

**SAVONIA**

# **IoT Threats – Hardware and Software Vulnerabilities, DoS Attacks**

Cybersecurity Fundamentals

Markku Kellomäki

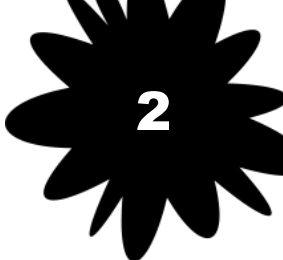
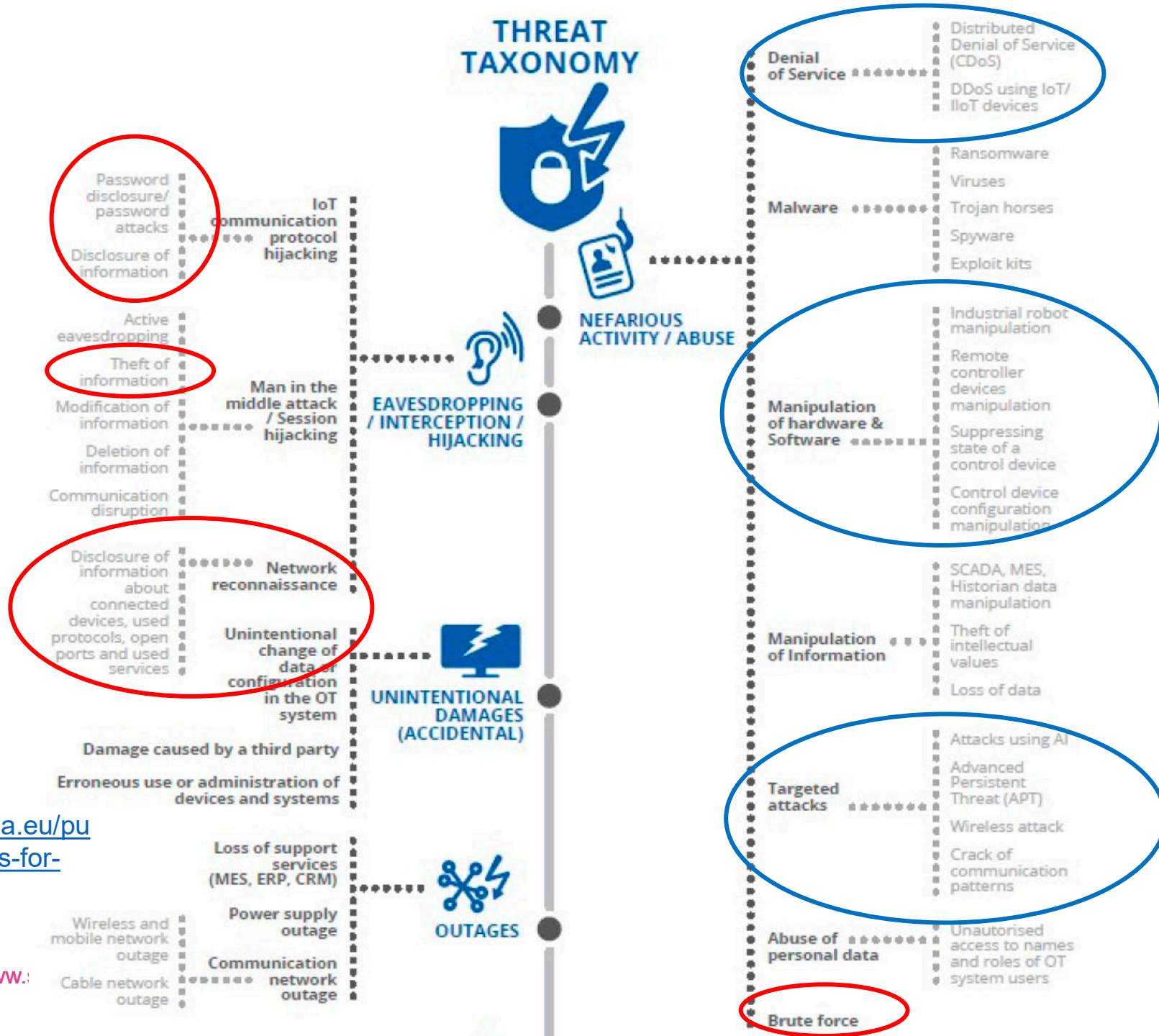


