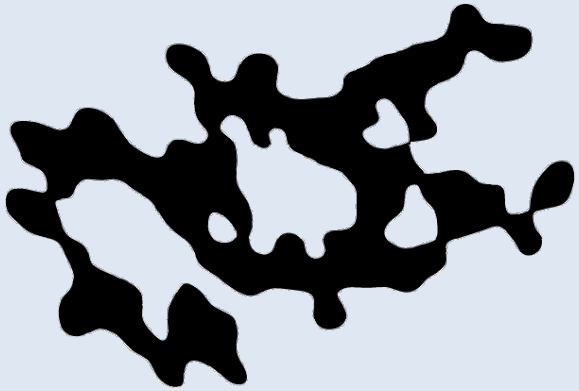
Machine-to-machine (M2M) security

Hunz
Zn000h at gmail.com



CCC Camp 2011
13.08.2011

Content

- What's machine-to-machine anyway?
- Attack vectors
  - Attacks over M2M communication channels
  - Physical attacks on endpoints (embedded devices)
- Attacking some actual M2M setups
  - Weaknesses
  - Attack
  - Impact
- Mitigation strategies
- Summary

What's machine-to-machine?

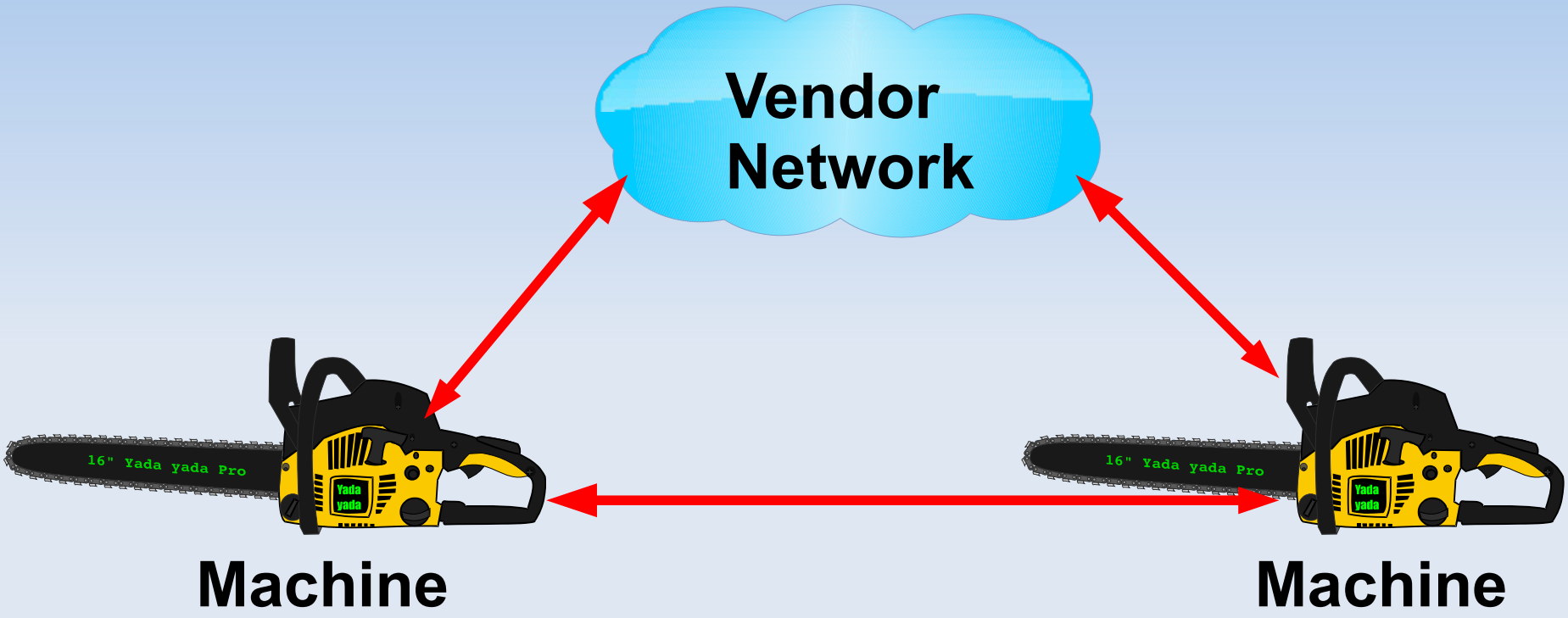
Definition guttenberged from wikipedia:

Machine-to-Machine (M2M) refers to technologies that allow both wireless and wired systems to communicate with other devices of the same ability. ...

