LinuxCon
Tokyo, Japan 2016
Secure IoT Gateway

Jim Gallagher
Senior Technical Marketing Lead, MontaVista Software
Setting the Stage

- This presentation will focus on developing Secure Gateways (Edge Computing & Connectivity) in the IoT Architecture
- Primarily discussion will be on Architecture, Security, and Maintainence features
Agenda

1. Architecture
2. Connectivity
3. Security
4. Maintainability
5. Summary and Q&A
Architecture: Modern vs Wild West
Architecture choices

Embedded processor considerations:

- Processor family
  - ARM
  - Intel x86
  - PowerPC and MIPS possible but not as popular

- Power consumption
  - ARM: low power, advanced PM features
  - Intel x86: limited PM options
Architecture choices (ctd)

Embedded processor considerations:

- Performance
  - ARM: Good core performance on lower Ghz
  - Intel x86: ”Add Ghz -> more perf”

- Optimizations
  - Security offload
  - Virtualization

- Deployment model
  - SOC model vs. ”generic compute”
  - Longevity?
Ecosystem

- Intel vs. ARM really
  - PPC and MIPS thin and fading ecosystem
  - Ubuntu, Fedora, Debian, OpenSUSE, MontaVista, WindRiver, and Enea all have/will have x86 and ARM support for mainstream distros
  - LINARO (ARM lead)

- Yocto project (Intel lead)
  - Consolidate embedded development on OE/bitbake
Connectivity
Sensors, Sensors everywhere!

- Simple sensor data drives the IoT engine
  - Fitness trackers, heart monitors, oil and pressure temperature gauges, & packet latency in SDN

- What connects them
  - Wireless: Bluetooth, Wi-fi, Cellular Modem, (3G/4G/5G), Zigbee, & 6loPAN
  - The bus lineup: Canbus, Profibus, & Modbus
  - Serial, SPI, I2C
  - Near Field Communication (NFC)
  - Proprietary

- Implications
  - Selected architecture must support (directly or USB/PCI) ALL
  - Drivers as well...possible port from different architecture
  - Enough performance
  - Maintain versions
  - Brace for the new
To the Cloud

- Data from sensors is the lifeblood of IoT
  - Connects to cloud or database
  - Gateways can filter/preprocess data
  - Push must be secure (encrypted and authenticated)
  - Connectivity is bi-directional so IoT Gateway must be secure from the cloud

- IoTivity
  - Community framework to connect end devices

- Alljoyn Open Source Framework
  - Connect and communicate across transports/OSes
Security
Recent Real-World Examples

• DHS confirms Public Sector Control system hacked
  – Attacking inadequate perimeter security, an attacker could compromise the SCADA system with capability to inject commands and read data at will
  – The controlled device was brought down for maintenance so no damage done

• Boeing and Airbus
  – Hacker used in-flight Wi-Fi connection to hack into flight control systems
  – Allegedly controlled thrust for engines, oxygen mask deployment, etc.

• Drones
  – Johns Hopkins University research demonstrated 3 different ways to send unwanted commands
  – Could force drones to land or just crash

• Personal vehicles
  – Jeep hacked through navigation and Corvette hacked by SMS
  – Activate wipers, apply brakes, disable engine & brakes
### Design Considerations

<table>
<thead>
<tr>
<th>Architectural</th>
<th>Functional</th>
</tr>
</thead>
</table>
| • Lifecycle: secure firmware updates and CVEs  
  – The Edge is relying on the IT-supported backend to handle the updates, requires careful consideration for the technology and process  
• Provide monitoring for end-to-end data on the Gateway  
  – Using DPI for heuristics-based detection of exploits  
• Combining types of security: physical, networking, system integrity and isolation of domains |
| • Building security primarily in the Gateway?  
  – Edge devices are constrained on hardened channel  
  – Requires encryption for the channel and two-way authentication for setup  
• Trusted edge vs. Edge Computing - two polars?  
  – Moving computing to the edge can help build end-to-end efficiency, but requires edge and gateway devices to handle the security  
  – Can also be seen as a way to fence out security threats for some layers of the processing so they cannot be exploited from the Cloud |
IoT Platform Virtualization & Security

