Android patching

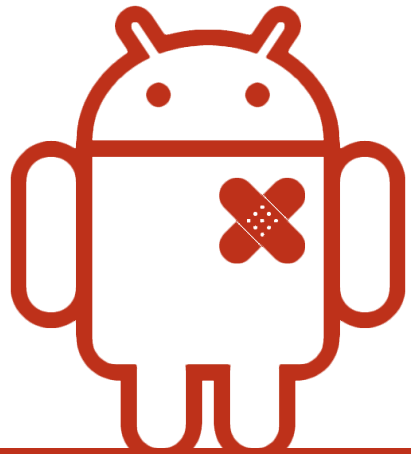
From a Mobile Device Management perspective

Cedric Van Bockhaven

cbockhaven@os3.nl

System and Network Engineering – RP2
University of Amsterdam

July 2nd, 2014



Outline

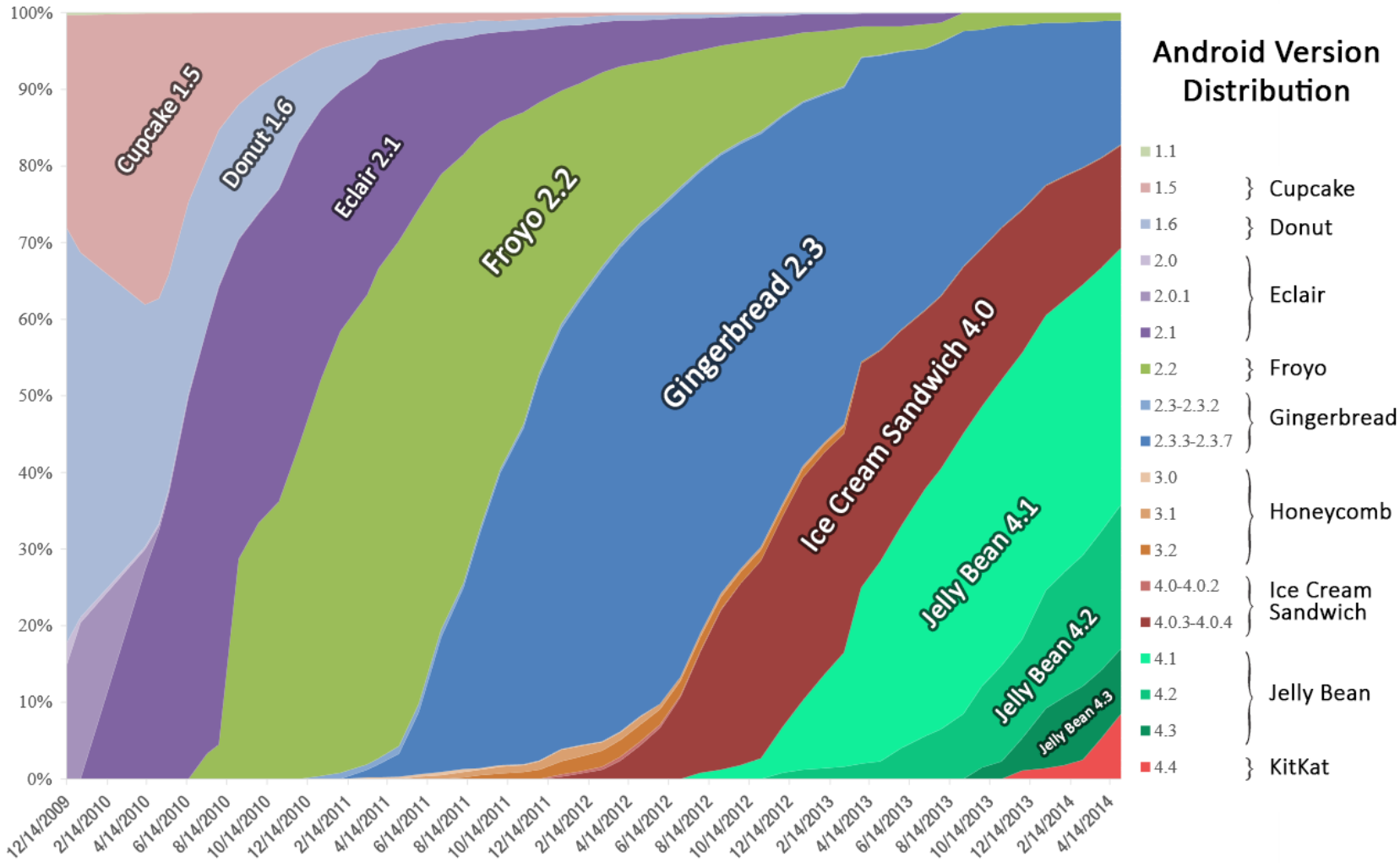
Introduction

- 80% of smartphones run Android
- Yet only 26% of devices in a BYOD setting run Android
- Different Android versions, ROMs, kernels, hardware

BYOD – Bring your own device

Numbers by Joost Kremers [2], TechCrunch `http://goo.gl/FJKHC6`, and Fjmustak's Android version history

Introduction



Introduction