This talk with a wider scope:

Machine2(Machine|Vendor|Maker)

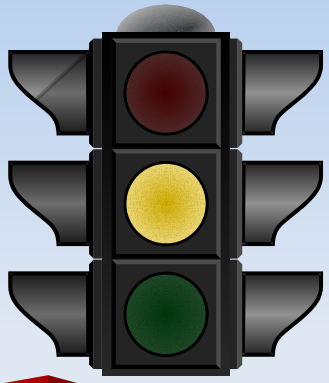
M2M communication

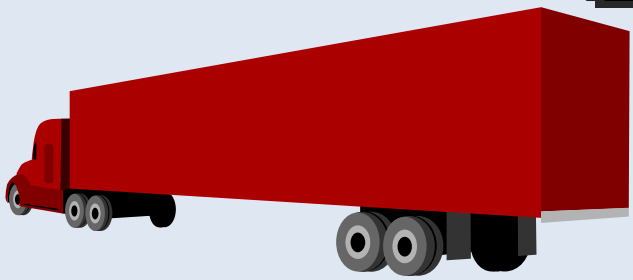


- Machines with embedded systems
- Focus here: devices with IP communication

Examples

- Smart grid (smartmeters, etc.)
- Vending machines
- Industrial control systems & machines
- Traffic control
- Motor vehicles
- (Entertainment devices (STBs, etc.))
  - Not really “machines”
  - But communication is similar





M2M de-mystified

- M2M is just a fancy buzzword
- There have been embedded systems with network access for years
- Example: PayTV STBs had integrated modem and dialup accounts in the firmware

- Now, there's just a lot more devices
- Some can do more immediate physical harm than normal PCs and PayTV-decoders

Communication channels

- Ethernet/Wireless LAN
- Mobile networks (GSM, 3G)
- Other (mostly ISM - ZigBee, etc. - not considered in this talk)

Ethernet / Wireless LAN

- Try usual exploits to compromise the device
- You know your tools

- But: Manufacturers know, that anyone can do some (Wireless) LAN hacking

  → Actual data often encrypted (SSL, VPNs, ...)

- Secret keys/certificates stored in devices
  - Physical attacks on devices (→ later)

Mobile networks: GSM, 3G

- Mostly GSM, less 3G yet

- Circuit-switched (dialin at vendor)
- SMS-based (for rare events & notifications)
- Packet-switched (GPRS)

- Contrary to (Wireless) LAN communication often no extra encryption
- “GSM is already encrypted”

Attacking GSM communications

- Passive sniffing + attacks on crypto
  - GSM (dialup/SMS): A5 broken for a while
  - GPRS: see GPRS talk of Karsten Nohl
  - Still, you probably want to send your own data
  - Either to device or network

