Security in Embedded Computing

René Burger

Hochschule Bonn-Rhein-Sieg

Table of Contents

Introduction

Motivation

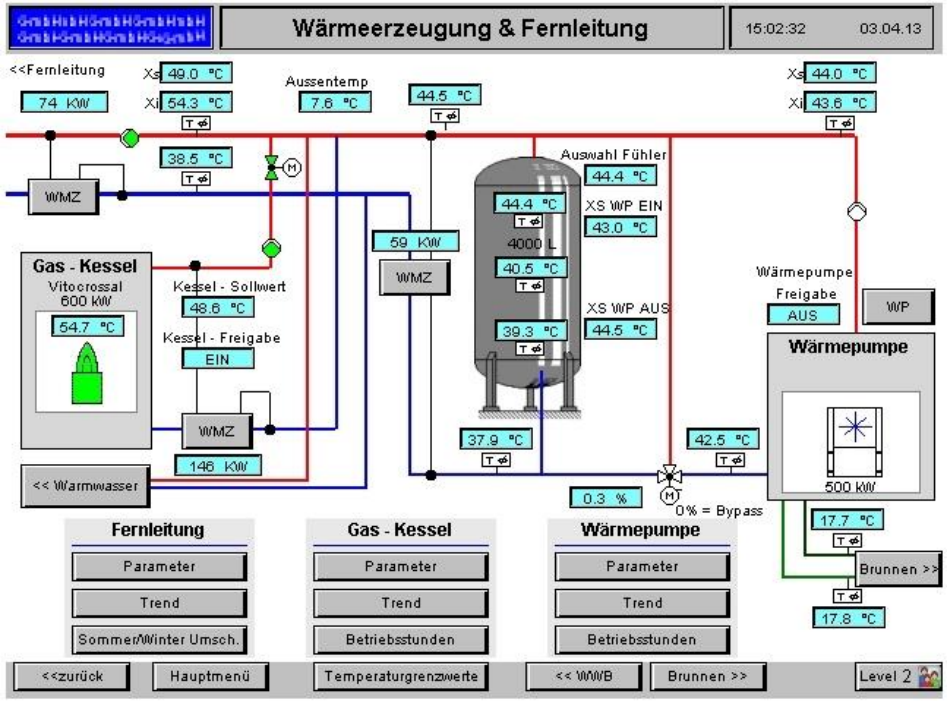
- Today, more and more devices get connected to the internet
  - Smart home: heating, lighting, home security (IP cams, motion sensors, door locks etc.)
  - Probably your washing machine and fridge will be accessible via internet in the next years
- Many embedded devices are used in nearly every field of technology
  - Smartphones
  - BluRay Players/ Smart TV sticks

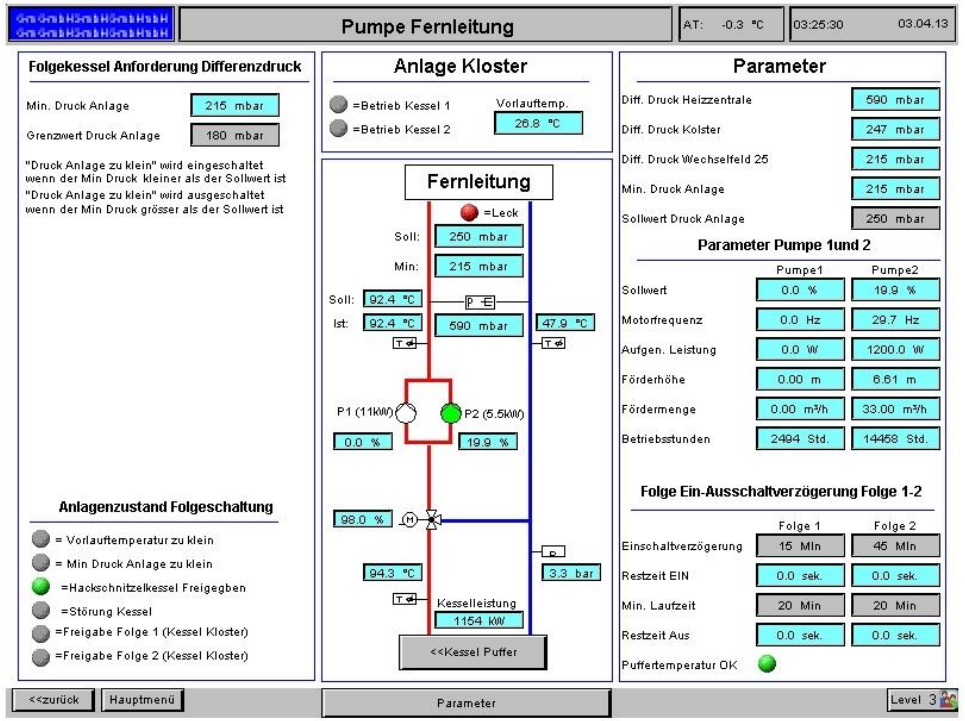
  →Also some with high-security requirements
    - Machines in hospitals, steering of industrial facilities
    - Cars (Tesla model S: Internet connectivity!!)

