Security aspects of feature rich, connected embedded devices

Till Adam - KDAB
Nicolas Mayencourt - Dreamlab
Who are we, and why are we on this stage?

Till Adam
- Responsible for services at KDAB
- Veteran of Qt consultancy
- Helping people build devices

Nicolas Mayencourt
- Founder and CEO of Dreamlab
- Veteran of security research and consultancy
- Helping people secure devices
The Big Picture

The Internet of Things

The Mobile Age

Commoditization
OUTSIDE is Scary
The Internet of Things

Nearly everything is expected to be connected to the internet.

Your thermostat, your toothbrush, your TV, your blood pressure monitor and your washing machine might already be.

Whatever you build as your job, chances are it will be connected very soon, if it is not already.
Everybody has a smart phone.

The user experience expectation is dominated by mobile devices and gaming consoles, not PCs or kitchen appliances.

Touch screens are becoming ubiquitous.

Instant access from anywhere and any device is the norm.
Commoditization

Cost pressure is increasing everywhere.

At the same time feature expectations increase.

More and more expensive hardware and software is needed.

As a result more and more commodity hardware and software is used to make devices.
Embedded devices are:

- becoming connected
- becoming very complex
- need interfaces “like the iPhone”
- use commodity boards, periphery, operating systems and tools
Changing Landscape of Embedded Devices
The Past

- Specialized hardware
  - Expensive
  - Integrated circuits
- Single process
- Optimized code (C/ASM/...)
- Update only via HW (EEPROM)
- No / limited connectivity
- Specialized solution / sw
Today

- Commodity Hardware
  - Full-blown Computer (SoC)
  - Cheap
  - Massive interface support
  - Connectivity (IP/Wireless/...)
- Internet visibility
- Complex Software-Stack
  - Need for updates
- ... more like a Computer-Appliance
Security Assumptions Change

- Embedded Systems are black boxes
- ... can't be analyzed without extensive knowledge and funds
- How should one find our «secret backdoor» if even we have trouble using it?
- Endusers will always stay up to date with our newest release
- ... because they know how to update
- ... read our newsletter regularly
- No one will connect this device directly to the Internet
Why does this matter?
I am not a target.

IOT & commoditization is a gamechanger
Really?
Electronic crime is different
Something to protect?
• Chinese appliances are shipping with malware-distributing WiFi chips
• 29. Oct. 2013
I'm not a target?

Infect System

Valuable Information?

YES

Exfiltrate Information

NO

Access to other Systems?

YES

Attack Neighbours

NO

High CPU? Good Network?

YES

Mining / Processing SW-Distribution

NO

Missuse HW

- Spam Zombie
- DDoS Agent

Intelectual Property >??? $
Identity 10-500 $
CC ~10 $
Service Accounts <10 $
Valid Email 0.01 $
Reputation Hijacking ?
Available & cheap tools

- Standard IT-Security tools
- LogicAnalyzer
- BusPirate / JTAGULATOR
- USB-Oszilloscope
- USB-Microscope
- Public chip specifications
- Free compiler toolchains
- Cheap development boards
Past: Specialized Platforms & Attacks

- Automotive
- Medical
- Aeronautic
- Finance
- Military
- Industrial
- Consumer
Today: Commoditytization
One exploit fits all

- Automotive
- Medical
- Aeronautic
- Military
- Finance
- Industrial
- Consumer
Connectivity

GSM
UMTS
LTE
WLAN
Bluetooth
GPS
USB
Software / Hardware Stack

Bus / Network

Custom Firmware
- \( \mu C \)

Custom Software
- (RT) OS

Custom Software
- Apps
- Service
- Drivers

BSP
- Service
- Service
- Apps

(RT) OS
- Driver
- Driver
- Driver

Local Bus

Internet Services
Software / Hardware Stack

Local Bus

Service

Internet Services

Service

Apps

Cloud

Bus / Network

Custom Firmware

µC

Custom Software

(RT) OS

Custom Software

Apps Service Drivers

BSP

Service Apps

Driver Driver Driver

Different manufacturer

SoC

SoC
BSP's

Disclaimer: TI clearly states that this is not meant / fit for production

Disclaimer disclaimer: There's nothing wrong with the BSP form TI!

```php
<?php
    $command = $_GET['command'];
    $output = shell_exec($command);
    echo $output
?
```
What you see != What you get
What you see != What you get

- Leftovers on Disk / Firmware-Images
  - .bash_history (typos / internal staging servers...)
  - Log-files
  - Logos / documents from different customer
  - SSH-keys / credentials / SSL-Certificates
  - Demo-Software (from BSP's)
  - Unneeded services
  - Development leftovers (debug / backdoors)
- Outdated software / Backup-Files
- Deleted files (still visible)
- License violations
- We are not yet even talking serious security here
# Modern Software Protection

<table>
<thead>
<tr>
<th>Against existence of Exploits</th>
<th>✓</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe programming / secure coding</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>Input validation</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>Static / dynamic code analysis</td>
<td>✓</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Against exploitation</th>
<th>✓</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII armored address space</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>Stack guard / stack protection (Canaries)</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>No executable stack (DEP)</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>Address space layout randomization (ASLR)</td>
<td>✓</td>
<td>?</td>
</tr>
</tbody>
</table>
Modern Software Protection

