Toward mobile Linux devices with auditable security

Copyright © 2015 pabr@pabr.org
All rights reserved.

This web page documents my attempts to assemble Linux smartphones and tablets with as few non-mainstream components, binary blobs and potential backdoors as possible. Currently the best candidate is a Lenovo Yoga Tablet 2 851 running Fedora 22.

READ THE HYPERTEXT VERSION HERE:
http://www.pabr.org/compsec/auditabletablet/auditabletablet.en.html

Revision History		
1.1	2015-10-08	French translation.
1.0	2015-09-23	Initial release.

Table of Contents

1. Motivations

Our mobile devices are too vulnerable to be used safely as Internet-facing devices:

- On the one hand, PC-compatible laptops now implement the same features as desktop computers and servers, including virtualization ("ring -1"), System Management Mode ("ring -2") and Active Management Technology ("ring -3"), all of which tend to turn operating systems into second-class citizens with limited control over the hardware. It is hard to make claims about security when you can't tell whether your OS is really running on bare metal rather than on a virtual machine set up by the BIOS. Or when your machine contains an always-on tamper-resistant co-processor that can monitor and control most peripherals even when the CPU is powered off.

- On the other hand, smartphones and tablets are still running locked-down operating systems on poorly-documented hardware platforms. As far as security is concerned, they become useless after a couple years when manufacturers stop shipping patches for major security holes.

In this project I aim to assemble a mobile device that is amenable to security assurance analysis. In practice this means I favour:

- **Open-source software .** Whether open-source software is intrinsically safer that closed-source software is debatable, as high-profile cases have shown in recent years (Heartbleed, Shellshock). However, there is little doubt that source code is easier to audit than binaries.

- **Mainline software .** Popular software projects are more actively maintained than obscure forks. On the one hand, this may lead to feature creep and an unstable code base. On the other hand, sustainable projects tend to have robust engineering practices, otherwise they would have collapsed. For example, the mainline Linux kernel is commonly used as a case study by academic research projects; hence, its source code is routinely scanned by a variety of static analyzers and other automated test tools. Branches maintained by OEMs and chip makers do not receive as much attention (see e.g. CVE-2012-6422 [http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6422]).

Note that several projects already claim to install Linux on various Android devices:

- ***KBOX*** **[http://kevinboone.net/android_nonroot.html] .** Provides a UNIX command-line experience. Essentially this is **busybox** and a few other tools in a terminal emulator, packaged as an Android app.

- ***Linux Deploy*** **[https://github.com/meefik/linuxdeploy],** ***LinuxonAndroid*** **[http://sourceforge.net/projects/linuxonandroid/] .** Run various distributions with a VNC server in a **chroot** sandbox, on top of Android.

- ***Ubuntu Touch*** **[https://wiki.ubuntu.com/Touch] .** Runs Ubuntu and a nice multitouch GUI alongside a stripped-down Android OS in a LXC container. Still depends on the Android kernel and associated closed-source drivers.

2. Status

	Yoga Tablet 2 851	**Yoga Tablet 2 830**	**Nexus 7 2013**
Release date	2014-10	2014-10	2013-07
Serial console	Maybe	Maybe	OK
Boot custom kernels	OK		OK
Boot mainline kernel	OK		OK
Framebuffer graphics	OK		
Accelerated graphics	Unstable		
Brightness control			
LED control			

	Yoga Tablet 2 851	Yoga Tablet 2 830	Nexus 7 2013
Orientation sensor			
Touchscreen pointer	OK		
Multitouch			
WLAN	OK		
Bluetooth	OK		
Battery charging	OK		
Battery monitoring	OK		
Suspend/resume			
Shutdown/reboot	OK		
Buttons	OK		
Sound			
Front camera			
Rear camera			
MMC		NA	
USB OTG	OK		

3. Lenovo Yoga Tablet 2 830/851

3.1. Overview

Pros:

- The SoC and GPU are from Intel. The WLAN/Bluetooth chip is from Broadcom.
- The platform (Bay Trail Atom Z3745) does not implement AMT.
- There is an Android version (830) and a Windows version (851). The Android version comes with Linux kernel source code. The Windows version has an unlockable UEFI BIOS.
- There are hints that a serial console can be accessed through the headset socket. The source code suggests that it is enabled by adding `switch_to_uart=1` to the kernel command line. See `osc_4_blade2_L_result/linux/kernel/drivers/misc/uart_audiojack_sw.c`
- There is a microSD card slot.
- Battery capacity is large, compared to other tablets. This will help in case mainline Linux fails to activate all the power-saving features.
- The partitioning scheme is straightforward:

```
mmcblk0p1      100 MiB   EFI
mmcblk0p2      128 MiB   Microsoft reserved (empty)
mmcblk0p3    23351 MiB   Microsoft basic data
mmcblk0p4     6239 MiB   Windows recovery environment
mmcblk0boot1     4 MiB   Empty
mmcblk0boot0     4 MiB   Empty
mmcblk0rpmb      4 MiB   Replay Protected Memory Block (content unknown)
```

