Forging the USB armory

Andrea Barisani

<andrea@inversepath.com>
2007: Unusual Car Navigation Tricks
Injecting RDS-TMC Traffic Information Signals

2009: Sniff Keystrokes With Lasers/Voltmeters
Side Channel Attacks Using Optical Sampling Of
Mechanical Energy And Power Line Leakage

2011: Chip & PIN is definitely broken
Credit card skimming and PIN harvesting in an EMV world

2013: Fully arbitrary 802.3 packet injection
Maximizing the Ethernet attack surface
Designed for personal security applications

- mass storage device with advanced features such as automatic encryption, virus scanning, host authentication and data self-destruct
- OpenSSH client and agent for untrusted hosts (kiosk)
- router for end-to-end VPN tunneling, Tor
- password manager with integrated web server
- electronic wallet (e.g. pocket Bitcoin wallet)
- authentication token
- portable penetration testing platform
- low level USB security testing
enhanced mass storage

Host -> mass storage gadget -> clamav -> gnupg -> microSD

Host -> USB armory

mass storage gadget -> please scan -> passed

secret.txt -> secret.txt.jpg

Copyright 2015 Inverse Path S.r.l.
enhanced mass storage
enhanced mass storage

- Chuck
  - show me the secret files
  - of course
  - nothing to see here ^_^

- Alice

- mass storage gadget

- microSD

- open secret folder gamma

- failsafe word detected!

- wipe command

- data self-destruct

- USB armory
SSH proxy
password manager

Host

webapp

Bank

TCP/IP over CDC Ethernet

password for my banking site?

what's your PIN?

1234

seriously?

yep

ok, password is ************

cut & paste ************ to banking site

*trivial example, better options planned
authentication token

Host <-> USB armory <-> TCP/IP over CDC Ethernet

Host <-> webapp <-> Whatever

gimme my token!

aye aye sir!

your token is 1453632

145632
USB device authenticates host

Host authentication

Host

SSH client

USB armory

TCP/IP over CDC Ethernet

get fingerprint

Hello!

matches!

Host

SSH client

stored fingerprint

Host authentication

Host

SSH client

stored fingerprint

USB armory

TCP/IP over CDC Ethernet

get ssh fingerprint

you are not my host ;(^

Host

SSH client

stored fingerprint

Copyright 2015 Inverse Path S.r.l.
Design goals

- Compact USB powered device
- Fast CPU and generous RAM
- Secure boot
- Standard connectivity over USB
- Familiar developing/execution environment
- Open design
Selecting the System on Chip (SoC)

Freescale i.MX53

- ARM® Cortex™-A8 800-1200 Mhz
- almost all datasheets/manuals are public (no NDA required)
- Freescale datasheets are “ok” (far better than other vendors)
- ARM® TrustZone®, secure boot + storage + RAM
- detailed power consumption guide available
- excellent native support (Android, Debian, Ubuntu, FreeBSD)
- good stock and production support guarantee
Domain Separation

Nonsecure

- user mode applications
- privileged kernel

Secure

- user mode applications
- privileged kernel
- secure device

Monitor mode

- device driver
ARM® TrustZone®

http://genode.org/documentation/articles/trustzone
ARM® TrustZone®

http://genode.org/documentation/articles/trustzone
Development time-line

2014/03: schematics development begins (Freescale chosen)
2014/04: PCB layout for breakout/prototyping board
2014/08: alpha board order
2014/09: USB armory alpha board delivery & evaluation
2014/10: project announcement
2014/10: order for 7 optimized beta revisions
2014/11: beta boards delivery & evaluation
2014/11: design finalization, Mk I production candidate order
2014/12: Mk I delivery
2015/01: first batch production (1k)
2015/03: shipping begins!
USB armory - Open source flash-drive-sized computer