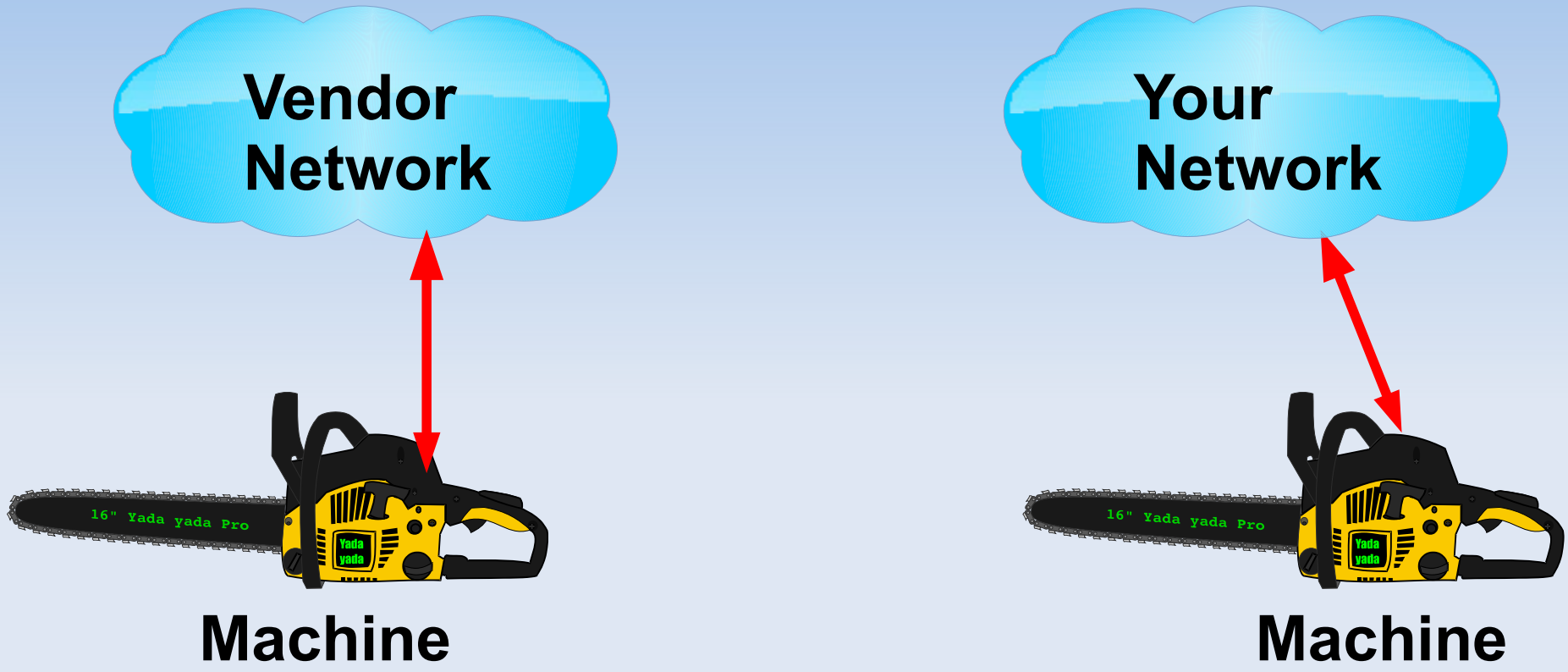
→ A rogue base station is your friend

→ there's OpenBTS & OpenBSC

Using a rogue BTS

- Interesting devices can be identified via IMEI
  - Type Allocation Code (TAC) identifies make+model of mobile equipment
  - There's a public TAC database: http://www.mulliner.org/tacdb/
- Make the device join your BTS
  - Some devices join foreign networks
  - Spoofing a “real” network is probably some kind of illegal...

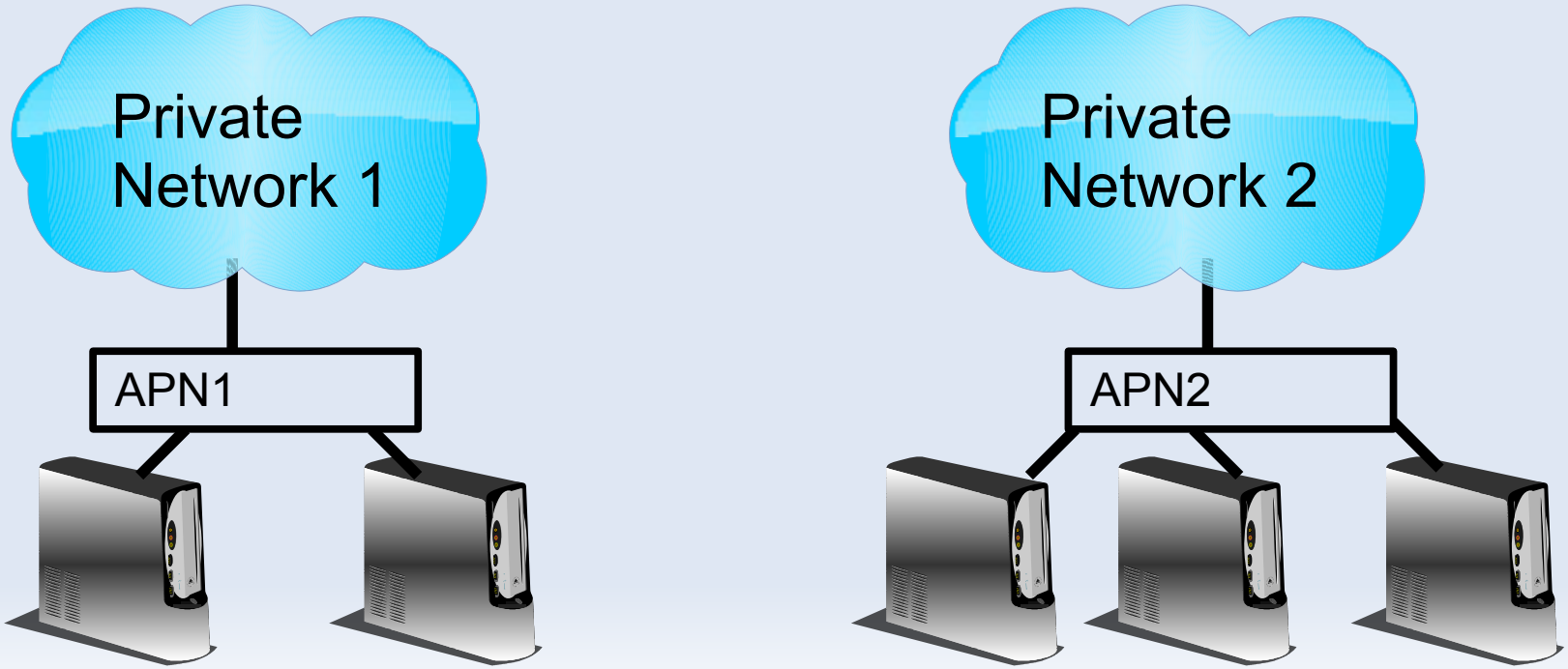
Using a rogue BTS (2)



