**Ping/tracert**

1. Ping IP-adres 4.2.2.2 met flag –i 4.

1. Geef een screenshot van het resultaat.

Text

Description automatically generated

1. Verklaar het resultaat.

Er wordt gepinged naar 4.2.2.2 waarin 32 bytes aan data meegestuurd wordt. (4 \* 8 bytes?)

Vervolgens krijg ik 4 keer een echo terug waaruit blijkt dat er een connectie is met het ip-adres.

2. Traceroute www.avans.nl vanuit jouw thuisnetwerk.

a. Geef een screenshot van het resultaat.

Graphical user interface

Description automatically generated with medium confidence

b. Specifieer in jouw screenshot:

§ Alle apparaten binnen jouw eigen thuisnetwerk. (eerste 2?)

§ Een regel die laat zien wie jouw provider is. (Ziggo)

§ Een regel die laat zien wie de provider van Avans is. (Bloomreach)

3. Ping jouw WIFI modem/router. Zorg voor packet loss. Laat hier een screenshot van zien. Maak hiervoor gebruik van de –t of –n flags.

Text

Description automatically generated with medium confidence

**Tcpview**

1. Zorg met de programmeertaal van jouw keuze dat een proces luistert op TCP, poort 12345.

1. Geef een screenshot van Tcpview of netstat waarin dit te zien is.

Graphical user interface, application

Description automatically generated

1. Geef de code van jouw applicatie.

Server:

use std::{net::{TcpListener, TcpStream}};

use std::io::Read;

fn handle\_client(mut stream: TcpStream) {

    println!("New connection: {}", stream.peer\_addr().unwrap());

    let mut buffer = [0; 1024];

    while let Ok(n) = stream.read(&mut buffer) {

        if n == 0 {

            break;

        }

        println!("Received: {}", String::from\_utf8\_lossy(&buffer[..n]));

    }

}

pub fn main() -> std::io::Result<()> {

    println!("Starting listener...");

    let listener = TcpListener::bind("127.0.0.1:12345").expect("Failed to bind to address");

    for stream in listener.incoming() {

        let stream = stream.expect("Failed to accept connection");

        handle\_client(stream);

    }

    Ok(())

}

Client:

use std::io::prelude::\*;

use std::net::TcpStream;

pub fn main() -> std::io::Result<()> {

    println!("Connecting to server...");

    let mut stream = TcpStream::connect("127.0.0.1:12345").expect("Failed to connect to server");

    println!("Connected to server! {}", stream.peer\_addr().unwrap());

    stream.write(b"Hello World!")?;

    let mut buffer = [0; 1024];

    let n: usize = stream.read(&mut buffer)?;

    println!("Received: {}", String::from\_utf8\_lossy(&buffer[..n]));

    Ok(())

}

**Wireshark**

1. Maak met Wireshark een capture van een bezoek aan Osiris.

1. Geef een screenshot van de capture.

Text

Description automatically generated

1. Laat in jouw screenshot zien hoe te zien is dat de verbinding beveiligd is.

Er wordt gebruik gemaakt van TLS en er moet je door de osi-auth-server komen om daadwerkelijk bij osiris te eindigen.