Automatic Control and Industrial Informatics Department
Faculty of Automatic Control and Computers

# Improving the security of a webservice: best practices and attack simulations

# Table of content

# Abstract & Introduction

- Most of the nowadays applications are client-server based
- Main target: expose a webservice from a low-level programming language
- Focus on best practices and security methods

# Objectives

1. Developing a webservice using C++
2. Gradually improve the security of the webservice
    1. Check internal and external dependencies
    2. Automate the management of resources
    3. Encrypt the communication between client and server
3. Analyze all data from empirical approach
4. Generate a series of conclusions

# Related work

- **Developed webservice:**
  - Armhf architecture
  - Raspberry PI
  - HTU21D humidity and temperature sensor
  - Get all data over I2C interface

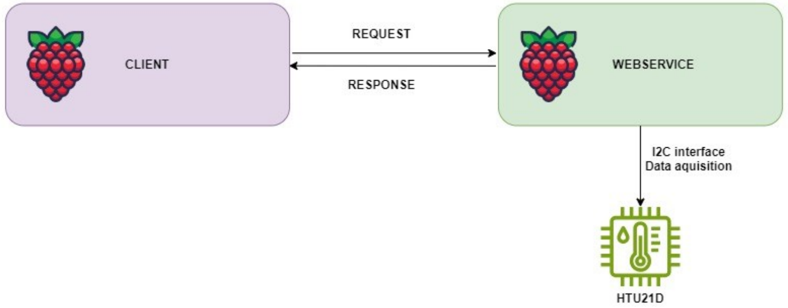


Fig. 1. Data flow from client to server

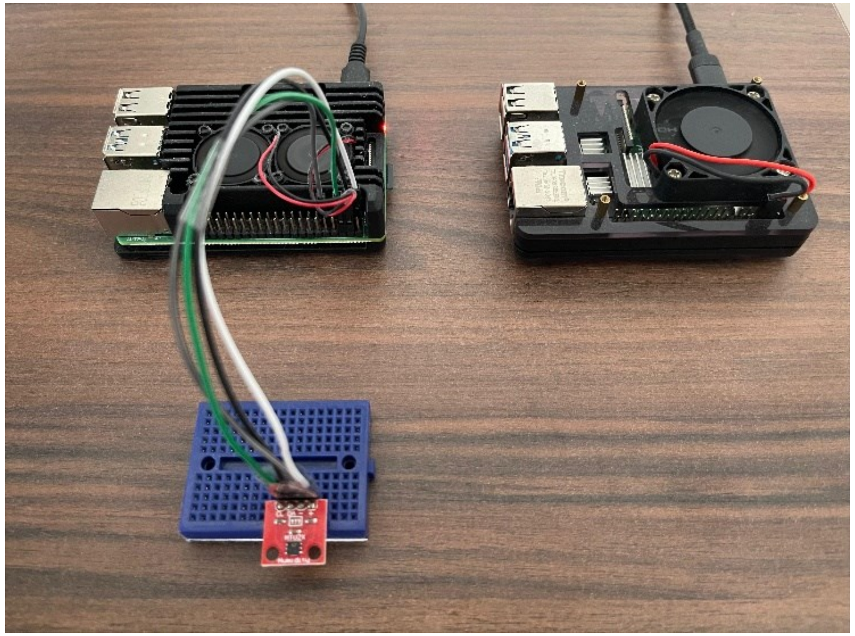

Fig. 2. Physical systems – server and client

# Security methods

**Checking internal and external dependencies**

- **How to:** check for all exploits found up to this point
  - National Vulnerability Database NIST
  - CVE Numbering Authorities
- **Result:** Avoiding potential exploits / Accepting and analyzing some possible consequences
- **Concrete example - webservice dependencies:**
  - Restbed: no exploits have been reported
  - RapidJSON: no exploits have been reported
  - OpenSSL: all versions have vulnerabilities
    - Ex: version 3.0.11 – CVE-2022-1434 (related to RC4-MD5 algortithm)

# Security methods

## Automating the resource management

- **How to:** Smart pointers & RAII idiom
- **Result:** Avoiding memory leaks
- **Why is important to avoid memory leaks?**
    - Such a small and seemingly harmless mistake can represent a significant opportunity for the attacker
    - Talk about a **concrete example**: a webservice which collects data from pressure, temperature and humidity sensors within an automated pipeline for the automotive industry

## Encrypting the communication between client and server

- **How to:** digital certificates and HTTPS
- **Result:** encrypted communication - end to end (confidentiality and integrity of data)