- Device is isolated from “real” network
  - Attacking the device over the air possible
- What about the vendor network?

GPRS

- GPRS network access via Access Point Name (APN)
- There's the “normal” internet APN
- And special APNs for private networks



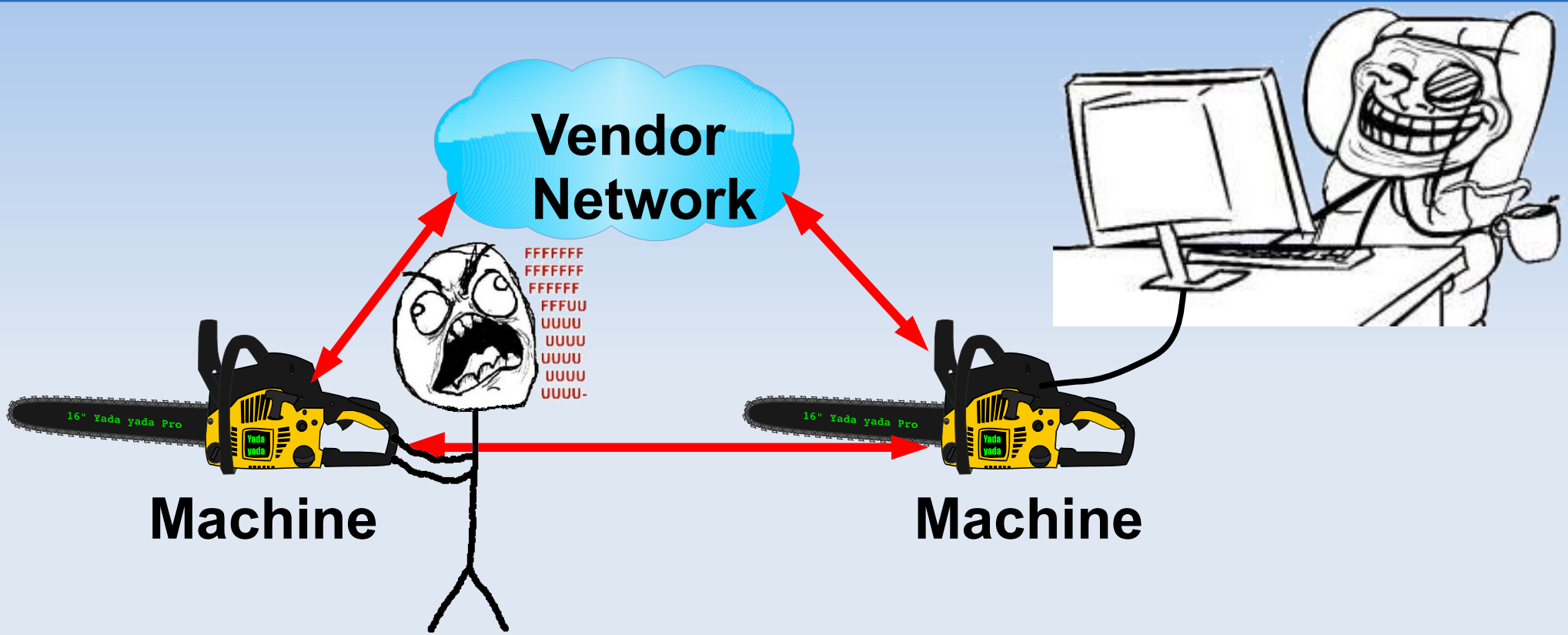
Mobile operators M2M solutions

- Authentication for special APNs
  - Via IMSI + GSM auth
  - APN Username/password
- How to get into the private network?