# Threats we covered last week

Lab 2 exercises

Today's lecture

## THREAT TAXONOMY



source:

<https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot>

## STRIDE

- Each threat is a violation of a desirable property for a system
- Today we will target Confidentiality and Availability

Threat	Desired property	Threat Definition
Spoofing	Authenticity	Pretending to be something or someone other than yourself
Tampering	Integrity	Modifying something on disk, network, memory, or elsewhere
Repudiation	Non-repudiability	Claiming that you didn't do something or were not responsible; can be honest or false
Information disclosure	Confidentiality	Someone obtaining information they are not authorized to access
Denial of service	Availability	Exhausting resources needed to provide service
Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

Source: [https://en.wikipedia.org/wiki/STRIDE\\_model](https://en.wikipedia.org/wiki/STRIDE_model)

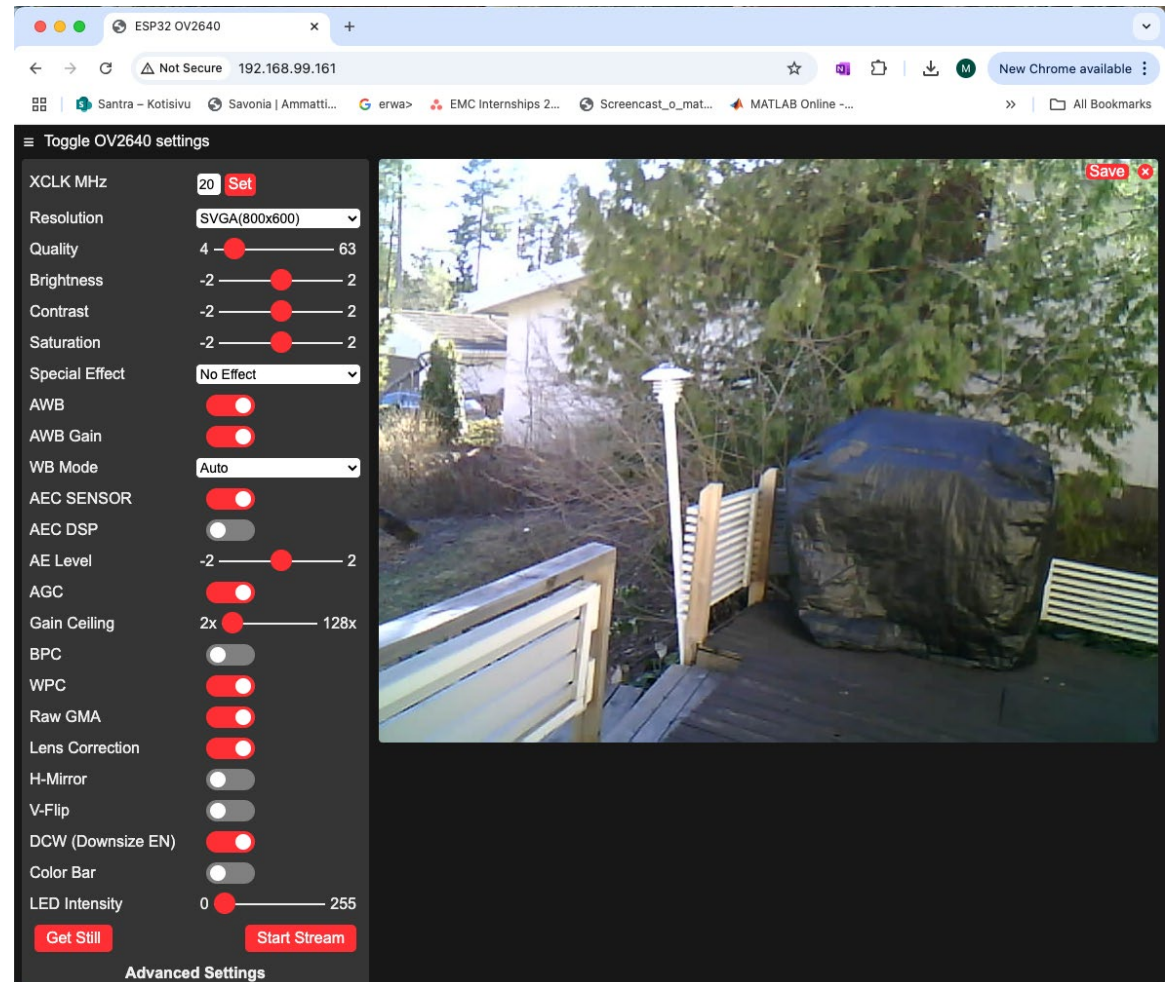
CIA triad

## Case study for today

- We become aware that there is a video streaming service hosted on the property of Company X
- It has been leaked / some intelligence done that this service is available in the following IP address: 192.168.99.161
- Our case:
  - Try to figure out if the network in which the streaming service is hosted is seen in our classroom.
    - Record the SSID
  - Find more information about the network
- Plot twist: one camera in the same network is found in the dumpster
  - What can be figured out from the hardware?
  - Gather information from the firmware of the device and try to get access to the network based on it.
  - If you get access, try to case a DoS attack against the video streaming service.
- Reporting:
  - Document the steps
  - Discuss this case: attack vectors involved, mitigation actions (how to prevent)