# Experimental results

- Empirical approach for validating the security methods
- All tests were conducted into a controlled environment
- Everything was done solely to validate some results and discover new possible exploits

- Man in the middle attack simulation
  - Method: ARP poisoning
  - Purpose: to validate the effectiveness of the TLS protocol

| System | MAC address |
|---|---|
| Client | E4:5F:01:45:D5:4C |
| Server | D8:3A:DD:19:9B:A8 |
| Attacker | 08:00:27:1E:36:4A |

Table 1. Systems used and their MAC addresses

# Experimental results

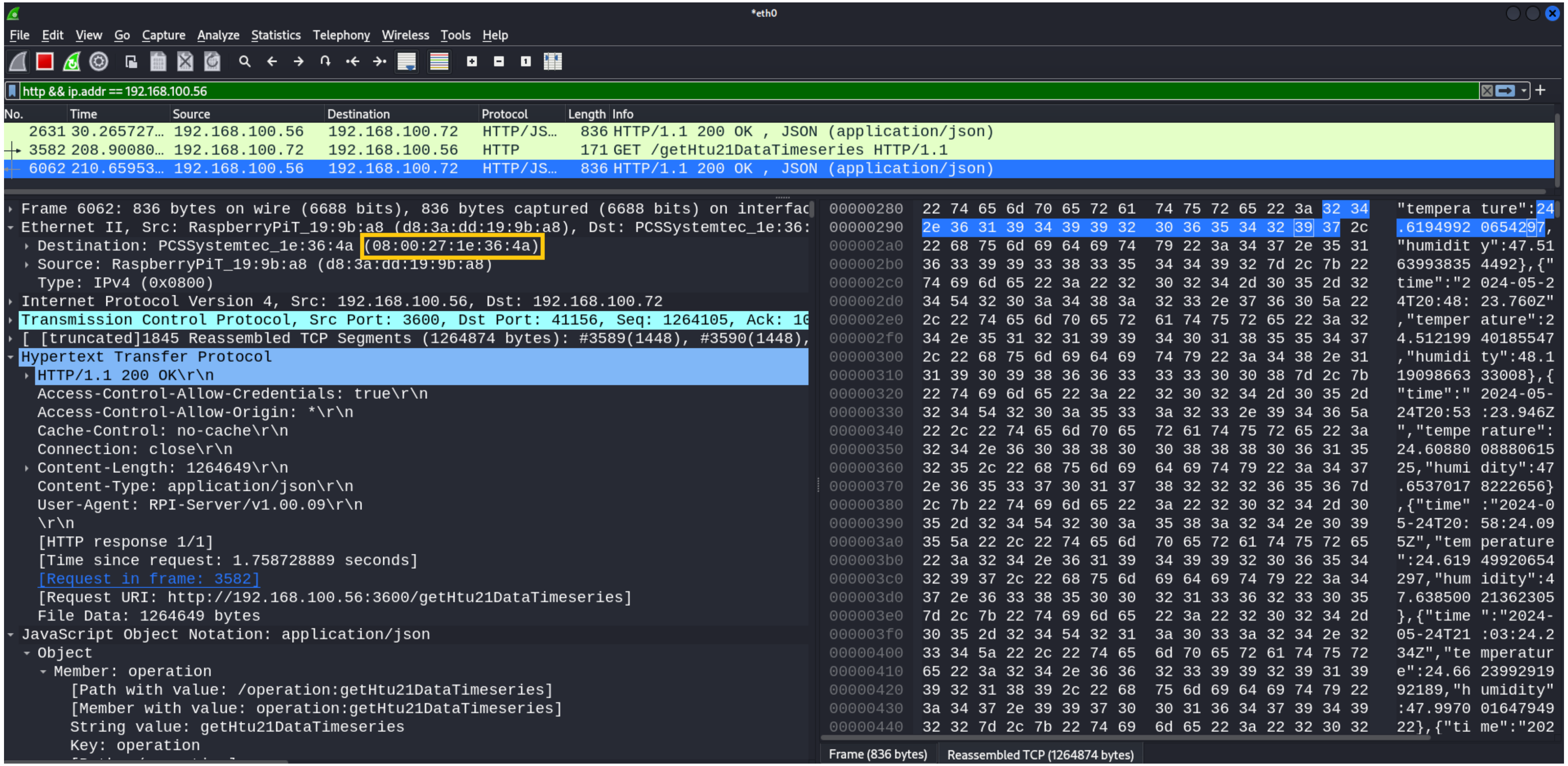


Fig. 3. Man-in-the-middle attack, HTTP protocol, no digital certificate and TLS disabled