  - Physical attack on device (→ later)
  - MITM w/ rogue BTS and patched cellphone

MITM on GPRS

- Original device connected to rogue BTS
  - Build a bridge to original network
  - Probably needs some hardcore Osmocomm hacking
  - Sane GPRS encryption can prevent this

Attacks on endpoints



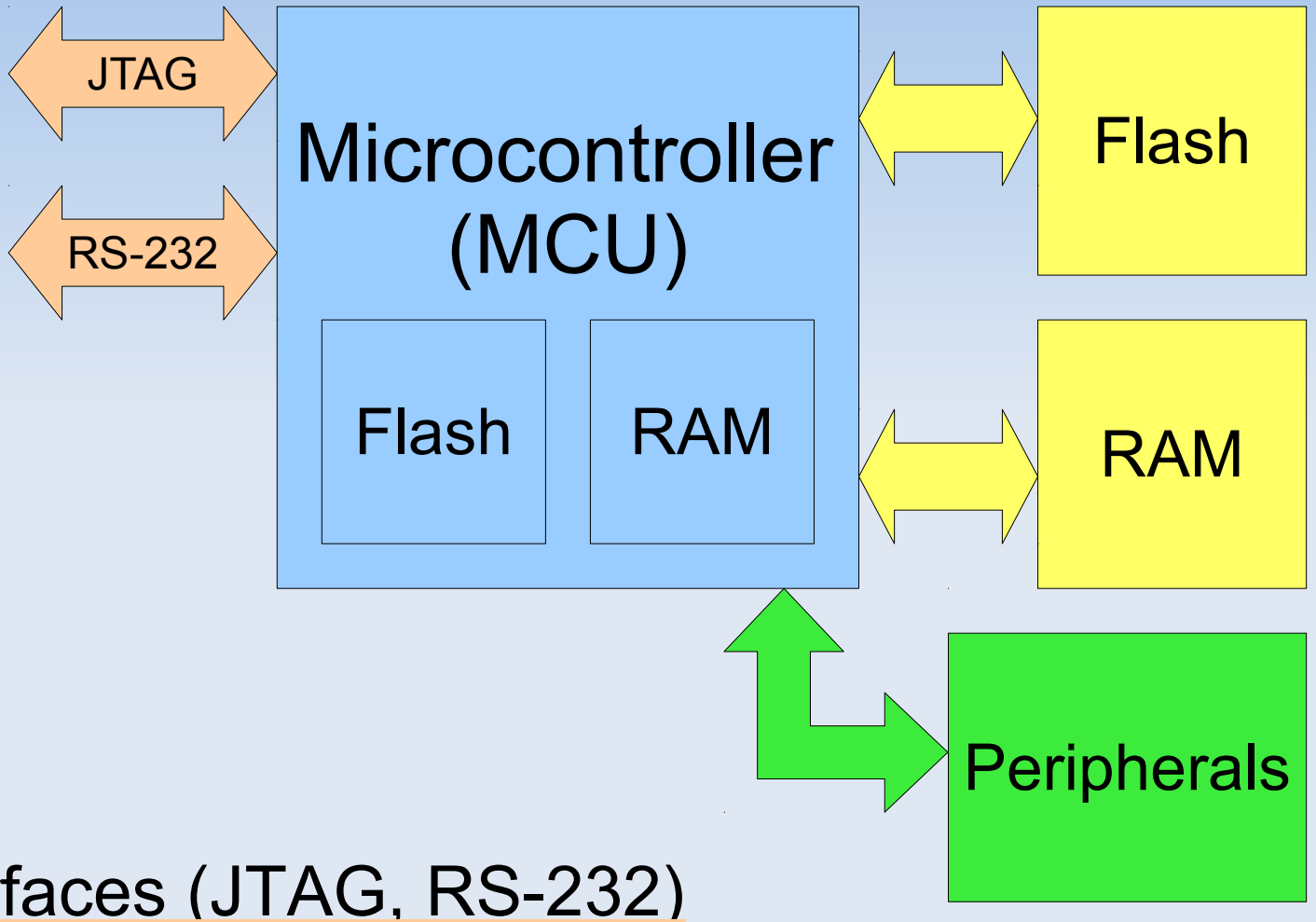
- Network can be compromised by rogue devices
- How to break into a device w/ physical access?

