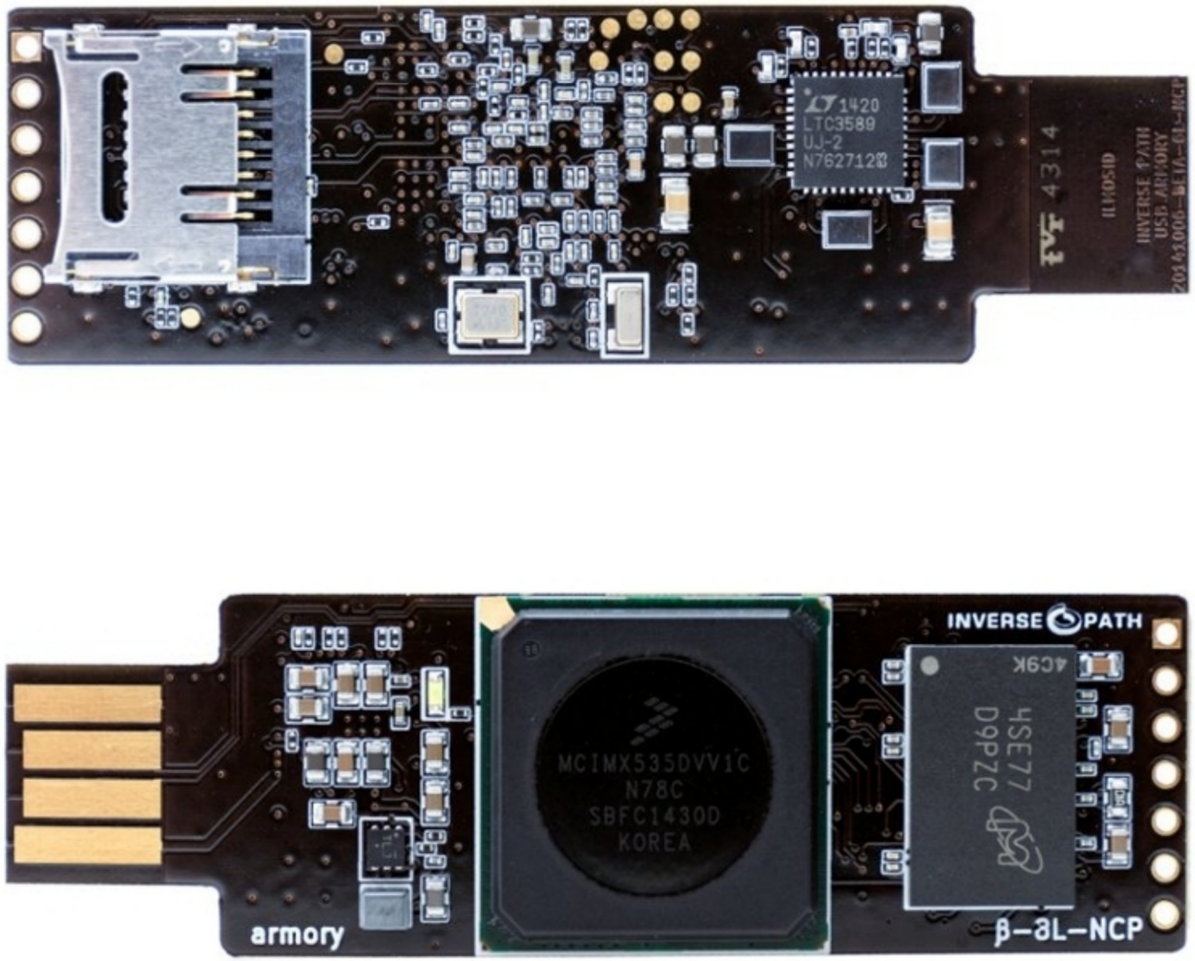
Filippo Cremonese

USB Armory

Much more than just linux on a stick



INVERSE PATH

inversepath.com/usbarmory

What is it?