- **Policy Configuration based on System Requirements**
- **Application**
- **LXC/Docker**
- **3rd Party Container**
- **Application**
- **LXC/Docker**
- **Guest VM**
- **vTPM**
- **DPDK**
- **KVM**
- **Root of Trust**
- **CVE Fixes and Maintenance**
- **Network Security**
  - IPSec HW offload
  - Policy Configuration
- **FIPS**
- **Live Patching**
- **Monitoring / Auditing**
- **SELinux / sVirt**
- **Embedded Linux**
- **Bootloader**
- **TrustZone (ARM)**
- **TPM (x86)**
- **Hardware Platform**

Network Security
- IPSec HW offload
- Policy Configuration
Types of Security Measures

• Reactive Measures
  – Common Vulnerabilities and Exposures (CVEs)
    • [https://cve.mitre.org/](https://cve.mitre.org/)
    • The standard list for holes in common systems
    • Very important to cover the affected parts in your product; MontaVista will do this for you
  – Intrusion-detection systems
    • Take action based on perceived attack
    • Several systems exist for Linux (LIDS, auditd, inotify, tripwire..)
  – Auditing and logging
    • Knowing you’ve been attacked prevents further damage
    • Collect evidence for litigation against the attacker
    • Example tools: Auditd, syslog, inotify, SELinux..

• Proactive Measures
  – Mandatory Access Control (MAC)
    • Minimizes the damage that unknown exploits can do to your system
    • Increases the chances to block 0-day exploits (unknown vulnerabilities)
  – System Certification
    • Provide Common Criteria or similar certification for your product or platform
    • MontaVista’s Linux is certifiable and we can help with the process
  – Root of Trust
Virtualization Technology for Isolation

- Full featured and lightweight virtualization solutions
  
  - **KVM**
    - Full virtualization
  
  - **Docker**
    - Application containers
  
  - **LXC**
    - Full-system Containers
  
  - **Core Isolation**
    - Dataplane and RT applications
“Do... or do not. There is no try.”

- Make Security a Priority
- Implement Mixture
  - Reactive
  - Proactive
- Stay Current
Sorry
We're
CLOSED
IoT Maintainability Requirements

• Product life cycle support
• Ability to upgrade application, kernel, drivers, userland, or whole system
• Upgrades done with little to no “human” interaction and downtime
  – Wireless delivery
• Secure updates
  – Authentication
  – Encryption
Addressing IoT Maintenance with Linux

- Long Term Support (LTS) Kernel
  - Can be extended beyond 10+ years in commercial Linux distributions
- SMART package manager
  - Allows for source or binary distribution
  - Flexible to update userland, application, etc.
- Live kernel patching
- Crypto API support
- Trusted Platform Module (TPM) and TrustZone for secure OTA updates
IoT: Signed OTA Updates

- IoT devices and Gateways have embedded requirements for small footprint but still a very high demand for security
- The process relies on the Kernel Live Patches, RPMs, or Container images being hashed and signed by a certificate that can be validated by the TPM or TEE on the target system if necessary
  - Can also support two-way signatures by using standard RPM signing using GPG keys, potentially enforced by the server-side TPM.
- Such processes are adopted by OSVs like Symantec, Redbend and practically all product manufacturers that are concerned about running trusted/secure SW on the devices.
- Without secure updates, the integrity of the platform cannot be maintained.
Summary

- Embedded Linux offers solid software platform to IoT Gateway developers
  - Architecture
  - Connectivity
- Security is IMPORTANT to implement
- High uptime maintainability
Thank You

Questions/Discussion

jgallagher@mvista.com