- Freescale i.MX53 ARM® Cortex™-A8 800Mhz, 512MB DDR3 RAM
- USB host powered (<500 mA) device with compact form factor (65 x 19 x 6 mm)
- ARM® TrustZone®, secure boot + storage + RAM
- microSD card slot
- 5-pin breakout header with GPIOs and UART
- customizable LED, including secure mode detection
- excellent native support (Debian, Ubuntu, Arch Linux ARM)
- USB device emulation (CDC Ethernet, mass storage, HID, etc.)
- Open Hardware & Software
device mode
α
8L-NOUSBH, 8L, 8L-DDR-LDO, 8L-DDR-NCP
6L, 6L-DDR-LDO, 6L-DDR-NCP

βs

Mk I

Forging the USB armory
lessons learned #1
tiny inductors are fragile
lessons learned #2 (the five-second rule)
gold plating traces cause under-voltage on hot swap
Compiling and running Genode OS (>= 15.02):

```
git clone https://github.com/genodelabs/genode
cd genode

./tool/create_builddir hw_usb_armory
cd build/hw_usb_armory

# in etc/build.conf add "--include image/uboot" to RUN_OPT

make run/tz_vmm
cp var/run/tz_vmm/uImage $SD_CARD_MNT

uboot> ext2load mmc 0:1 0x70200000 /boot/uImage-genode; bootm 0x70200000
```

Requires minimally patch Normal world kernel compiled as follows:

```
make ARCH=arm zImage LOADADDR=0x80008000 modules
```
Secure Mode Monitor (LED example)

@ set GPIO4 to SECURE
    movw   r0, #0x33
    movt   r0, #0xff
    ldr    r1, =CSU_CSL
    add    r1, r1, #4   @ CSL1
    str    r0, [r1]

@ set IOMUXC to SECURE
    movw   r0, #0x33
    movt   r0, #0xff
    ldr    r1, =CSU_CSL
    add    r1, r1, #20  @ CSL5
    str    r0, [r1]

@ set OCRAM to SECURE
...

_secure_monitor:
    mov     r10, #0xcafe
    cmp     r0, r10
    beq     smc_handler
    beq     to_nonsecure
Secure Mode Monitor (LED example)

```assembly
smc_handler:

    ldr    r10, =IOMUX_LED
    mov    r0, #1
    movt   r0, #0
    str    r0, [r10]               @ set the pad to GPIO

    ldr    r10, =GPIO4_DIR
    movw   r0, #0xffffffff
    movt   r0, #0xffffffff
    str    r0, [r10]               @ set direction to output

    ldr    r10, =GPIO4_DR
    ldr    r0, [r10]
    mvn    r0, r0
    str    r0, [r10]               @ toggle LED output

    movs   pc, lr
```

Copyright 2015 Inverse Path S.r.l.
Secure Mode Monitor (LED example)

```c
static int beg_for_led_switch(void)
{
    printk("dear smc, kindly switch the LED\n");

    /* give control to the secure monitor */
    asm volatile ("movw r0, #0xc0f0e");
    asm volatile ("smc #0");

    return 0;
}
```

The LED is hardware restricted via TrustZone to Secure monitor control.

A trivial interface implementation between Nonsecure Linux and Secure monitor illustrates a simple request for LED switching.
Open source file encryption front-end developed, but not limited to, usage with the USB armory.

Provides a web accessible file manager to unlock/lock LUKS encrypted partition and perform additional symmetric/asymmetric encryption on stored files.

Take advantage of disposable passwords, “nuking” option.
Design Goals

Clear separation between presentation and server layer to ease auditability and integration.

Minimum amount of external dependencies and footprint.

Encrypted volumes: LUKS encrypted partitions
Asymmetric ciphers: OpenPGP
Symmetric ciphers: AES-256-OFB w/ PBKDF2 + HMAC
Security tokens: Time-based One-Time Password Algorithm (Google Authenticator)
Authentication credentials are directly tied to LUKS partition.
Files can be further encrypted on the USB armory...
...and later downloaded.
Logging out locks the encrypted partition.
Thank you!

Q & A

Andrea Barisani
<andrea@inversepath.com>