- Older Android versions 2.3+ still omnipresent
- Responsibility of the vendors to push updates
- Many devices remain unpatched and vulnerable

→ Out-of-band update mechanism needed that doesn't rely on the vendor.

Research question

Main research question:

> **Is it possible to patch security vulnerabilities in Android devices through the MDM?**

MDM – Mobile Device Management solution

Related work

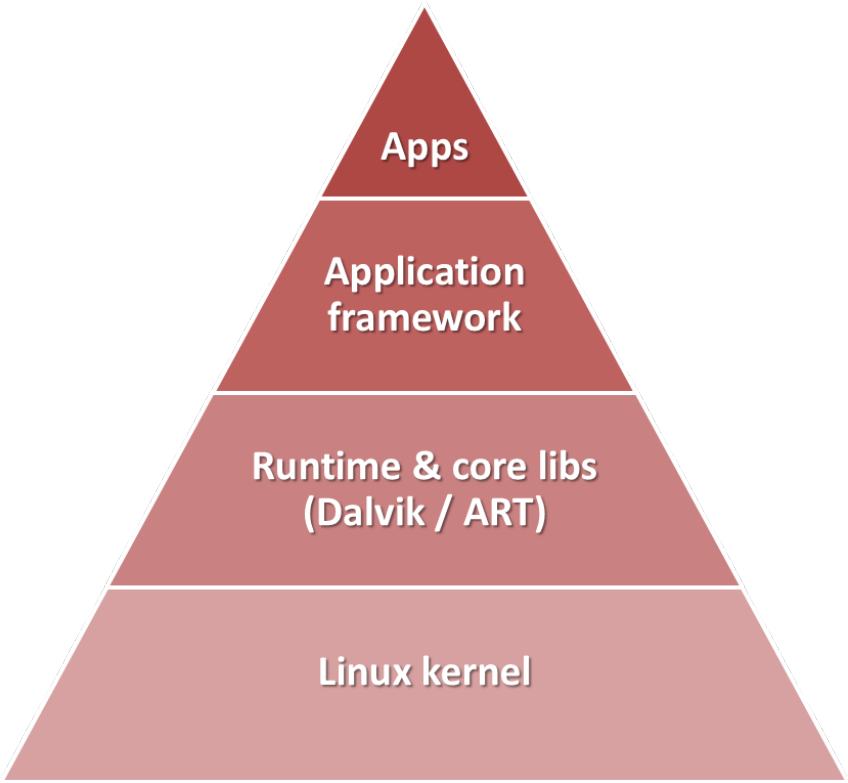
PatchDroid: Scalable Third-Party Security Patches for Android Devices.

Collin Mulliner, Jon Oberheide, William Robertson, and Engin Kirda.