- An ARM system in USB pen drive form
- Designed for security applications
- Open hardware (& software)
  - Schematics available on github
  - SoC datasheet and manuals available for free

Hardware

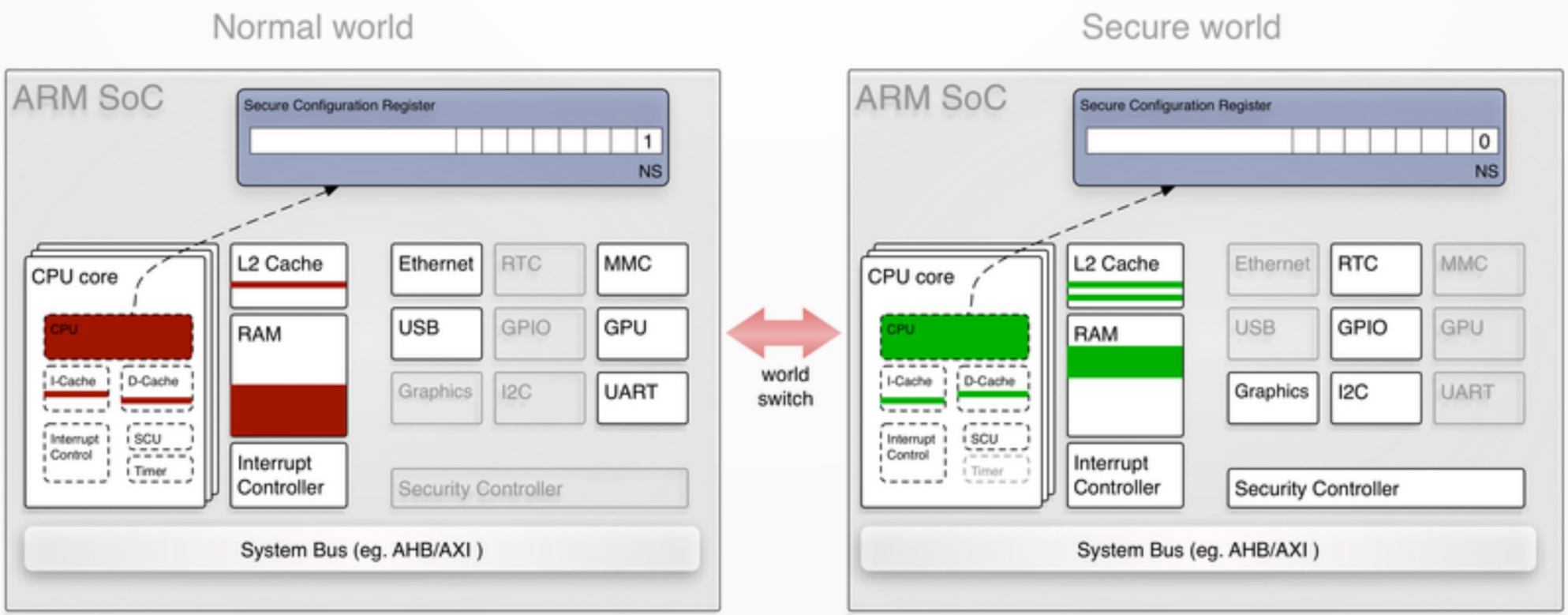
- ARM Cortex A8 Processor (Freescale iMX.53)
  - 800MHz clock
  - 512MB RAM
- MicroSD card reader
- LED
- USB A connector
  - Controller suppots both host and device mode
- 3 GPIO + 2 UART/GPIO pins
- JTAG

HW security features

- ARM TrustZone
- High assurance boot (HABv4)
- Security controller (SCCv2)
  - Secure RAM
  - AES with unique embedded secret key
- Cryptographic coprocessor (SAHARAv4 Lite)
  - Various ciphers and hashes
  - True random number generator

ARM TrustZone

- Separates resources between two worlds
- Implementation details are vendor dependant



iMX53 TrustZone

Both worlds can be assigned

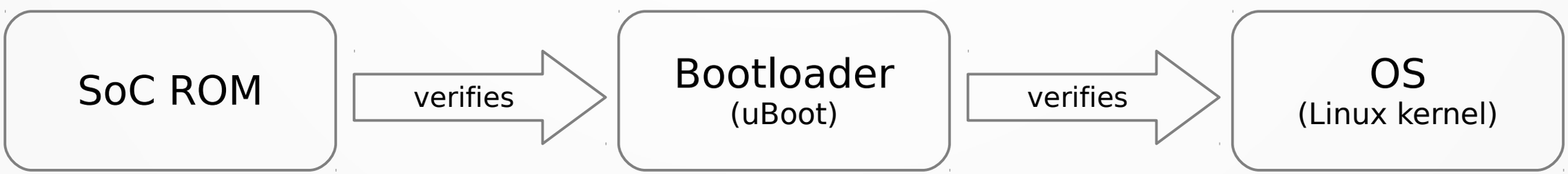
- Ranges of physical RAM
  - Not transparent to the normal world
  - There are problems with GPU/IPU separation
    - They share the same DMA channel ID
    - Not a concern in the USB armory scenario
- Groups of devices
  - Splitting devices into groups is up to the device manufacturer
- Interrupts