## Leak of available service

- We become aware that there is a video streaming service which runs in this IP address: 192.168.99.161
  - What can we learn about the security of this service from this photo?



CIA triad

## Network reconnaissance

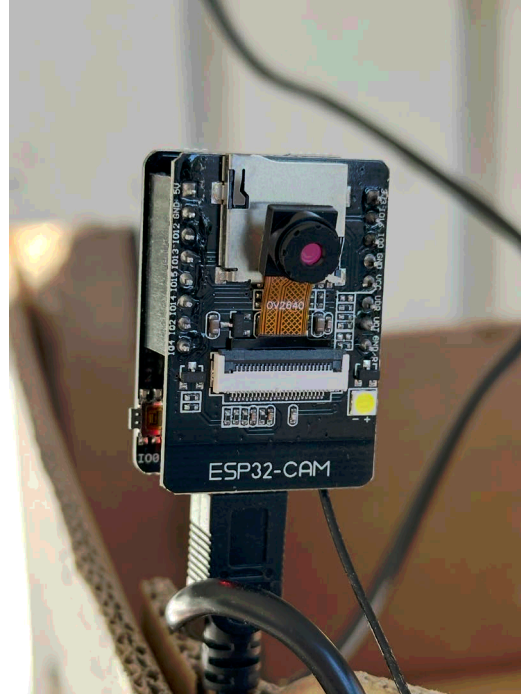
- Check which WiFi networks are available in the classroom.
  - Can you recognize names of organizations?
  - What do you know about their sizes? How many clients should they be able to serve?
- What can you tell about the service in question based on its IP address?
  - How many clients there can be in the yet unknown network?
  - What does this tell about the size of the organization?
- Does any SSID stand out?
- Check the signal strengths with some software or app.

## Network reconnaissance

- Does any SSID name stand out? List the suspected ones.
- Check the classroom. Do you see any suspicious device brought in by the teacher which might give a clue about the network?
- Write down the SSID or SSIDs when you have your suspects

## Plot twist: physical incident

- You realize someone takes a security camera to SER collection bin (electronic waste) or someone steals one of their security cameras. That camera ends up on your table.
- Opening up the camera reveals a microcontroller. Can you figure out what it is?



## **Do research on the microcontroller**

- Does it have any easily accessible ports?
- Is there a way to connect your computer to it?
- Can you view or download the contents of the microcontroller?

## Microcontroller flash memory contents

- It appears the microcontroller has USB port
  - Alternatively: UART pins are discovered and wires soldered
- Can the flash memory contents be read?
  - Search for methods for this particular microcontroller

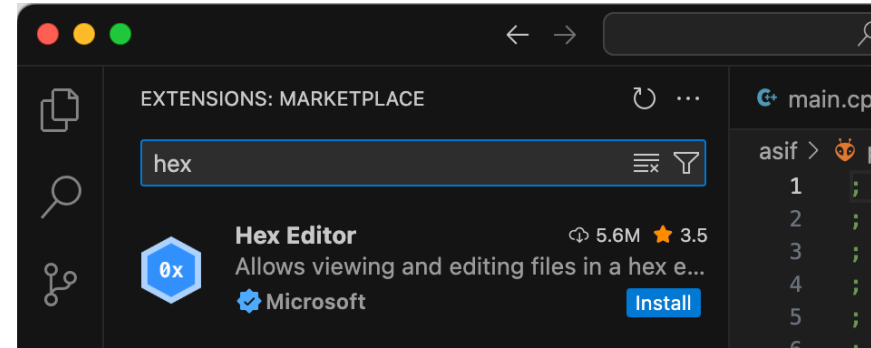
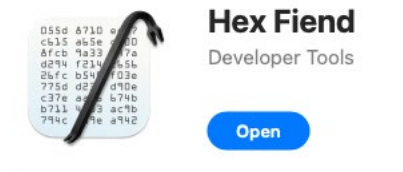
## Microcontroller type: ESP32 CAM