<table>
<thead>
<tr>
<th>Feature</th>
<th>✓</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploit prevention / detection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Antivirus</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>Hostbased Intrusion Detection Systems</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>Intrusion Prevention System</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>Software maintenance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daily Updates</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>Encrypted Communication Channels</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>Signed Software</td>
<td>✓</td>
<td>?</td>
</tr>
</tbody>
</table>
Where does it lead to?
Get Development Device

Texas Instruments AM335x Starter Kit
Fingerprint BSP

- Comes with openembedded „Arago 2013.12 – am335x-evm“
- Busybox Linux (Kernel 3.1)
- Telnet
- Dropbear sshd 2012.55
- lighttpd 1.4.32 (on port 80)
- thttpd 2.25b 29dec2003 (on port 8080)
- Only root user with default password „root“
- Matrix GUI (HTML5 / PHP Webapplication)
<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
<th>REASON</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>21/tcp</td>
<td>open</td>
<td>ftp</td>
<td>syn-ack</td>
<td>vsftpd 2.2.2</td>
</tr>
<tr>
<td>22/tcp</td>
<td>open</td>
<td>ssh</td>
<td>syn-ack</td>
<td>Dropbear sshd 0.51 (protocol 2.0)</td>
</tr>
<tr>
<td>23/tcp</td>
<td>open</td>
<td>telnet?</td>
<td>syn-ack</td>
<td>Dropbear telnet ssh 0.51 (protocol 2.0)</td>
</tr>
<tr>
<td>80/tcp</td>
<td>open</td>
<td>http</td>
<td>syn-ack</td>
<td>lighttpd 1.4.33</td>
</tr>
<tr>
<td>161/udp</td>
<td>open</td>
<td>snmp</td>
<td>udp-response</td>
<td>SNMPv1 server (public)</td>
</tr>
</tbody>
</table>
### Find Devices

<table>
<thead>
<tr>
<th>Description</th>
<th>Current Value</th>
<th>Set Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring Method</td>
<td>Ethernet Eagle</td>
<td>--- Select ---</td>
</tr>
<tr>
<td>Primary IP Addr</td>
<td>192.168.100.1</td>
<td></td>
</tr>
<tr>
<td>Primary Port</td>
<td>5050</td>
<td></td>
</tr>
<tr>
<td>Alternate IP Addr</td>
<td>192.168.101.43</td>
<td></td>
</tr>
<tr>
<td>Alternate Port</td>
<td>5050</td>
<td></td>
</tr>
<tr>
<td>Listen Port</td>
<td>5050</td>
<td></td>
</tr>
<tr>
<td>Automatic Update Interval</td>
<td>1 Hour(s)</td>
<td>--- Select ---</td>
</tr>
</tbody>
</table>
Locate Devices
Upgrade for free

July 24, 2013 – Oscilloscope
$800, 70MHz to $1600, 200MHz
Upgrade with a «Key-Gen»

/*
** XXXXX XXXXXX keygen / cybernet & the-eevblog-users
**
** to compile this you need MIRACL from https://github.com/CertiVox/MIRACL
** download the master.zip into a new folder and run
** 'unzip -j -aa -L master.zip'
** then run 'bash linux' to build the miracle.a library
**
** BUILD WITH:
**
** gcc rikey.c -I../MIRACL ../MIRACL/miracl.a -o rikey
**
.............
Mitigation - Hardware

Hardware

- Use implemented security features
  - Fuses
  - Flash read protection
  - Tamper-switches
  - ...

Mitigation - Software

- Know and control your Software-Stack
  - Drivers / BSP
- Apply Software-Version-Control
- Release-Management
- Production Software QA
  - Clean builds (Leftovers)
  - Strip debug symbols
  - Remove development backdoors
  - Remove unneeded software
  - Harden / tighten
How about Qt?

Qt helps:

• As a 'vendor'
• As a community
• As a role model
Qt as a vendor

Qt is a mature and well monitored code base.

Many mistakes were already made and corrected.

Good integration avoids security problems caused by interfaces.

Collective maintenance of sensitive code is safer.

Track record of responses to alerts very good.
Qt's source code is public, problems are found early and communicated openly.

Community members with security interest and knowledge keep an eye on things.

Nothing can be swept under the rug.

Help is available.
Qt as a role model

Institutionalized review process.

Collective code ownership.

Culture of security consciousness.

Systematic encapsulation and clean architecture.

Strict release process with audit trail.

Easily reachable, responsive, responsible and professional security response team.
Best practices

Don't bypass Qt, use safe infrastructure (strings, sockets, SSL, XML parsing, temporary files, byte arrays, database access, etc.).

Accept that any connected device can and will be attacked.

Systematic threat modelling
  => secure by default architecture
  => security as integral part of the development process
  => security analysis of finished product
  => security analysis of operational context

Don't trust vendors, no matter how big.
Mitigation - Other

- Less is more!
  - Is full IP-Stack really needed?
  - Remove debug-interfaces physically
  - Reduce used software libraries / products
- Do security reviews
  - Before someone else does
  - Bug-Bounty-Programms?
  - Incident handling?
- Take customers (data-) privacy seriously