init_early_t
    chcon / init system_u:object_r:init_exec_t
    chcon /sbin/abdb system_u:object_r:adbd_exec_t
    chcon /dev/null system_u:object_r:devnull_t
    chcon /dev/ ashmem system_u:object_r:ashmem_t
```

(a)

- enc

1) An

   – s

     •

2) An ol for Loa

   – Not loading the policy on boot or init process

     • added three new commands to init.rc

       – Loadpolicy 、chcon 、context

6.Integrating SELinux in Android (2/4)

3) Creating a Custom SELinux Policy for Android
   - The default SELinux reference policy is irrelevant on Android:
     - it assumes a Linux standard base layout
     - it's too big for an embedded system.
   - Solutions
     - construct a policy without using the reference policy .
     - Remove irrelevant modules (Apache)
4) Android's File System Doesn't Support Extended Attributes
   - yaffs2 doesn't support extended attributes (xattrs)
     - used the chcon command to set xattrs on memory file systems.
5) It's Difficult to Apply SELinux Policy to Dalvik Processes
   - Dalvik limits SELinux's applicability
     - altering the zygote code

6.Integrating SELinux in Android (3/4)

- Steps in Porting SELinux to Android
  - 1. compile kernel with selinux support
  - 2. design an Android-specific security policy
  - 3. We modified the init process code and init.rc script to support additional commands to load the policy at system startup and set initial labels.
  - 4. We built a new disk image containing the updated init and policy files and updated it on the device.

6.Integrating SELinux in Android (4/4)

```
1 allow kernel_t rootfs_t:dir { search };
2 allow init_early_t tmpfs_t:filesystem { mount };
3 allow init_t init_t:process { fork setpgid };
4 allow init_t init_t:unix_stream_socket { create bind };
5 allow init_t tmp_t:sock_file { create setattr unlink };
6 allow init_t adbd_t:process { transition };
7 allow adbd_t devnull_t:chr_file { read write };
8 allow adbd_t ashmem_t:chr_file { read write };
9 type_transition init_t adbd_exec_t:process adbd_t;
```

(b)

```
# ps -Z
      PID CONTEXT                    STAT    COMMAND
1     system_u:system_r:init_t       S       /init
2     system_u:system_r:kernel_t     SW<     [ kthreadd ]
3     system_u:system_r:kernel_t     SW<     [ ksoftirqd / 0 ]
4     system_u:system_r:kernel_t     SW<     [events0/]
```

7.Benchmarking(1/4)

- I/O bandwidth、latency、CPU consumption、memory footprint.
- Device
  - HTC G1
  - Qualcomm MSM7201A
  - 192 Mbytes of RAM
  - internal flash storage of 256 Mbytes
- Software
  - Kernel 2.6.25.rc30
- The evaluation tools
  - bonnie++
    - file system and disk performance benchmark
  - Lmbench
    - a set of microbenchmarks for low-level Linux functionalities

7.Benchmarking(2/4)

Table 1. Evaluation results.

Measure	Without SELinux	With SELinux	Slowdown (%)
bonnie++			
Seq output, Chr (KBps)	52.3 ± 1.26	62.76 ± 1.89	–16.66*
Seq output, Chr (% CPU)	96.82 ± 0.92	96.33 ± 0.89	–0.50
Seq output, block (KBps)	3,674.64 ± 15.76	3,682.06 ± 25.41	–0.20
Seq output, block (% CPU)	13.42 ± 0.5	13.79 ± 0.48	+2.71
Seq output, rewrite (KBps)	2,623.7 ± 5.7	2,624.27 ± 7.64	–0.02
Seq output, rewrite (% CPU)	11.7 ± 0.47	11.94 ± 0.24	+2.07
Seq input, Chr (KBps)	205.45 ± 16.44	157.85 ± 12.97	+30.16*
Seq input, Chr (% CPU)	99 ± 0	98.97 ± 0.17	–0.03
Seq input, block (KBps)	8,914.91 ± 10.22	8,895.91 ± 10.54	+0.21*
Seq input, block (% CPU)	20.3 ± 0.47	20.61 ± 0.5	+1.49
Random, seeks (KBps)	104.75 ± 2.93	103.48 ± 2.29	+1.22
Random, seeks (% CPU)	41.82 ± 2.32	43.88 ± 2.52	+4.93
Seq output, Chr, latency (ms)	340.48 ± 110.83	326.27 ± 109.23	–4.17
Seq output, block, latency (ms)	1,368.21 ± 70.93	1,332.24 ± 108.47	–2.63
Seq output, rewrite, latency (ms)	2,175.7 ± 107.89	2,132.18 ± 234.56	–2.00
Seq input, Chr, latency (ms)	47.66 ± 4.16	60.29 ± 4.64	+26.51*
Seq input, block, latency (ms)	34.72 ± 14.59	33.99 ± 14.17	–2.11
Random, seeks, latency (ms)	225.45 ± 37.23	234.88 ± 36.81	+4.18

7.Benchmarking(3/4)

Table 1. Evaluation results.