Access violations are synchronously reported to the secure world

It is possible to run a complex preemptive kernel in the secure world

High Assurance Boot

- Allows cryptographic verification of software
- Chain of trust starts from SoC firmware



High Assurance Boot

- Similar in principle to Secure Boot
- All the signing keys are user managed
- Once activated it cannot be reset
    - If you lose the keys you gain a USB sized brick

HAB Setup process

- A full blown PKI is created
- Public keys hashes are fused into the SoC
  - 4 key slots
  - Up to 3 keys can be revoked
  - Done from uBoot serial console
- Key slots are locked and HAB enabled
- uBoot is recompiled and signed
  - A public key is embedded for OS verification

Security controller (SCCv2)

- Has its own secure RAM
- Implements AES-256 (CBC or ECB)
- Uses a random, nonvolatile, static secret key
  - Fused at manufacturing time
  - Can’t be extracted
  - Available only if HAB is enabled
    - Otherwise the NIST standard test key is used

SCCv2 applications

- Ideal for tying crypto operations to the device
- Example: Derive encryption keys from the SCC
  - The disk image by itself becomes useless
- Exposed as a device on Linux

SCCv2 example pseudocode

```
fd = open("/dev/scc2_aes", O_RDWR)
ioctl(fd, SET_IV, iv)

ioctl(fd, SET_MODE, ENCRYPT_CBC)
write(fd, plaintext, 4096)
read(fd, ciphertext, 4096)

ioctl(fd, SET_MODE, DECRYPT_CBC)
write(fd, ciphertext, 4096)
read(fd, plaintext, 4096)
```

Operating systems

Prebuilt images are available for

- Ubuntu
- Debian
- Arch
- Genode OS + Linux
  - Hypervisor running in TrustZone
  - Runs linux in the normal world
- And other distros

Genode OS

- Framework for writing microkernels
- Uses hierarchies to manage and restrict resources

- UART, SD card, LED assigned to secure world
  - Used through a para-virtualized driver by linux
  - Linux and VMM can only see a specific partition
  - UART output gets prefixed to distinguish worlds
  - LED indicates execution context

Personal applications

- Smart mass storage
  - transparent encryption
  - tamper detection
  - virus scanning
- VPN/Tor (semi)transparent router
- 2FA token
- Password manager