Embedded devices

- Often proprietary devices, less complex than PCs
- Design goals: low price, fast time2market, availability, safety, low-power
- Security features from the 80s or so
  - DEP? That's some fancy new shit!

- New problem: Hardware security
- Hardware security is difficult

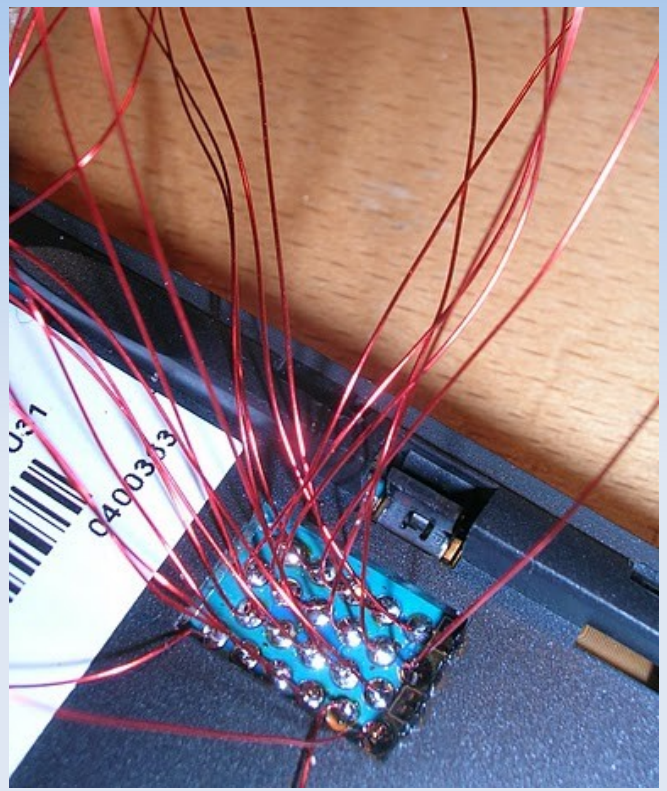
Embedded devices



Attack points:

- Debug interfaces (JTAG, RS-232)
- External memory
- Peripherals

Embedded devices: Debug interfaces



- Debug access
  - Bootloader or OS often has RS-232 access enabled
  - JTAG can be used to access the system