In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 259–268. ACM, 20143.

Architecture

- Kernel vulnerabilities
  - *E.g.* Towelroot[1]
- Framework vulnerabilities
  - *E.g.* Master Key exploit



[1] Towelroot uses CVE-2014-3153 only, by George Hotz

Runtime hooking

- Available for Dalvik VM
  - DDI toolkit (Dynamic Dalvik Instrumentation) [1]
  - Xposed framework
- No hooks yet for ART

Patching the kernel

- Kernel module
  - Hooking with Kprobes
  - Kernel sources are needed
- Kpatch / Kgraft / Ksplice
  - Easy patch creation with unified diff
  - Kernel sources are needed

} Needed for each vuln and device

- Dynamic patching: *expatting*
  - Universal, cross-device solution
  - Using exploit or other kernel memory access technique
  - Slightly unorthodox

Dynamic patching

How to modify kernel memory and hook/patch vulnerable functions?

1. Find the kernel symbols
2. Get read/write access to kernel
3. Conduct patches

① Finding the kernel symbols

```
c0008000 T stext
c0008000 T _text
c000804c t __create_page_tables
c0008100 t __turn_mmu_on_loc
c000810c T secondary_startup
c0008148 t __secondary_switched
c0008154 t __secondary_data
c0008160 t __enable_mmu
```

- Read `/proc/kallsyms` or `/proc/ksyms`
  - `kptr_restrict` nullifies kernel pointers `%pK` in user space
- Scanning the memory for the correct addresses
  - Using `/dev/mem`, `/dev/kmem`, or `/proc/kcore`
  - Using exploit to read kernel memory
  - *E.g.* locate `%pK %c %s` and replace with `%p %c %s`

```
c014eca4 t lookup_mnt
c014ed14 t lock_mount
c014edcc t mnt_set_mountpoint
c014ee68 t attach_mnt
c014eee4 t attach_recursive_mnt
c014f01c t graft_tree
c014f09c t do_add_mount
c014f15c t do_move_mount
c014f340 t release_mounts
```

② Get read/write access to kernel

- Using exploit or `/dev/(k)mem`
- `mmap`: map devices or files into memory
- Backdoor original `mmap` system call
- Allows to r/w arbitrary kernel memory from user space

3 Conduct patches

Use the `mmap` backdoor to:

- Hook vulnerable kernel functions in-memory
- Patch Dalvik/ART framework functions as root

Evaluation

- Kernel patches become device independent
  - Still need to make the patch work for different architectures...
  - Quasi all Android devices are ARM
- Tricky: an error can cause kernel panic
  - Needs some fault tolerance
- Expat lives only in memory, non-permanent
  - Gone after reboot

Conclusion

- Patches can be made in a universal way
  - For both the kernel and the runtime
- Basis for an MDM setup to provide patches

Proof of concept

DEMO!

- Expat MDM, consists of agent and server module
- Exploiting and patching a kernel vulnerability

Many thanks to Deloitte!

References

PatchDroid: Scalable Third-Party Security Patches for Android Devices.

Collin Mulliner, Jon Oberheide, William Robertson, and Engin Kirda.

In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 259–268. ACM, 2013.

Security Evaluation of Mobile Device Management Solutions.

Joost Kremers.

Master's thesis, Radboud Universiteit Nijmegen, 2014.

Appendix: Boot hooking

	Pros	Cons
init script	+ cross-platform	– dm-verity – init.rc overwritten on boot
app_process binary	+ always in the same place	– dm-verity – architecture specific
broadcast receiver	+ cleanest	– allows race condition

Appendix: Exploiting

- *E.g.* PTMX device[2] as stepping stone: `ptmx_fops->fsync`
  Open `/dev/ptmx` and call `fsync`
- Transfer kernel execution to payload in user space:
  - Use commit_creds to run as fully privileged root user

[2] Doesn't reside in read-only kernel memory