Measure	Without SELinux	With SELinux	Slowdown (%)
Lmbench			
Simple syscall (μs)	4.83 ± 0.05	4.78 ± 0.05	–0.86
Simple read (μs)	7.84 ± 0.12	9.64 ± 0.14	+22.96*
Simple write (μs)	7.25 ± 0.83	10.02 ± 1.72	+38.13*
Simple stat (μs)	97.39 ± 2.03	184.95 ± 5.21	+89.91*
Simple fstat (μs)	12.81 ± 0.18	27.9 ± 1.81	+117.75*
Simple open/close (μs)	139.19 ± 5.36	261.5 ± 6.75	+87.87*
Signal handler installation (μs)	6.64 ± 0.05	6.49 ± 0.02	–2.22
Signal handler overhead (μs)	39.29 ± 0.51	44.25 ± 1.82	+12.63*
Protection fault (μs)	6.77 ± 1.12	1.81 ± 1.31	–73.19*
Pagefault (μs)	146.09 ± 1.13	149.27 ± 3.05	+2.17*
Pipe bandwidth (MBps)	39.85 ± 1.01	38.61 ± 0.3	+3.23*
Pipe latency (μs)	312.91 ± 8.38	454.06 ± 13.66	+45.11*
AF_UNIX sock stream bandwidth (MBps)	43.95 ± 1.09	42.7 ± 0.5	+2.93*
AF_UNIX sock stream latency (μs)	408.68 ± 14.03	677.62 ± 14.25	+65.81*
Context switches, size = 16k (2)	724.01 ± 8.77	826.33 ± 10.83	+14.13*
Context switches, size = 16k (4)	1,057.49 ± 15.33	1,147.86 ± 28.63	+8.55*
Context switches, size = 16k (8)	1,107.91 ± 13.48	1,190.78 ± 23.16	+7.48*
Context switches, size = 16k (16)	1,120.88 ± 19.38	1,200.32 ± 20.3	+7.09*
Context switches, size = 16k (24)	1,106.83 ± 17.98	1,188.16 ± 22.52	+7.35*
Context switches, size = 16k (32)	1,099.9 ± 25.15	1,178.78 ± 21.03	+7.17*
Context switches, size = 16k (64)	1,077.3 ± 31.04	1,157.98 ± 35.53	+7.49*
Context switches, size = 16k (96)	1,054.5 ± 33.86	1,126.61 ± 49.04	+6.84*

7.Benchmarking(4/4)

Table 1. Evaluation results.

Measure	Without SELinux	With SELinux	Slowdown (%)
Disk and memory usage			
Free RAM	53,177 ± 828 Kbytes	51,945 ± 929 Kbytes	–2.32
Init file size	98,260 bytes	98,296 bytes	+0.04
Init.rc	8,630 bytes	9,469 bytes	+9.72
Kernel size	1,379,032 bytes	1,477,188 bytes	+7.11
Binary policy	N/A	17,400 bytes	N/A

8.Conclusion

- Implementing SELinux in Android hardens the Android system and enforces low-level access control
- This path is a low-cost, high-gain solution.
- future works
  - additional evaluation of SELinux
  - other LSMs such as Smack and AppArmor.