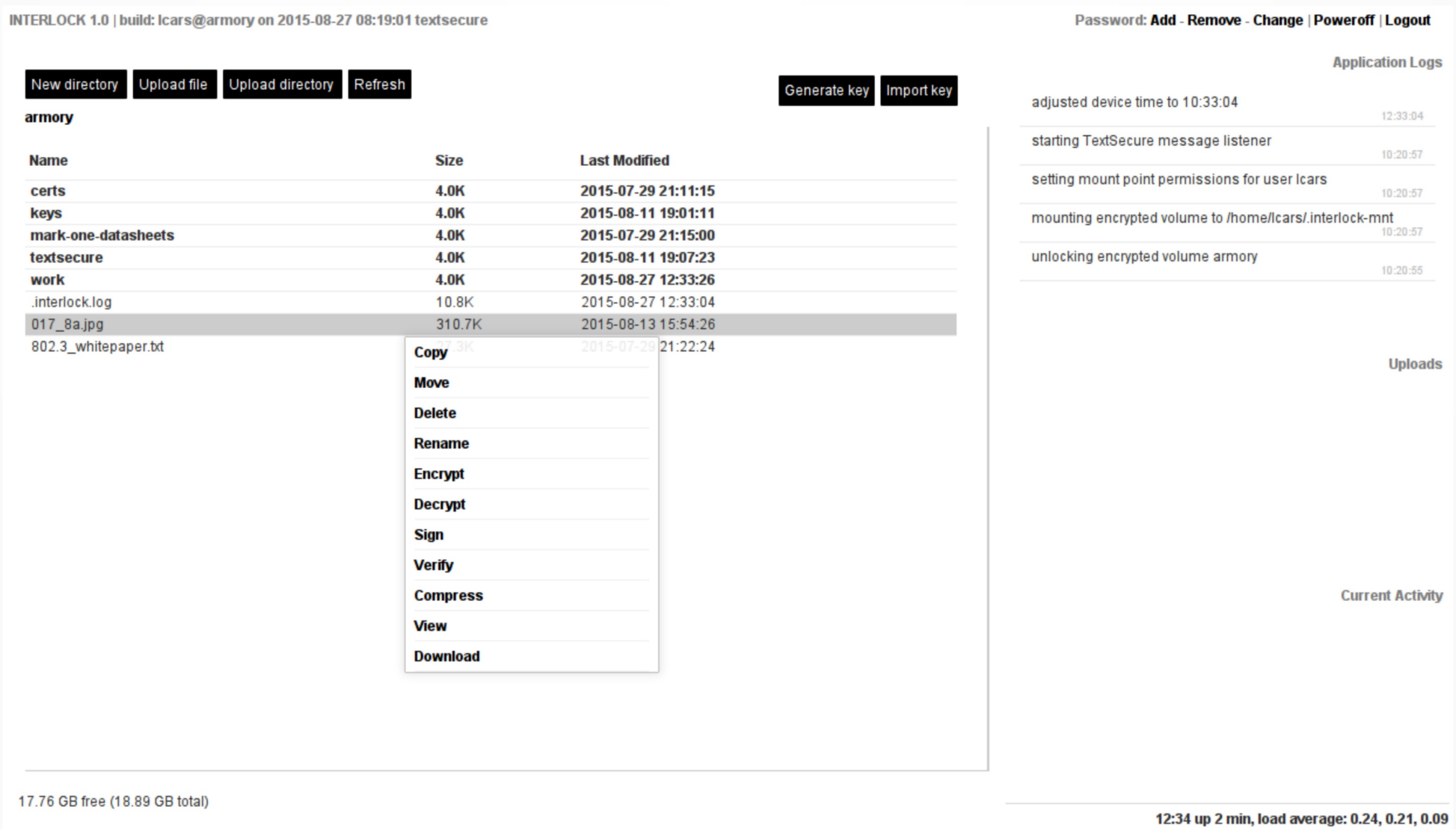
Pentesting applications

- Pwny keyboard (HID emulation)
- MITM device (CDC/NDIS Ethernet)
- Low level USB security testing/fuzzing
  - If you can deal with the Linux USB stack

Existing applications

- Armory-pass
  - Proof of concept password manager for chrome
  - Releases just the password for the current origin
- Interlock
- FAT Abuser

Interlock


INTERLOCK 1.0 | build: lcars@armory on 2015-08-27 08:19:01 textsecure
Password: Add - Remove - Change | Poweroff | Logout
New directory
Upload file
Upload directory
Refresh
Generate key
Import key
armory
Name
Size
Last Modified
certs 4.0K 2015-07-29 21:11:15
keys 4.0K 2015-08-11 19:01:11
mark-one-datasheets 4.0K 2015-07-29 21:15:00
textsecure 4.0K 2015-08-11 19:07:23
work 4.0K 2015-08-27 12:33:26
.interlock.log 10.8K 2015-08-27 12:33:04
017_8a.jpg 310.7K 2015-08-13 15:54:26
802.3_whitepaper.txt 21:22:24
Copy
Move
Delete
Rename
Encrypt
Decrypt
Sign
Verify
Compress
View
Download
Application Logs
adjusted device time to 10:33:04 12:33:04
starting TextSecure message listener 10:20:57
setting mount point permissions for user lcars 10:20:57
mounting encrypted volume to /home/lcars/.interlock-mnt 10:20:57
unlocking encrypted volume armory 10:20:55
Uploads
Current Activity
17.76 GB free (18.89 GB total)
12:34 up 2 min, load average: 0.24, 0.21, 0.09

FAT Abuser

Sources/Must read links

Here are some interesting reads and sources from which I ~~stole content~~ got inspiration from

- https://genode.org/documentation/articles/trustzone GenodeOS exploration of TrustZone
- https://genode.org/documentation/articles/usb_armory GenodeOS on the USB armory
- https://github.com/inversepath/usbarmory/wiki USB Armory wiki & code
- Andrea Barisani CCC and Polimi talks
- https://dev.inversepath.com/download/usbarmory/forging_the_usb_armory.pdf Andrea Barisani slides
- http://www.nxp.com/assets/documents/data/en/application-notes/AN4581.pdf High Assurance Boot
- http://www.nxp.com/assets/documents/data/en/reference-manuals/iMX53RM.pdf iMX53 manual
- http://www.nxp.com/assets/documents/data/en/data-sheets/IMX53IEC.pdf iMX53 datasheet
- https://github.com/mweissbacher/armory-pass
-

That's all!

Questions?