- In addition to the eMMC storage, there is a 8 MiB SPI flash chip which could be physically replaced or reprogrammed if necessary.

Cons:

- The bootloader of the Android version (830) is locked. **fastboot boot** returns *"boot command stubbed on this platform"*.
- The kernel source code release is only paying lip service to the GPL. No version control information is included; not even file modification timestamps.
- The Windows version (851) is more expensive than the Android version: 249 EUR instead of 149 EUR. The difference buys 16 additional GiB of flash, a Windows button on the right side, an activity LED on the front panel, a black anodized finish, and a Windows license.
- As a full-featured x86 device with a PC-like BIOS, the 851 is a more complex platform than an ARM tablet. It is hard to really know what is going on under the hood, e.g. in relation to virtualization and SMM.
- The manufacturer has been caught installing dangerous software on other devices. It is believed that this only affects Windows.

Related work:

- *Fedlet: a Fedora Remix for Bay Trail tablets.* [https://www.happyassassin.net/fedlet-a-fedora-remix-for-bay-trail-tablets/] Release 20150810 boots to GNOME with `nomodeset`. Hardcoded WiFi MAC address, no Bluetooth, bad clocksource.
- *Ubuntu (or other Linux) on the Asus Transformer Book T100.* [http://www.jfwhome.com/2014/03/07/perfect-ubuntu-or-other-linux-on-the-asus-transformer-book-t100/]

3.2. Installing Fedora 22 on the Lenovo Yoga Tablet 2 851

My 851F came with BIOS version 02WT18WW (2015-07-24). There are reports that earlier versions will not allow the user to disable Secure Boot.

This is what I have done so far:

3.2.1. Preparation

- Copy the content of `Fedora-Live-Workstation-x86_64-22-3.iso` to the root of a FAT-formatted USB disk. We can't flash the read-only ISO9660 image because we will need to make a few changes.
- Copy `bootia32.efi` [https://github.com/jfwells/linux-asus-t100ta/raw/master/boot] to `EFI/BOOT/`. This is required because the 32-bit UEFI BIOS cannot boot the Fedora x86_64 EFI loader. It cannot boot the legacy ISOLINUX loader from Fedora i386 either.
- Copy `EFI/BOOT/grub.cfg` to `boot/grub/grub.cfg`. Otherwise `bootia32` cannot find it and drops to a GRUB prompt.
- Edit `boot/grub/grub.cfg`:

```
linux /isolinux/vmlinuz0 root=live:LABEL=FEDLIVE ro rd.live.image nomodeset
initrd /isolinux/initrd0.img
```

- Rename the USB disk partition so that GRUB can find it:

```
# fatlabel /dev/sdX1 FEDLIVE
```

3.2.2. Installation

- Connect the USB disk and a USB keyboard to the tablet through a USB hub and a USB OTG cable. There are reports that a powered hub is required.

- Power off, hold VolumeUp, press Power. This enters the "Novo Menu".
- Enter "BIOS Setup" and disable Secure Boot.
- Reboot into "Novo Menu" again.
- Enter "Boot Menu", select "EFI USB Device".
- Run the Fedora installer. Mount `/dev/mmcblk0p1` on `/boot/efi` and `/dev/mmcblk0p3` on `/`. Preserve `/dev/mmcblk0p4` (Windows restoration tool) just in case.
- Patch GRUB again:

```
# cp /run/initramfs/live/EFI/BOOT/bootia32.efi /boot/efi/EFI/Boot
# mkdir -p /boot/efi/boot/grub
# cp /boot/efi/EFI/fedora/grub.cfg /boot/efi/boot/grub/
```

  Edit `/boot/efi/boot/grub/grub.cfg`:

```
linux /boot/vmlinuz-4.0.4-301.fc22.x86_64 root=UUID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxx
initrd /boot/initramfs-4.0.4-301.fc22.x86_64.img
```

  Without `clocksource=tsc` the kernel switches to `refined-jiffies` and the system is abnormally slow. HPET is blacklisted on Bay Trail [http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/arch/x86/kernel/early-quirks.c?id=62187910b0fc7a75cfec9c30fda58ce2f39d689b].

