Cursul #3

Internet of Things Security

3/10/2025

Outline

- Intro: IoT architectures
- IoT Security challenges
  - Cryptography in IoT
  - Provisioning & Root of Trust
  - IoT: Remote control & upgrading
    - Local vs cloud-based backend
- Case Study:
  - IoT security incidents

IoT Architectures

Intro: The “Things”

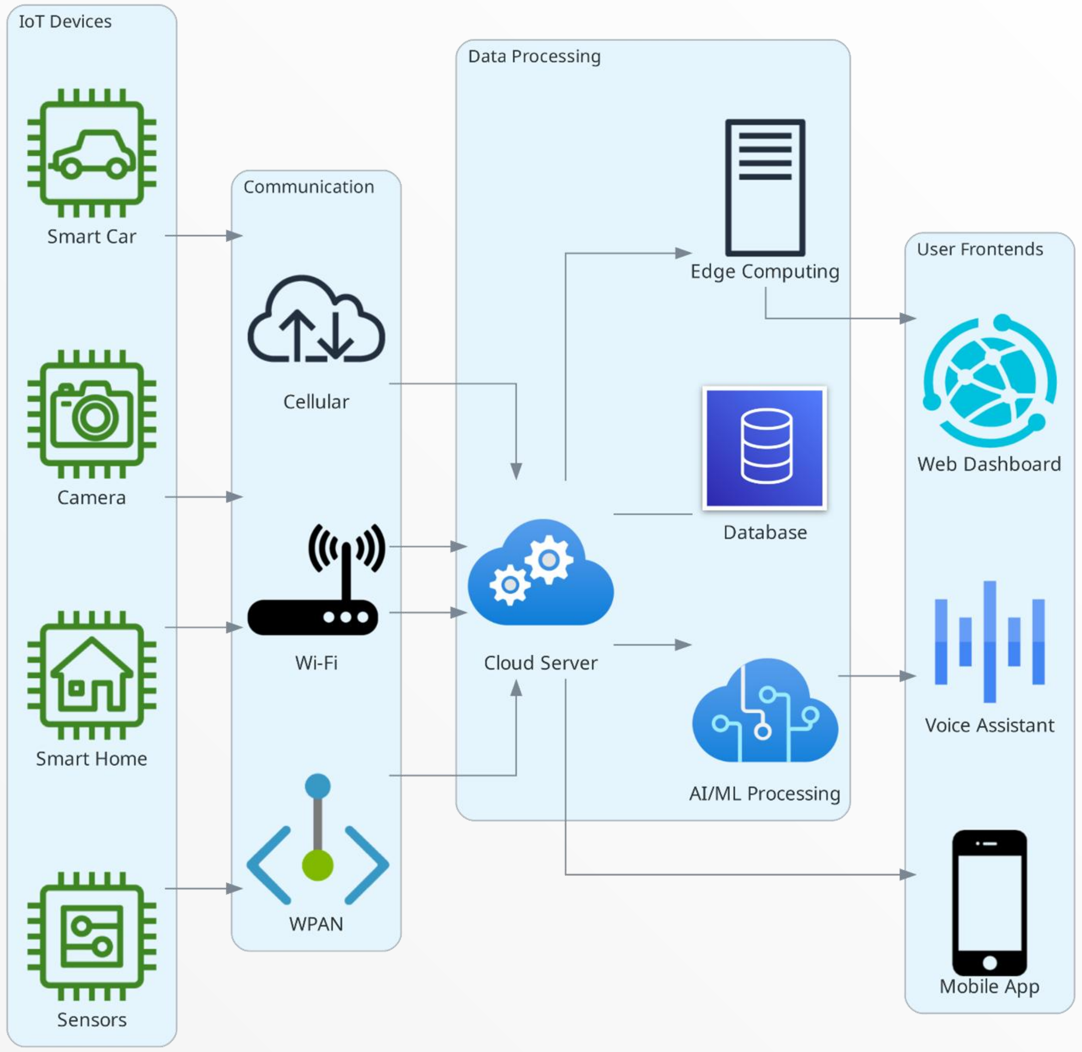
- Computers: PCs, laptops, mobile phones
- Networking: routers/gateways, NAS, VoIP…
- Entertainment: smart TVs, speakers, consoles
- Physical Security: cameras, doorbell / locks etc.
- House Appliances: thermostats, vacuum cleaner, fridge, air conditioning, smart lights etc.
- Wearables, medical etc.
- Transportation: e-scooters, cars, gas/charging pumps
- Industrial equipment (!!!)
- *Things + Internet = Security Nightmare*

