Secure updates for memory-constrained XIP system

Vitaly Wool, Konsulko Group
About me

- Has been with embedded Linux since 2003
- Worked for MontaVista
- Currently living in Sweden (Skåne)
- Staff Engineer at Konsulko Group
- Managing Director at Konsulko AB
About this presentation

- What’s OTA
- What’s XIP
- OTA and XIP
  - And memory constraints
- Conclusions
What’s OTA?
OTA / FOTA

- [Firmware] Over-The-Air update
  - No need to physically connect device being updated

- Widely used for mobile devices and routers
  - NB: infamous router updates

- Coming to automobiles, IoT devices etc.
  - Non-OTA update would require a service visit
    - E. g. driving to car service center
    - ...or a visiting technician
  - Some IoT devices may be far away or hard to access
FOSS OTA updaters

- OSTree (libostree)
  - Used by AGL, Fedora

- swupupdate
  - Partial OE integration

- RAUC
  - Good OE integration

- update_engine
  - Used by Android
OTA updater requirements

- Fail-safe
  - No “partial updates”

- Recoverable: rollback to a previous software state
  - Basically implies having 2 versions of software
  - Sometimes not possible due to size limitations

- Capable of updating all software / firmware
  - Bootloader, kernel, root file system, data

- Secure
  - Update package authenticity and integrity
OTA classification 1

OTA

Updater method

Single copy
With rescue partition

Boot → APP SW → SWU
Update process

Double copy
A/B update

Boot → APP SW A → APP SW B
Update process
OTA classification 2

OTA
  └── Updater execution
       ├── No reboot required
       │    └── Called from userspace
       │         └── userspace
       │                  └── Ordinary Application
       │                      └── Trustzone
       │                                   └── Trusted application
       │                                      └── Special kernel/ramdisk
       │                                           └── bootloader
       └── Reboot required
            └── Scheduled on reboot
Double-copy OTA

RAM constraints

Self-recovery

Security

Double copy OTA
What’s XIP?
XIP: execute in place

- Code executed directly from persistent storage
  - Typically NOR flash
  - QSPI

- XIP kernel
  - Option selected at compile time

- XIP userspace
  - Requires a special filesystem
    - Cramfs (legacy), AXFS
Kernel XIP

Traditional XIP design (userspace can be anywhere)
Kernel/Userspace XIP

More expensive design but we do save on RAM
XIP advantages

- Less RAM needed
  - Usually up to 10x smaller RAM footprint
  - Sometimes no RAM at all is needed

- Lower idle power consumption
  - May be crucial for IoT running on battery

- Shorter boot time
  - No copy on boot

- Faster execution
  - QSPI flash
XIP obstacles

- You can’t write to flash and execute from it at the same time
- However, you can write to flash using special tricks
  - Code copied/executed from RAM
  - No other code may be executed during that time
- XIP requires more space on flash storage
  - At least kernel code can not be compressed
- All addresses are defined at compile time
  - Which may be a security compromise
OTA and XIP
OTA and XIP: Same goals...

- Smaller footprint
- Faster boot
- Convenient and cost effective
- Easy maintenance
- Remote IoT
- Automotive

XIP

OTA
...sharper underwater rocks

- Fail-safety is crucial
  - Easier to brick device
  - Possible security breaches

- Memory-constrained system
  - Integral update image may not fit

- That calls for a double-copy mechanism

- We’ll show that existing double-copy are no good with XIP
RAM disk (initrd) OTA

- Single copy
- Will it work with XIP? **YES**
  - updater can occupy userspace / kernel data area
- Requires the whole update image to fit in memory
Bootloader OTA

- Basically the same as initrd, but updater is in the bootloader
  - Likely to consume less space

- Very “thick” bootloader
  - [part of] bootloader should run from RAM
  - Should be aware of system internals
  - Harder to debug
  - Less secure

- Will it work with XIP? **YES**
Userspace OTA

- Simple in non-XIP case
  - update inactive kernel/application partitions
  - Verify, mark as active and reboot

- Kernel A can not execute during Kernel B update
  - Interrupts and preemption must be disabled during update

- Userspace may be XIP too
  - Updater should be copied to RAM with all the libraries it would use
Trustzone OTA (ARM)

Secure monitor

- Linux kernel
- App 1
- Updater

Trusted OS

- Real updater

RAM
Conclusions

- XIP can add value to OTA solutions
  - But it adds complexity, too

- XIP puts certain requirements on updaters

- Existing FOSS updaters don’t play together well with XIP

- Secure updates with trusted application work well with XIP
  - But there are no known FOSS solution for that yet
Questions?

Vitaly.Wool@konsulko.com