→ Security of embedded devices is important in our everyday life!

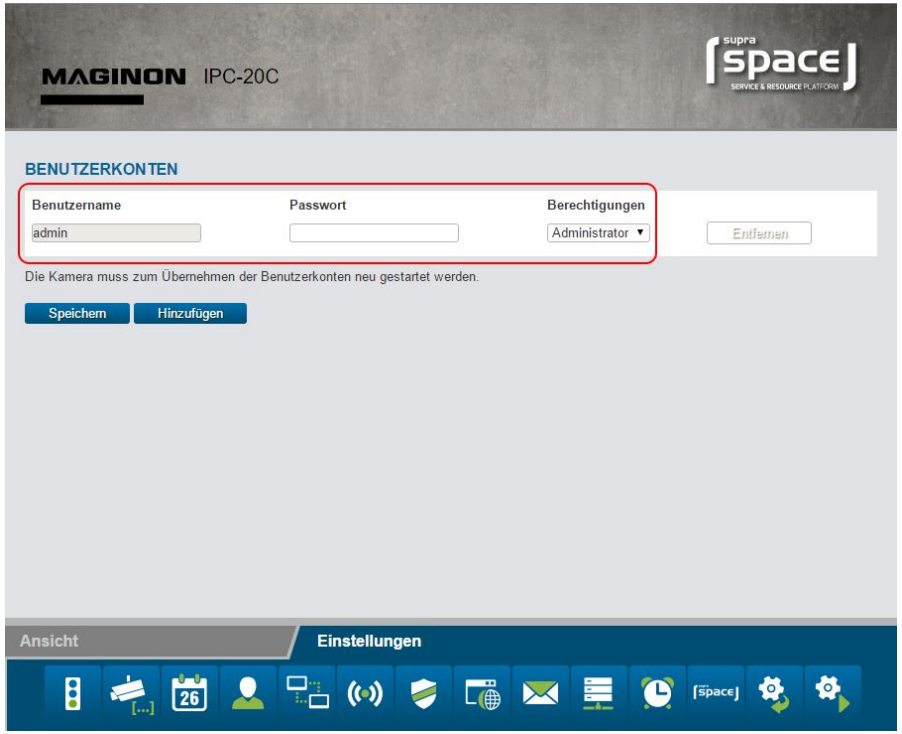
Then we see this:

- Did you ever want use to control a pumping station?
- Do you want to alter the brewing process of a brewery in Black Forest?
- NO PROBLEM: Just use the web interfaces you find from open port scans and try out standard passwords (c't 11/2013)





- Did you ever want to look into private houses with an IP cam of their owner? Just try the ones that were sold from ALDI in 2015 (heise.de, 01/2016)
- Many have not been patched yet



General Definitions of security terms

(from [Quelle TPM 2.0 Buch])

What means security?

Definiton of Security differs from the point of view:

- End-User: concerned of his personal data being stored/transported through the internet
- Manufacturer: wants to keep the secret of the proprietary firmware on his device
- Content provider: wants copy protection of his media (DRM)

→A malicious entity could steal your data, impersonate you, or change documents without being detected

→also the malicious entity can vary, depending on who wants which information

Secrecy (Datengeheimhaltung)

- Preventing an unauthorized observer of a message from determining its contents
  - Done by encryption of the message
- Message: An array of bytes sent between two parties

Shared secret

- A value that is known to two parties
- Can be
  - A password
  - An encryption key

Integrity

- An indication that a message has not been altered during storage or transmission
- If integrity is not tested, a man in the middle could change the message without the sender or receiver knowing about it
- Usually done by HMAC hashes

Authentication

- A means of indicating that a message can be tied to the creator

→Recipient can verify that only the creator could have sent the message

- Can be obtained by asymmetric encryption

Authorization

- Proof, that the user (or the device) is permitted to perform an operation

Anti-Replay

- A means of preventing an attacker from reusing a valid message

Nonrepudiation

- A means of preventing the sender of a message from claiming that they did not send the message

