Software update for IoT
the current state of play

Chris Simmonds

OpenIoT Summit 2016
License

These slides are available under a Creative Commons Attribution-ShareAlike 3.0 license. You can read the full text of the license here http://creativecommons.org/licenses/by-sa/3.0/legalcode
You are free to

• copy, distribute, display, and perform the work
• make derivative works
• make commercial use of the work

Under the following conditions

• Attribution: you must give the original author credit
• Share Alike: if you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one (i.e. include this page exactly as it is)
• For any reuse or distribution, you must make clear to others the license terms of this work
About Chris Simmonds

- Consultant and trainer
- Author of *Mastering Embedded Linux Programming*
- Working with embedded Linux since 1999
- Android since 2009
- Speaker at many conferences and workshops

"Looking after the Inner Penguin" blog at http://2net.co.uk/

https://uk.linkedin.com/in/chrisdsimmonds/

https://google.com/+chrissimmonds
Overview

- Software update 101
- Update clients
- OTA update
- OTA implementations
What could possibly go wrong?

- **Mirai**: a recent > 600 Gbps DDoS attack
- Very simple: looks for open Telnet ports and logs on using default, well-known, name and password
- Prime target: Dahua IP CCTV cameras

Details on PenTestPartners:

https://www.pentestpartners.com/blog/optimising-mirai-a-better-iot-ddos-botnet
Problems

Problem 1
- Embedded software is non-trivial (=> has bugs!)
- Devices are often connected to the Internet
  - Allowing intruders to exploit the bugs remotely

Problem 2
- We would like to deploy new features, improve performance, etc.

Conclusion
- We need a software update mechanism
Requirements for SW update

• Secure, to prevent the device from being hijacked
• Robust, so that an update does not render the device unusable
• Atomic, meaning that an update must be installed completely or not at all
• Fail-safe, so that there is a fall-back mode if all else fails
• Preserve persistent state
What to update?

- Frequency
- Ease of update
- Bootloader
- Kernel
- Root file system
- System applications
Update granularity

• File:
  • not an option: hard to achieve atomicity over a group of file updates

• Package:
  • \texttt{apt-get update} works fine for servers but not for devices

• Container:
  • neat idea, so long as you have containerised applications

• Image:
  • the most common option: fairy easy to implement and verify
Device update != server update

- Server
  - Secure environment, no power outage, no network outage
  - If update fails, human intervention is possible
- Device:
  - Intermittent power and network mean update quite likely to be interrupted
  - Failed update may be difficult (and expensive) to resolve
Options for image update

Symmetric A/B (Android after Nougat)

Asymmetric normal/recovery (Android before Nougat)
Statelessness

- Image update of a filesystem implies no state is stored in that filesystem
Update agent

- Update agent is the code on the device that manages the update
- Tasks
  - Receive update from local storage (e.g. USB) or from remote server
  - Apply the update
  - Toggle boot flag
swupdate

- Image-based update client
- License: GPLv2
- Code https://github.com/sbabic/swupdate
swupdate features

- Symmetric and asymmetric update
- Bootloader support: U-Boot
- Volume formats: MTD, UBI, MBR and UEFI partitions
- Yocto Project layer: meta-swupdate
- Remote/streaming using curl (http/https/ssh/ftp)
- integrated REST client connector to hawkBit
- Signed images
RAUC - Robust Auto-Update Controller

- Image-based update client
- License: LGPLv2.1
- Source Code: https://github.com/jluebbe/rauc
- Documentation: https://rauc.readthedocs.org/
RAUC features

- Symmetric and asymmetric update
- Bootloader support: grub, barebox
- Volume formats: MTD, UBI, MBR and UEFI partitions
- Build systems: Yocto Project (meta-pxt), PTXDist
- Remote/streaming using curl (http/https/ssh/ftp)
- Cryptographic verification using OpenSSL (signatures based on x.509 certificates)
OTA update

- Solutions so far are mostly suitable for
  - Local update (man with a USB thumb drive)
  - User initiated/attended remote update
- Local or attended remote update does not scale
- Hence, OTA (Over The Air) update
  - Updates pushed from central server
  - Update is automatic (or semi-automatic as with Android/iOS)
OTA update components

- Device software build system
- Firmware images
- Update server
- Sign with authentication key
- Device
  - Update agent
Complexities of OTA update

- Authentication (is this update legit?)
- Security (am I receiving what you are sending?)
- Roll-back (if update fails to boot, switch to previous version)
- Scale (roll out to large populations)
- Monitoring (keeping track of status of the population of devices)
Roll-back

- Boot limit count
  - Feature of bootloader (e.g U-Boot)
  - Increment count in bootloader
  - Reset after successful boot
  - If reboot with count > 0, bootloader knows boot failed and loads alternate rootfs

- Hardware watchdog
  - If hang in early boot, watchdog times out and resets CPU
  - Bootloader checks reset reason
  - If watchdog, loads alternate rootfs
Mender.io

• OTA update server and client
• Full system image update
• Licenses: Server and Client: Apache 2
• Code (client): https://github.com/mendersoftware/mender
• Documentation: https://docs.mender.io
Mender.io features

- Symmetric A/B image update client
- Bootloader support: U-Boot
- Volume formats: MBR and UEFI partitions
- Update commit and roll-back
- Build system: Yocto Project (meta-mender)
- Remote features: deployment server, build artifact management, device management console
Resin.io

• OTA update server and client
• Container (Docker) based updates
• Licenses: Client: Apache2; Server: proprietary
• Code (client): https://github.com/resin-os/meta-resin
• Documentation: https://docs.resin.io/introduction
resin.io features

• Symetric A/B rootfs for core OS ("Resinhup")
• Applications packaged into Docker containers
• Build integration: Yocto Project (meta-resin)
• Docker images can be preloaded into YP build
• Remote features: deployment server, integration with git
Brillo

- Brillo is cut-down Android for IoT
- License: Apache 2.0
- Android OTA update client
- Symmetric and asymmetric image update
- Licenses: Client: Apache2; Server: proprietary
- Code (client): https://android.googlesource.com
- Documentation: https://developers.google.com/brillo
Conclusion

• Software update is a hot topic
• Open source solutions described in this presentation:
  • Stand-alone update clients
    • swupdaed
    • RAUC
  • End-to-end solutions
    • mender.io
    • resin.io
• Questions?

Slides on Slide Share

http://www.slideshare.net/chrissimmonds/
software-update-for-iot-the-current-state-of-play