- Reboot from the internal disk.

3.2.3. Customizations and workarounds

- In Gnome settings -> Power, disable automatic suspend. Suspend/resume is not reliable yet.
- Retrieve the WLAN firmware configuration data, including the device-specific MAC address, from EFI.

```
# cp /sys/firmware/efi/efivars/nvram-XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX /lib/firmwar
```

  There are two `nvram` files in that directory. The right one contains about 3 KB of settings in text format.

- Create `/etc/modprobe.d/brcmfmac.conf`. This avoids "*brcmf_escan_timeout: timer expired*" which is apparently caused by sdio entering power-saving mode.

```
install brcmfmac echo on > /sys/bus/platform/drivers/sdhci-acpi/INT33BB:00/power/contr
```

- Apparently the WiFi chip is configured to save power very aggressively. Outgoing connections work, but unexpected incoming packets buffer up somewhere until the `brcmfmac` driver wakes up. This makes it impossible to **ssh** into the tablet. I found three workarounds:
  - Make sure there is enough outgoing traffic to keep the driver busy:

```
yogatablet# ping wlanap &
```

  - Or poke the tablet with broadcast packets. For some reason they always reach the driver without delay.

```
wlanap# arping -I ath0 -b yogatablet
```

  - Or flood the tablet with enough incoming packets to prevent the chip from entering power saving mode. The maximum interval is about 200 ms, so pinging every 90 ms allows some packet loss. Note that one of the other techniques above is still needed to get the process started.

```
wlanap# ping -i 0.090 yogatablet
```

- `/sys/bus/acpi/devices/80860F0A:00/BCM2E84:00/path = \_SB_.URT1.BTH0` tells us that the Bluetooth portion of the BCM43241 is attached to the first port of a serial interface with ACPI ID 80860F0A.

  So, enable `CONFIG_SERIAL_8250_DW=m` to build `8250_dw.ko` (not included in Fedora kernels).

  Start **hciattach** e.g. with this `/etc/systemd/system/hciattach.service`:

```
[Unit]
Description=Broadcom BCM43241 in Yoga Tablet 2
After=syslog.target
Before=bluetooth.service

[Service]
Type=simple
ExecStartPre=/usr/bin/bash -c '/usr/bin/echo on > /sys/class/tty/ttyS1/power/control'
# bcm43xx_init is required after power up, but will fail after warm reboot.
# Don't know where to find the bdaddr. Use the WLAN MAC instead.
ExecStart=/usr/bin/bash -c 'MAC=$(sed -e "s/macaddr=\(..:..:..:..:..:..\)/\1/;t;d" /sy

[Install]
WantedBy=multi-user.target
```

```
yogatablet# systemctl enable hciattach.service
yogatablet# systemctl start hciattach.service
```

- At this point **Xorg** is running on top of the EFI framebuffer, which does not support run-time rotation with **xrandr**. To permanently use the screen in landscape mode, create `/etc/X11/xorg.conf.d/90-RotateCW.conf`:

```
Section "Device"
    Identifier  "Card0"
    Driver      "fbdev"
    Option      "Rotate" "CW"
EndSection

Section "InputClass"
    Identifier  "RotateCW"
    MatchIsTouchscreen "True"
    Option      "TransformationMatrix" "0 1 0 -1 0 1 0 0 1"
EndSection
```

  Unfortunately this causes **gnome** to complain about missing RANDR support and segfault. I was not planning to use it anyway. Legacy window managers work fine in landscape mode.

3.2.4. Known problems

- The GPU can be enabled by replacing `nomodeset` with `i915.modeset=1` in the kernel command line. Most of the time it will hang for 60 seconds in `intel_crtc_wait_for_pending_flips()` on startup. Afterward **xrandr -rotate** works, and **glxgears** runs full-screen at 60 fps. Unfortunately when the GPU is enabled the machine usually hangs after a few minutes.

- Suspend/resume does not work.

4. Fedora on the ASUS / Google Nexus 7 2013 (flo)

4.1. Overview

Pros:

- Nexus devices are considered as developer-friendly.
- The bootloader is unlockable. It is easy to compile and boot custom kernels with the [AOSP] tools.
- The bootloader is re-lockable. This provides some security against physical attacks.
- Source code for the open components is maintained as part of AOSP.
- A serial console can be accessed through the headset socket.
- There are hints of JTAG signals on the circuit board, hidden under a barcode label.
- The device can be powered inductively while the USB port is used in OTG mode for development.