IoT Characteristics

- Processing Power
  - Application CPUs vs Microcontrollers
  - Freq.: 10MHz ~ 1GHz
  - Storage range: 10KB ~ 100GB (Flash)
  - RAM range: 1KB ~ 1GB
- Software: Linux / RTOS / bare-metal firmware
- Connectivity:
  - [W]PAN: USB ;) Bluetooth, IEEE 802.15.4 (ZigBee, Thread),
  - Local LAN: WiFi, Ethernet / Serial (wired) etc.
  - WAN: LoRa, 2G ~ 5G mobile networks, NB-IoT, SigFox

IoT Architecture

- IoT device hardware
- Networking protocol & topology
  - Distributed/centralized
  - Directly / via gateways
- Data processing & storage
  - Cloud / Edge
- User frontend
  - Local vs Cloud



IoT Architecture

IoT Security: challenges

- Cryptography:
  - Big problem for microcontroller-based devices
  - Symmetric -> more efficient -> key provisioning ??
- Provisioning & root of trust:
  - Device identity & integrity
  - Vendor lock-in / security by obscurity!
- Firmware security updates
  - EOL products? :((
- Remote control of devices
  - Hopefully, by their rightful owners only 🥹

IoT Security

Cryptography

- Math processing: 128-bit vs 2048-bit numbers
- Ciphers: AES, RSA, ECC, ECDSA, ChaCha20-Poly1305
- Embedded Libraries: mbedTLS, WolfSSL, SharkSSL…
- MCU requirements for TLS:
  - ~50-100KB RAM, ~500KB flash
- Common workarounds:
  - symmetric ciphers only [don't!] => hardcoded keys => attacker can easily extract them and spoof any device!
- No problem for Linux IoT devices (app-grade CPUs)?

Cryptographic protocols

- TLS – de-facto standard, very high complexity
- Use lightweight cryptographic algorithms [1]:
  - Elliptic Curve Cryptography (e.g., 224-384 bit numbers)
  - ChaCha20 (stream cipher) + Poly1305 (integrity)
- CurveCP [2]
- Custom TLS profiles [3]

[1] Dhanda, S. S., Singh, B., & Jindal, P. (2020). Lightweight cryptography: a solution to secure IoT. Wireless Personal Communications, 112(3), 1947-1980.
[2] Bernstein, D. (2011). CurveCP: Usable security for the Internet. URL: http://curvecp.org, 5.
[3] NXP, A Light-Weight TLS and X.509 Profile, URL: https://www.nxp.com/docs/en/white-paper/LWTLSWP.pdf

IoT crypto fails

- Gree HVAC [1]
  - AES128 encrypted JSONs, hardcoded key (ROM), network sniffable!
  - => easy to integrate into Home Assistant ;)
- “Millions of embedded devices use the same SSH and TLS private keys” [2], top 10 hardcoded keys CVEs [3]
  - *“Another 80,000 devices, mostly WiMAX gateways from Green Packet, Huawei Technologies, Seowon Intech, ZTE and ZyXEL, use a “MatrixSSL Sample Server Cert” certificate.”*

[1] https://github.com/tomikaa87/gree-remote
[2] https://www.networkworld.com/article/945798/millions-of-embedded-devices-use-the-same-hard-coded-ssh-and-tls-private-keys.html
[3] https://cqr.company/web-vulnerabilities/hard-coded-cryptographic-keys/

Provisioning

- First-time setup / pairing
    - Enroll device to cloud service, remote control etc.
    - Cloud-device mutual authentication!
    - Obtain shared secret (encryption/authorization key)
- Often, via mobile phone / app
    - Must protect against malicious apps / MitM...
- Use digital signatures & certificates (PKI)
    - Trusting server: hardcoded CA public key / hash
    - Device identity: must generate unique & secure private key for each device (e.g., during manufacturing)!

Root of Trust

- Device’s secrets may be stored in EEPROM / Flash / fused registers etc.
- Hackers may extract the firmware and/or keys via:
  - debugging interfaces (UART [1], JTAG)
  - SPI/eMMC probing (Bus Pirate, Raspberry PIs [2] ...); https://www.flashrom.org ;)
  - Firmware vulnerabilities, side channel attacks etc.
- Attacks: reverse engineering, cloud user/device impersonation, data pollution / database breaches etc.
- Use tamper-proof memory (few chips support this)!

[1] https://diverto.hr/en/blog/2020-06-30-Extracting-credentials-from-IoT-devices/
[2] https://hacklido.com/blog/379-firmware-extraction-from-spi-flash

Firmware Vulnerabilities

- Software is buggy
  - => security issues (e.g., buffer overflows)
  - complexity is often high!
- CVE infrastructure
  - responsible disclosure => software gets fixed
- Most embedded software uses open-source components
  - E.g.: Linux, FreeRTOS, mbedTLS etc.
  - Software Bill of Materials (SBoM)
- Solution => upgradable firmware