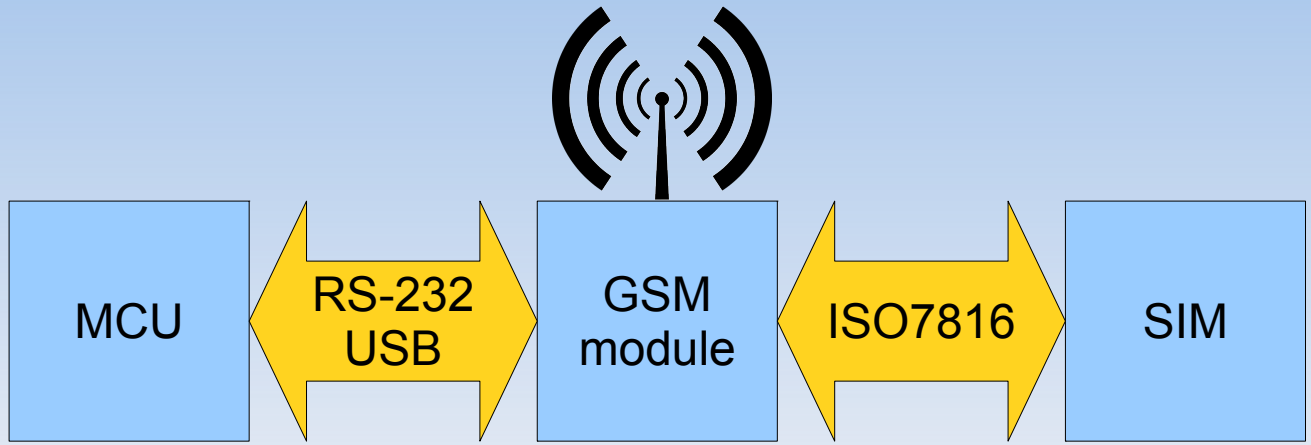
    There was a nice talk at the 26C3

Embedded devices: External memories

- External memory can be dumped/modified
- Most Flash-ICs can be read with a MMC-reader
- Otherwise a tiny microcontroller will do in most cases

- Look for other talks that cover hacking of embedded devices (too much for this talk)

Embedded devices: Peripherals



- Example: GSM module
- GSM/GPRS encryption done in GSM-module
- Communication between MCU, module and SIM not encrypted

  → sniffing & MITM possible

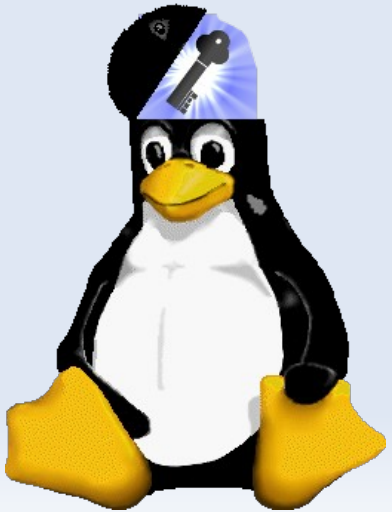
Fun with a M2M device

- Smartmeter
- Uses Ethernet w/ SSL end-to-end crypto

  → needs some secret key storage
- I can haz keys?

A closer look

- Physically disassembled the thing
- Traced RS-232 wires, connected a PC
  - But: nothing to see here
- Found boot parameters in a serial EEPROM by sniffing the I²C bus (used a Bus pirate)
  - Enabled serial console there
  - Got U-Boot prompt
- Ohai Linux!
  - init=/bin/sh, cat privatekeys
  - KTHXBYE

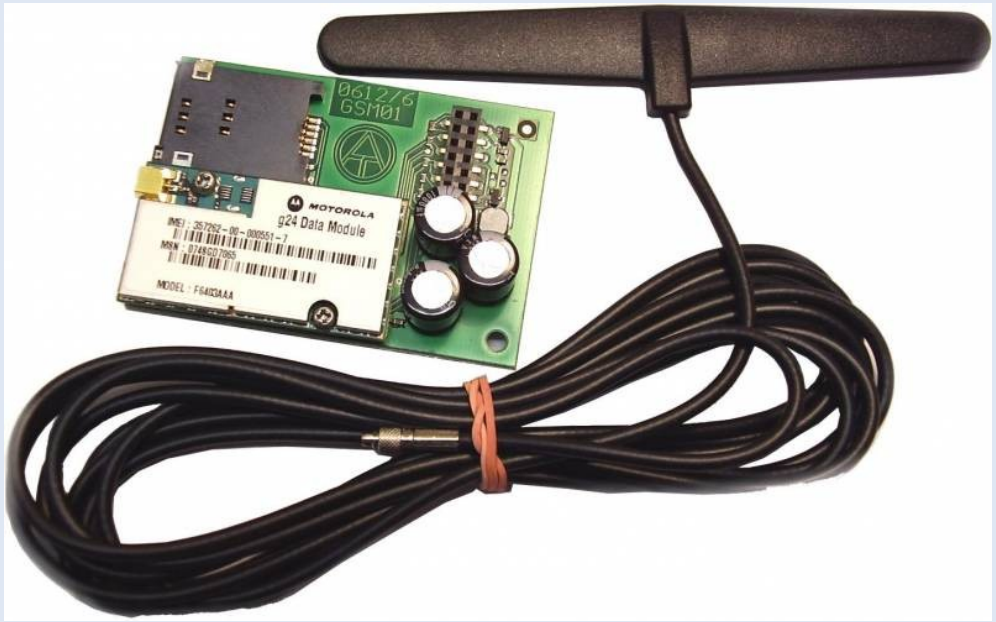


Roundup of this analysis

- Weaknesses
  - Unencrypted Flash memory in device
  - No internal Flash
  - RS-232 debug was easy to reactivate
- Attack
  - Reactivate RS-232 debug interface
  - Dump secret keys
- Impact: limited
  - Single device compromised (secret keys dumped)
  - No VPN access – just SSL to server