Example: electronic purchase in an online-shop

- Message: contains number of items ordered, confidential customer information (credit card number, address…)
- Integrity: Message can not be altered in transit (e.g. from 3 items to 300)
- Authentication: Proves that the order really came from the buyer
- Authorization: Checks that the buyer is permitted to purchase these items
- Anti-Replay: prevents the attacker from sending the buyer’s message again to purchase the items multiple times

Key- algorithms in cryptography

Secure Hashes

- Cryptographic hash: digests a message of any length into a hash of fixed length
- e.g. SHA-256 produces hashes of 256 bits / 32 bytes

Properties of a good cryptographic hash

- It is infeasible (really improbable, takes very much time to compute) to construct two different messages with the same hash
- It is infeasible to derive the message from a given hash

→ Hashes are building block of many other operations (HMACs, tickets, digital signatures…

HMAC – Message Authentication Code

- Keyed hash: secure-hash operation, but mixes in a shared-secret key (HMAC-Key)
- Only a party knowing the HMAC-Key can calculate the same hash
- Used to check that a user is authorized (because he knows the key) and that the message has not been altered (integrity)

Example:

- User sends a message containing a command to a machine, and HMACs the Message with his “authorization code”
- Machine knows that the user was authorized and the message is integer

Symmetric-Encryption Algorithms

- Uses a symmetric key to encrypt a message
- The same key is used to decrypt the message again
- Typical symmetric-key-algorithm: AES (advanced encryption standard)
- Used to provide secrecy, keeping data secret from all observers and to secure communication

- Symmetric encryption does NOT provide integrity or authenticity
  →To ensure this, you can HMAC the encrypted data

Asymmetric Algorithms

- Uses two Keys: one public, one private
- In RSA algorithm: private key can encrypt the data, public key can decrypt it and vice versa
- Calculating the public key from the private key is relatively easy, calculating the private key from the public key is computationally infeasible
- Much slower than symmetric algorithms
- Typical asymmetric Algorithms: RSA, ECC

Asymmetric Algorithms - Applications

- Signing data:
  - Owner encrypts data (e.g. a digest of a message) with his private key
  - Everyone can decrypt it with the public key, and knows only the owner of the private key could have encrypted that data
- To share data (usually a symmetric key to start encrypted communication)
  - Everyone can encrypt the message with the public key, only the owner of the corresponding private key can decrypt the message

How can communication be secured?

How can communication be secured?

- Example: Communication between a server and a client (client could be an IOT device)
- 1st Idea: Just use asymmetric encryption
  - Client uses the server's public key to encrypt the message, only the server will be able to read the plaintext
  - Server can use the client's public key to encrypt the message so only the client can decrypt it
- 1st Problem: Asymmetric encryption is really slow
  - Solution: use asymmetric encryption only to exchange a symmetric key
  - After key- exchange the communication is encrypted with the shared-secret symmetric key

How can communication be secured?

- 2nd Problem: How do we know that the public key is really belonging to the wanted server??
  - Secrecy won't be preserved if you encrypt a message with the attacker's public key
  - Man in the middle (MITM) attacks work like this: (here, the client wants to send a message to the server)
    - The MITM sends the client his public key (instead of the server's) and encrypts the message received by the client
    - Now he has the message in plaintext and can alter it in the way he wants
    - He then encrypts the altered message with the server's public key and sends it to him

    →No one knows that the message has been altered
  - → We need certificates to ensure, that a key really belongs to its owner

- If the public key of the server is certified, and the client sends a symmetric key to the server encrypted by this public key, a secure communication is obtained

How does digital certification work?

- A Certificate confirms, that a public key really came from whom it was supposed to come from
  - →A message (encrypted with the public key) really can only be decrypted by the declared owner
- A Certificate includes
  - The public part of the key being certified
  - Attributes of that key
- This is then signed by a certificate authority (CA)
  - Certificate is encrypted with the CA's private key
  - Can be decrypted by the (well-known) CA's public key

How does digital certification work?