- Camera module documents: <https://docs.ai-thinker.com/en/esp32-cam>
- Microcontroller manufacturer: <https://www.espressif.com/>
- Software download tool for Windows:  
[https://docs.espressif.com/projects/esp-test-tools/en/latest/esp32/production\\_stage/tools/flash\\_download\\_tool.html](https://docs.espressif.com/projects/esp-test-tools/en/latest/esp32/production_stage/tools/flash_download_tool.html)
- Arduino IDE includes a tool which can be used as well: **esptool**
- Mac OS terminal command to read entire flash memory:  

```
/Users/markkukellomaki/Library/Arduino15/packages/esp32/tools/esptool_py/4.9.dev3/esptool --chip esp32 --port /dev/cu.usbserial-14210 --baud 460800 read_flash 0x00000 0x400000 camera_binary.bin
```
- The flash file camera\_binary.bin is available in Moodle

## Microcontroller flash memory contents

- Let us study the contents of the binary file using hex editor. Options:
  - Online: <https://hexed.it/>
  - Visual Studio Code: Hex Editor extension
  - Mac OS:



- Install one of the tools and start exploring

## Hex editing

- This is how it looks like in Hex Fiend:

```
camera_binary.bin
Hex Text Find
Replace
[Replace All] [Replace] [Replace & Find] [Previous] [Next]
64692 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
64728 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
64764 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
64800 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
64836 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
64872 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
64908 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
64944 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
64980 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65016 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65052 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65088 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65124 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65160 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65196 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65232 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65268 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65304 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65340 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65376 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65412 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65448 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65484 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
65520 FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF E906022F AC290840 E0000000 00000000 00FFFFFF00
65556 00000001 2000403F 44F10200 3254CDAB 00000000 00000000 00000000 62623736 63623100
65592 00000000 00000000 00000000 00000000 00000000 00000000 61726475 696E6F2D 6C69622D
65628 6275696C 64657200 00000000 00000000 00000000 30363A31 343A3039 00000000 00000000
65664 40617220 32382032 30323500 00000000 76352E34 2E312D031 2D673266 37646364 38363261
65700 2D646972 74790000 00000000 C565C72C A271F640 50AEFA9C CC87792E E85BAF30 7758CEFC
65736 F6E5B989 1B4C021C 00006300 10000000 00000000 00000000 00000000 00000000 00000000
..... /.) @. ..
@?D. 2T.. bb76cb1
arduino-lib-
builder 06:14:09
Mar 28 2025 v5.4.1-1-g2f7dcd862a
-dirty .e.,.q.@P.....y...[.0w[...
.... L c
Signed Int le, dec (select some data)
65004 out of 4194304 bytes
```

# Hex editing

- This is how it looks like in HExEd.it website (remember to check encoding settings if you do not have Nordic language settings)

The screenshot shows the HExEd.it web application interface. On the left, there is a sidebar with file information and a 'Data Inspector (Little-endian)' panel. The main area displays a hex editor with a hex dump and its corresponding ASCII representation. The hex dump shows a series of 'FF' bytes, followed by some ASCII characters like 'e.../|..@' and 'P.....v.5.4.1-g2f7dcd862a-dirty.....Mar 28 2025 06:29:33.....'. The ASCII representation shows a stack trace error message: 'Assert failed in %s, %s: %d (%s)...abort() was called at PC 0x%08x...boot.E (%Lu) %s: load partition table error!.. is not bootable.E (%Lu) %s: Factory app partition...E (%Lu) %s: Factory test app partition...E (%Lu) %s: OTA app partition slot %d%\$.DROM.E (%Lu) %s: Image contains multiple %s segments. Only the last one will be mapped...IROM.E (%Lu) %s: Error in write\_otadata oper'.