Cons:

- The CPU, GPU and WLAN chips are from Qualcomm.
- The AOSP release contains lots of closed-source components; not just firmware blobs, but also ARM binaries. It looks like low-level graphics drivers are running in userspace rather than in the kernel, possibly to shield them from the GPL.
- The Nexus 7 (2013) is still not supported by mainstream Linux more than two years after its release.

Related work:

- *Running mainline kernels on a 2013 N7(flo)* [https://plus.google.com/111524780435806926688/posts/S4ajVewveSR]
- *Kernel development – vanilla all the way* [http://delta.zauner.one/168-nexus-7-kernel/]

4.2. Procedure

This is what I have done so far:

- Build a serial console cable. See [CONSOLEJACK]. The bootloader detects the cable and switches the audio socket to serial mode.
- Compile mainline Linux 4.2.0 with gcc 4.6. Do NOT use the 4.8 and 4.9 toolchains from the current AOSP repository.

```
$ export ARCH="arm"
$ export CROSS_COMPILE="arm-eabi-"
$ make flo_defconfig
```

- Build `fixup.bin` and `qcom-apq8064-nexus7.dtb` as explained here [https://plus.google.com/111524780435806926688/posts/S4ajVewveSR]:

```
arm-eabi-gcc -c fixup.S -o fixup.o
```

```
arm-eabi-objcopy -O binary fixup.o fixup.bin
cat fixup.bin linux-4.2.0/arch/arm/boot/zImage linux-4.2.0/arch/arm/boot/dts/qcom-apq8
```

- Power off, hold VolumeDown, press Power. This enters "FASTBOOT MODE".
- Boot to a shell using any recovery image that contains **busybox**:

```
fastboot boot -n 2048 --base 0x80200000 --ramdisk_offset 0x02000000  \
   -c "console=ttyHSL0,115200,n8 user_debug=31 msm_rtb.filter=0x3F ehci-hcd.park=3 vma
   fixup-zImage-dtb recovery/initrd.img
```

At this point I am testing the mainline `msm` driver (`drivers/gpu/drm/msm/`). This involves writing a DTS file and enabling:

```
CONFIG_DRM_MSM=y
CONFIG_CMA=y
CONFIG_DMA_CMA=y
```

5. Final thoughts

Smartphones and tablets used to be proprietary hardware platforms with walled-garden software ecosystems. I started this project under the assumption that Android had shattered that status quo seven years ago, and that it should now be easy to install mainstream Linux distribution at least on Google's developer-friendly devices. Sadly, it turned out that the easiest route to running mainline Linux on a mobile device is to choose a tablet that is actually an Intel x86 PC. Maybe all the competent arm-linux developers have landed dream jobs with phone manufacturers and there are few people left to do mainline work.

On x86, the plug-and-play mechanisms built into UEFI, ACPI and PCI help a lot. On ARM, plug-and-play was only introduced around 2009. Eventually the Open Firmware/device tree infrastructure will allow a universal kernel to boot on a multitude of ARM platforms. The Nexus 7 is one of many Android devices that still use compiled-in `platform_data` structures. Most recent Nexus devices now ship with a `dtb` blob appended to the kernel; see e.g. `tegra124-flounder.dts` [https://android.googlesource.com/kernel/tegra/+/android-tegra-flounder-3.10-lollipop-release/arch/arm/boot/dts/tegra124-flounder.dts] and `tegra124-flounder-generic.dtsi` [https://android.googlesource.com/kernel/tegra/+/android-tegra-flounder-3.10-lollipop-release/arch/arm/boot/dts/tegra124-flounder-generic.dtsi] (Nexus 9). I do not mind writing DTS files myself; unfortunately the dominant vendor(s) of mobile ARM SoCs do(es) not publish technical documentation, so this requires a tedious process of scavenging I/O addresses, interrupt numbers and GPIO assignments from various places.

Another difficulty in this project, of course, is that there are not so many Android devices with unlockable bootloaders to choose from. It is surprising that manufacturers are still getting away with such practices, while it is now commonly agreed that PC hardware should not be artificially tied to a specific operating system.

Bibliography

[CONSOLEJACK] *A better audio jack console cable for Google Nexus devices* . http://www.pabr.org/consolejack/consolejack.en.html .

[AOSP] *Android Open-Source Project*. https://source.android.com/.