Fun with another M2M device

- GSM-based M2M module from some motor vehicle
- Bought via ebay
- Had a closer look at this thing



similar GSM module (image source)

A closer look

- SIM card present, but PIN-protected
- However, device sends PIN to SIM when powered up
- So I sniffed it :-)
  - Easy to do with a simple microcontroller or SIMtrace
- Used SIM in a phone with firmware patched to IMEI of M2M module
- Made a phone call with that SIM
- SIM still active :-)

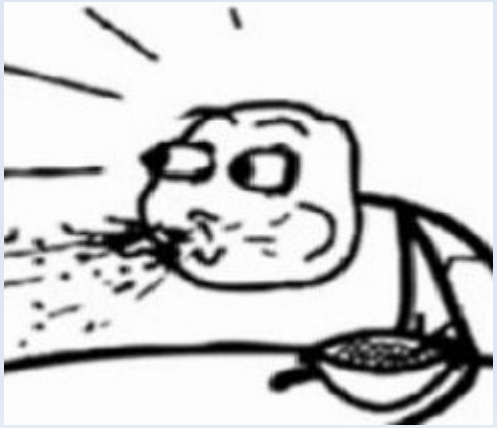
A closer look (2)

- Hooked the original GSM module w/ original SIM up to a PC via USB
- AT-command interface via USB (/dev/ttyACM*)
- *AT+CGDCONT?* to show APN

→ special-maker-apn

A closer look (3)

- IP interface activate! (normal PPP stuff)
- Private IP range, no Internet access
- Started pinging some IPs...
  - Some IPs w/ rtt of several seconds
  - Huge rtt variations
  - Suggests those IPs are of moving vehicles!??



Roundup of this analysis

- Weaknesses
  - Device side: PIN number can be sniffed
  - Network side: Generous packetfilter configuration
  - No rogue device detection?
- Attack
  - Use M2M module with PC
  - Connect to vendor network
- Impact: frightening
  - Extensive access to vendor network

Mitigation strategies

- Hacker Space Program needs sane M2M security!

- Based on previous findings:

  What can be done?

- Two attack vectors:
  - Attacks over communication channels
  - Physical attacks on devices

Securing communication channels

- This one is easy – at least in theory:

- **Never trust the communication channel**
- Always use **extra + sound** (well-reviewed) **encryption + authentication**

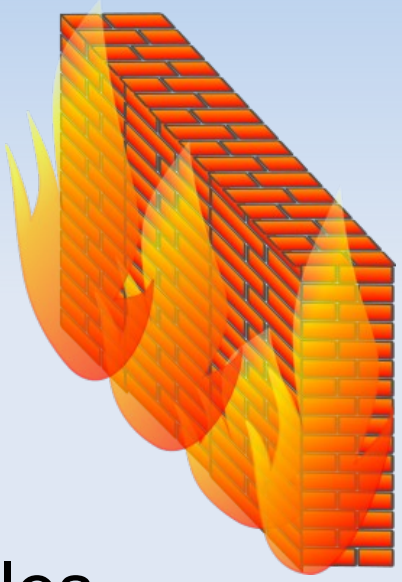
- Secret data needs to be stored in the devices
- Here, things get more complicated

Securing embedded devices

- Basic idea: Protect the secret data
- Disable debug interfaces
- Internal memory of microcontroller for secret data
- No unencrypted secret data over external busses!
- Tamper-detection

Rogue devices

- Hardware security (expensive?)
- **Still: accept that devices will be compromised**
- Early detection of rogue devices
  - Behavior profiling
  - Easy to realize: Well-defined action profiles
- Limit impact of rogue devices
  - Easy to realize: Well-defined action profiles
  - Whom do they need to talk to? → paketfilters
  - Device-individual secret data (=keys)!



Summary

- Currently: M2M security? Hard to find!

- Problems identified
- Some mitigation strategies provided
- What needs to be done
  - Manufacturers need to consider security
  - Network operators should provide some M2M security guidelines to their customers
  - M2M security initiatives? Awareness?

- Thanks for your attention!