- The CA's public key can also be certified by another CA's key → forming a certificate chain
- At the end, the chain terminates at a root certificate, that is conveyed to a verifier and that is trusted without cryptographic proof

Different kinds of software attacks

Exploiting a flaw in the design

- Discovering bugs and security gaps in the firmware of a system
- For example
  - An algorithm being used in the system, that is not safe anymore (SHA-1)
  - A security gap in the web interface of a router or printer
  - Unhandled variable overflows (C)

Using Brute Force

- Trying any combination of a key or password many times, until you have successfully found the right one
- A good cryptographic algorithm should be “computationally infeasible
  - It takes an impractical amount of time for trying every possible combination

  →Brute Force attacks won’t success
- Many passwords can be cracked by brute force attacks
  - A human has to be able to remember the password

  →it’s often NOT a random combination of Characters but contains words that can be found in the dictionary
  - “Good” attackers first try dictionary words, or combinations of words and numbers, special characters and so on

  → Brute force attacks are often successful when trying to crack passwords

How are embedded devices different from normal computers?

Embedded Systems have slower processors

- It is more difficult to implement safe cryptographic systems, because they often need a relatively high computing performance

Embedded Systems can’t be easily patched or updated

- Have you ever updated your router?
- Many parts of a system can’t be patched, because many drivers or other components of a system only exist as a “binary blob”

→ even if an OS is updated, underlying systems could have security gaps that often can’t be closed

Old chips often are being used for a long time

- If a security gap is discovered in an older chip, no one will be trying to fix it
- Even if you could patch the gap, maintaining older chips is not a priority

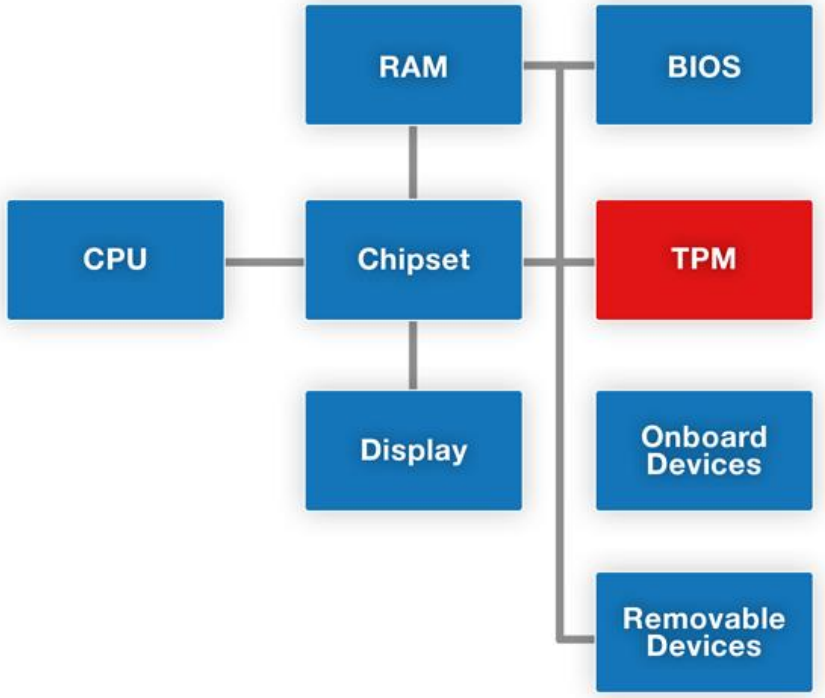
→ The security of an embedded system has to be considered in the design cycles. It is really hard to patch an embedded system after it has been deployed.

TPM and HSM: Centers of Security

TPM

- TPM means “trusted platform module”
- TPM Module is a chip mounted on the mainboard of a device
- Used to perform cryptographic tasks

- A TPM can
  - Store and manage keys
  - Perform cryptographic computations
  - Store and manage certificates
  - Generate random numbers
    → Useful to generate new keys



Cases where a TPM can be very useful

- Identifying a machine - not by their MAC or IP address, but by security identifiers
  - For access to networks
  - To authorize a payment or to allow an action only authorized machines are allowed to execute
- Encryption
  - A TPM can encrypt and decrypt (so manage) keys for the encryption of files on the hard drive or to decrypt files that the system got from another device
  - Full Disk encryption (like BitLocker on Windows Systems with a TPM)
  - Encryption of Passwords (Password Manager) or other Data
- Checking if the Boot Sequence has not been compromised (by using PCR values)

HSM

- HSM: Hardware Security Module
  - External Module - in a PC or used for an entire Network
  - Also used for cryptographic processing and key- management (in large numbers)
  - Is also protected against tampers like bus probing, (attacks against the hardware) not only software attacks
  - Mostly used in very-high security environments like bank servers