## Hex editing

- Open the .bin file in hex editor
- Browse through and check which parts are human readable
  - General observations? What can you find?
- Search for the potential SSIDs you saw in Wifi listing
  - Use Find function of the tool
  - Which one do you find?
- Check all instances of the found SSID: is there any string that follows the SSID string at the same distance in memory? Does it look like a password?
  - Identify the string and try to connect to the Wifi router

# Hex editing

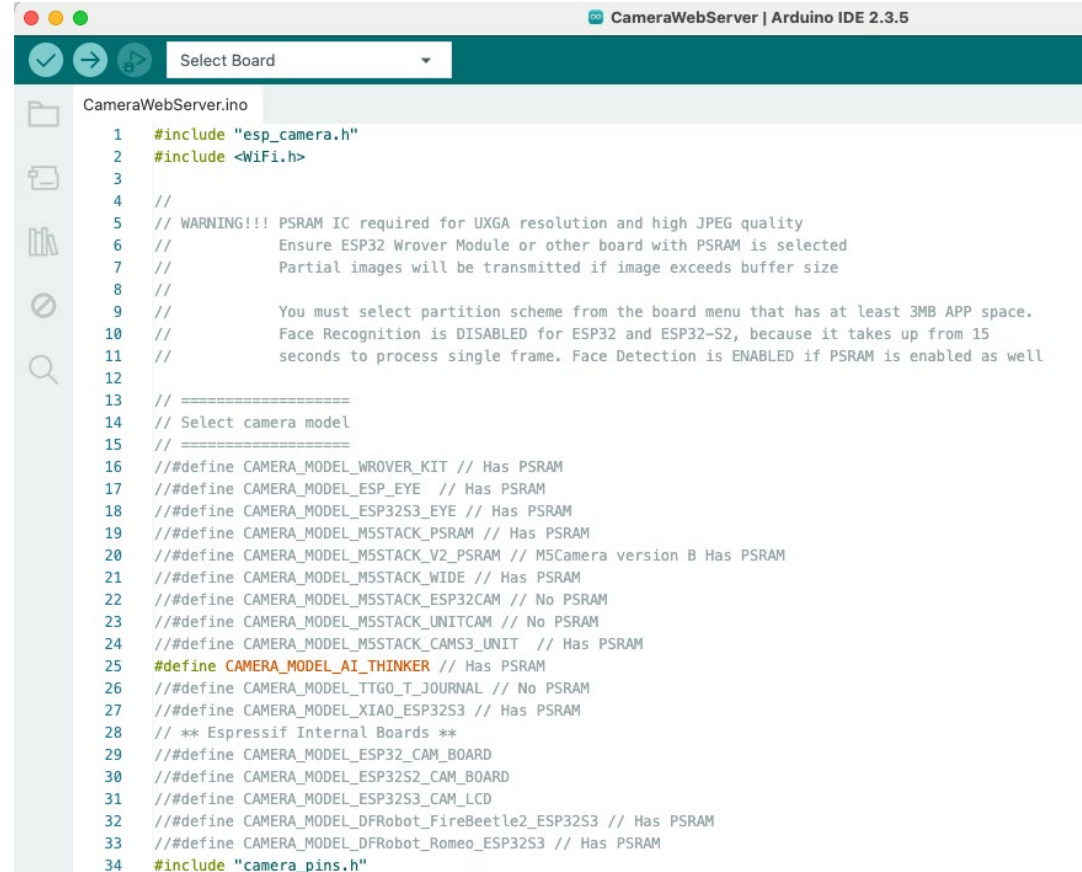
```

00009858 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00009870 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .H. <.z~
00009888 63 61 6C 5F 64 61 74 61 00 00 00 00 00 00 00 00 70 07 00 00 01 00 FF FF cal_data.....p.....
000098A0 03 42 02 00 EE 52 24 6A 63 61 6C 5F 6D 61 63 00 00 00 00 00 00 00 00 00 00 .B..εR$jcal_mac.....
000098B8 06 00 FF FF 87 A8 77 B9 F8 B3 B7 32 B5 A0 FF FF FF FF FF FF FF FF FF FF .. ç;w||°|η2|á
000098D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 03 48 01 FF 60 8C 90 C6 .H. `iÉf
000098E8 63 61 6C 5F 6D 61 63 00 00 00 00 00 00 00 00 06 00 00 00 01 00 FF FF cal_mac.....
00009900 03 04 01 FF 99 D7 F2 F9 63 61 6C 5F 76 65 72 73 69 6F 6E 00 00 00 00 00 00 ... ö|>:cal_version....
00009918 FC 12 00 00 FF FF FF FF 02 42 03 00 7E B4 3B B3 73 74 61 2E 73 73 69 64 ".... .B..~|;|sta.ssid
00009930 00 00 00 00 00 00 00 00 24 00 FF FF 92 F5 18 5F 08 00 00 00 52 61 73 50 .....$. Æ|.....RasP
00009948 77 6E 4F 53 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 wnOS.....
00009960 00 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
00009978 FF FF FF FF FF FF FF FF 02 48 01 FF 0C E6 3D 50 73 74 61 2E 73 73 69 64 .H. .μ=Psta.ssid
00009990 00 00 00 00 00 00 00 00 24 00 00 00 01 00 FF FF 02 42 04 00 DF 18 1F 94 .....$. .... .B..ö
000099A8 73 74 61 2E 70 73 77 64 00 00 00 00 00 00 00 00 41 00 FF FF 4A 57 32 B3 sta.pswd.....A. JW2|
000099C0 49 6E 35 33 63 75 72 33 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 In53cur3!.....
000099D8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000099F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF .....