Firmware Upgrades

- Hard engineering problems:
  - Local vs remote (OTA), size vs bandwidth considerations…
  - Bootloader support for upgrades
  - Firmware validation / signatures
  - Reliability / rollback: A/B partitioning / FS overlays etc.
- Open-source solutions:
  - Microcontrollers: MCUboot (https://mcuboot.com/)
  - Linux: RAUC (https://rauc.io/), SWUpdate (https://swupdate.org/), Rugix (https://oss.silitics.com/rugix/)

IoT Gateways

- Some IoT devices require gateways to reach Internet
- Secure networking:
  - Use sound cryptographic protocols & authentication
  - Network compartmentation / isolation
  - Use stateful firewall on IoT device (e.g., netfilter)
  - Change default admin password (e.g., HTTP / SSH)
  - Keep firmware components up to date!
- IoT device discovery: https://www.shodan.io/
- Risks: lateral network movement
- Don’t forget: routers are IoT devices too!

Routers

- Most vulnerable consumer IoT devices are routers!
  - TP-Link, Asus, Huawei, D-Link, Netgear, Linksys/Cisco etc.
- Mirai (2017), Quad7/CovertNetwork-1658 [1] (2024)
- Primary issue: no updates after product launch!
- Personal recommendation:
  - Buy a Mikrotik (https://mikrotik.com)
    - 15-year old devices still receives latest OS [2]!
    - Supports latest protocols (e.g. Wireguard)
  - ... or OpenWRT-compatible products (most TP-Links are!)

[1] https://blog.sekoia.io/solving-the-7777-botnet-enigma-a-cybersecurity-quest/
[2] https://www.reddit.com/r/mikrotik/comments/1ilrm9o/the_rb750_was_launched_15_years_ago_and_is_in/

IoT Hacking Tools

- Flipper Zero https://flipperzero.one/
  - Sub 1GHz frequencies supported!
- Sniffing + MitM: https://www.bettercap.org/
  - many protocols supported: USB, Wi-Fi, Bluetooth, CAN (requires external HW adapters) etc.
- Wi-Fi cracking: AirCrack-NG / https://pwnagotchi.ai/
- Hardware tools: BusPirate, HydraBus, Glasgow, cheap FT2232H-based dongles, Bluefruit LE Sniffer
- AirSpy / HackRF / RTL-SDR (Software Defined Radio)
  - Radio hacking! https://www.rtl-sdr.com/

IoT cloud services

- IoT equipment controlled by cloud
  - A device is as secure as its cloud!
- Technologies: MQTT, COAP, HTTP (REST-ful)
- Web vulnerabilities galore!
  - Insecure web/mobile API
  - Broken authentication/authorization
  - Insecure data transfer / storage
  - **Lack of privacy protections**

IoT cloud fails

- Hacking Kia [1], Subaru [2], Hyundai etc. (Sam Curry)
  - *HTTP Request to Unlock Car Door on the "owners.kia.com" website*
  - with just the VIN!
- *“Bug in Xiaomi smart cams showed users images from other homes”* [3]
- Verkada video surveillance service hack (2021 [4])
- *Internet of Dildoes* [5]

[1] https://samcurry.net/hacking-kia
[2] https://samcurry.net/hacking-subaru
[3] https://www.bitdefender.com/en-us/blog/hotforsecurity/bug-xiaomi-smart-cams-showed-users-images-homes
[4] https://www.verkada.com/security-update/report/
[5] https://sec-consult.com/blog/detail/internet-of-dildos-a-long-way-to-a-vibrant-future/

IoT Frontends

- Some IoT devices offer local control via mobile apps / LAN IP (SSH / HTTP)
- Local web server attacks: browser-based port scanning, CSRF, code injection (shell / SQL)
- Mobile: hardcoded passwords, MitM, lack of Bluetooth encryption etc.
- TrendNet webcam [1], Smart Locks [2], Hyundai [3] etc.

[1] Bugeja, Joseph, Désirée Jönsson, and Andreas Jacobsson. "An investigation of vulnerabilities in smart connected cameras." *2018 IEEE PerCom workshops*, https://www.diva-portal.org/smash/get/diva2:1409755/FULLTEXT01.pdf
[2] https://www.theregister.com/2024/04/15/critical_vulnerability_chirp_lock/
[3] https://cybernews.com/news/hyundai-bug-allows-unlocking-car-remotely/

Solve using regulations?

- US: IoT Cybersecurity Improvement Act, 2020
  - Guidelines + restrictions on federal agencies' use of IoT devices
- EU's Cyber Resilience Act [1]
  - Manufacturers must prioritize security, must perform cybersecurity risk assessment, provide updates for the entire prod lifetime ...
  - Product classification (default – self-assessment; Class I: OSes, Password Managers; Class 2: firewalls, routers – require independent body assessment & EU certifications)
  - Must provide SBOM & even open-source components must be certified (by a commercial support vendor)!
  - will start applying from 11 December 2027!

[1] https://digital-strategy.ec.europa.eu/en/policies/cyber-resilience-act