Differences TPM/HSM

- TPM: Small onboard module with very low cost-factor
  - Mostly used for authentication and to manage a small amount of keys/certificates
  - Only used on a single device

- HSM: Bigger dedicated module, as extension of a PC, Server or Network
  - Used to very securely manage keys and certificates (tamper-resistance)
  - Sometimes also used for high-performance cryptographic computations

Hardware attacks:

Hardware attacks

- Systems can not only be compromised with software, but also with hardware methods (and expensive equipment and sophisticated techniques):
- Invasive attacks:
  - Probing inter-component communication, e.g. to read out memory
  - Reverse-engineering
  - Access to the internals of a device and the ability to manipulate and interfere with the system is needed
- Non-invasive attacks
  - Timing-attacks
  - Power analysis
  - Electromagnetic analysis
  - Try to detect what a component is computing, depending on which step (of a cryptographic algorithm) it executes, it draws different currents, takes more time or emits different electromagnetic fields

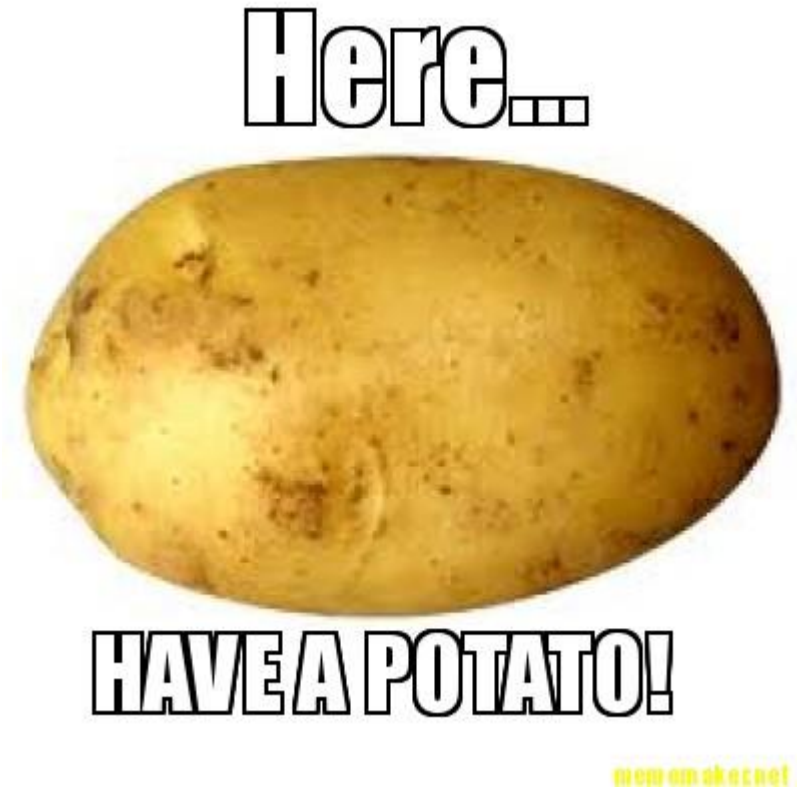
Short conclusion:

- Security gaps in embedded systems are really hard to patch
- It’s a real problem: Many IOT devices are vulnerable and easily accessible through the Internet
- Systems can be successfully secured, but only, if:

  Security is considered during the design process of a device.

  You can’t change much after release!

Thank you for listening!



Sources

- Security Risks of Embedded Systems, Bruce Schneier, 09.01.2014, accessed on 11.08.2016 https://www.schneier.com/blog/archives/2014/01/security_risks_9.html
- A Practical Guide to TPM 2.0, Will Arthur, David Challener, Kenneth Goldman, 2015
- Computer Organization and Design, David A. Patterson, John L. Hennessy, 2014 Elsevier Inc.
- Security as a New Dimension in Embedded System Design, Paul Kocher et al, *DAC 2004,* June 7–11, San Diego, California, USA, pages 753-760

Picture Sources

- https://upload.wikimedia.org/wikipedia/commons/7/7c/NShield_Connect_45_left.jpg
- http://electronicdesign.com/site-files/electronicdesign.com/files/archive/electronicdesign.com/content/content/74661/74661_fig1.jpg
- http://www.heise.de/newsticker/meldung/Verwundbare-Industrieanlagen-Fernsteuerbares-Gotteshaus-1902245.html
- http://www.heise.de/security/meldung/IP-Kameras-von-Aldi-als-Sicherheits-GAU-3069735.html