# Experimental results

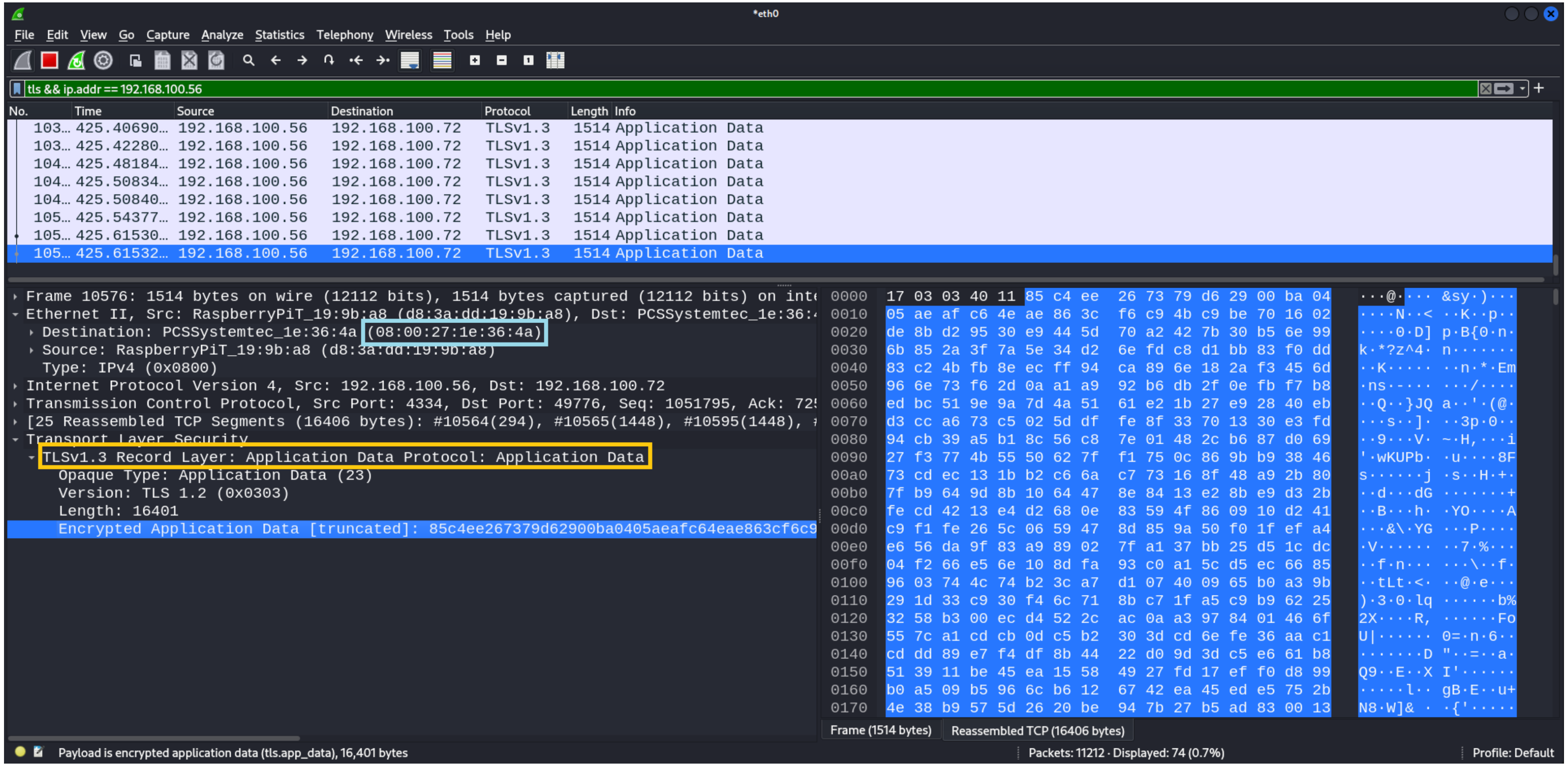


Fig. 4. Man-in-the-middle attack, TLS protocol enabled, digital certificate on server side

# Conclusions

- Combined a theoretical and empirical approach to improve the security of a webservice

- Proposed a series of good practices to be applied for avoiding memory leaks

- Our personal thoughts:
  - Digital certificates play an important role in mitigating MITM attacks
  - However, we did not find any scientific research which validates that this method provides 100% protection against MITM
  - The presence of digital certificates and TLS protocol do not guarantee that such an attack cannot be successfully executed

Thank you for your time and attention!