```

## Check the source code

- Download the source code file CameraWbServer.ino from Moodle
- Inspect the source code: how are the network credentials handled?



```
CameraWebServer.ino
1  #include "esp_camera.h"
2  #include <WiFi.h>
3
4  //
5  // WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
6  //       Ensure ESP32 Wrover Module or other board with PSRAM is selected
7  //       Partial images will be transmitted if image exceeds buffer size
8  //
9  //       You must select partition scheme from the board menu that has at least 3MB APP space.
10 //       Face Recognition is DISABLED for ESP32 and ESP32-S2, because it takes up from 15
11 //       seconds to process single frame. Face Detection is ENABLED if PSRAM is enabled as well
12
13 // =====
14 // Select camera model
15 // =====
16 //#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
17 //#define CAMERA_MODEL_ESP_EYE // Has PSRAM
18 //#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
19 //#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
20 //#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
21 //#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
22 //#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
23 //#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
24 //#define CAMERA_MODEL_M5STACK_CAMS3_UNIT // Has PSRAM
25 #define CAMERA_MODEL_AI_THINKER // Has PSRAM
26 //#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
27 //#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
28 // ** Espressif Internal Boards **
29 //#define CAMERA_MODEL_ESP32_CAM_BOARD
30 //#define CAMERA_MODEL_ESP32S2_CAM_BOARD
31 //#define CAMERA_MODEL_ESP32S3_CAM_LCD
32 //#define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
33 //#define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
34 #include "camera_pins.h"
```

## Try to access website in the network

- After connecting to the wifi, check your IP address
  - Windows: ipconfig
  - Mac OS: if config

```
en0: flags=8863<UP, BROADCAST, SMART, RUNNING, SIMPLEX, MULTICAST> mtu 1500
options=6460<TS04, TS06, CHANNEL_IO, PARTIAL_CSUM, ZEROINVERT_CSUM>
ether 4a:4f:05:89:6c:81
inet6 fe80::849:cac4:554e:45a3%en0 prefixlen 64 secured scopeid 0xa
inet 192.168.99.158 netmask 0xfffff00 broadcast 192.168.99.255
nd6 options=201<PERFORMNUD, DAD>
media: autoselect
status: active
```

- Ping the hotspot: ping 192.168.99.1
- Try the website shown earlier for video streaming
- Try also this website: 192.192.99.13

## Denial of Service attack

- Let us try to bring down the website in address: 192.168.99.13
- Study Kali linux tool slowhttpstest: <https://www.kali.org/tools/slowhttpstest/>
- Try the tool and simultaneously try to access the website
- Did it work? How do you know?
  
- Repeat the DoS attack to the streaming service
  - Write down your observations

## Summary

- Network reconnaissance
  - Wifi network SSID must be chose carefully
    - Sometimes it may reveal too much
- Embedded device stores suprising information
  - Design of the device should minimize amount of connectors and potential attack ports
  - Software design for embedded devices must ensure credentials are not stored unencrypted ↔ hex editing tools
  - Confidentiality was not preserved in our example
- Network services need proper firewall for DoS attack prevention
  - Simple, automated tools can prevent availability of website
  - Availability was not preserved in our example

## Assignment

- Make a report on the phases of today's exercise
- Report commands you used with screenshots of the different phases
- Write what you learned about embedded device security and DoS attacks