# **Introduction**

## **Understanding how data moves across networks is crucial — and that’s where Wireshark comes in. As one of the most powerful network analysis tools out there, Wireshark allows you to capture, filter, and inspect data packets flowing through a network in real-time**

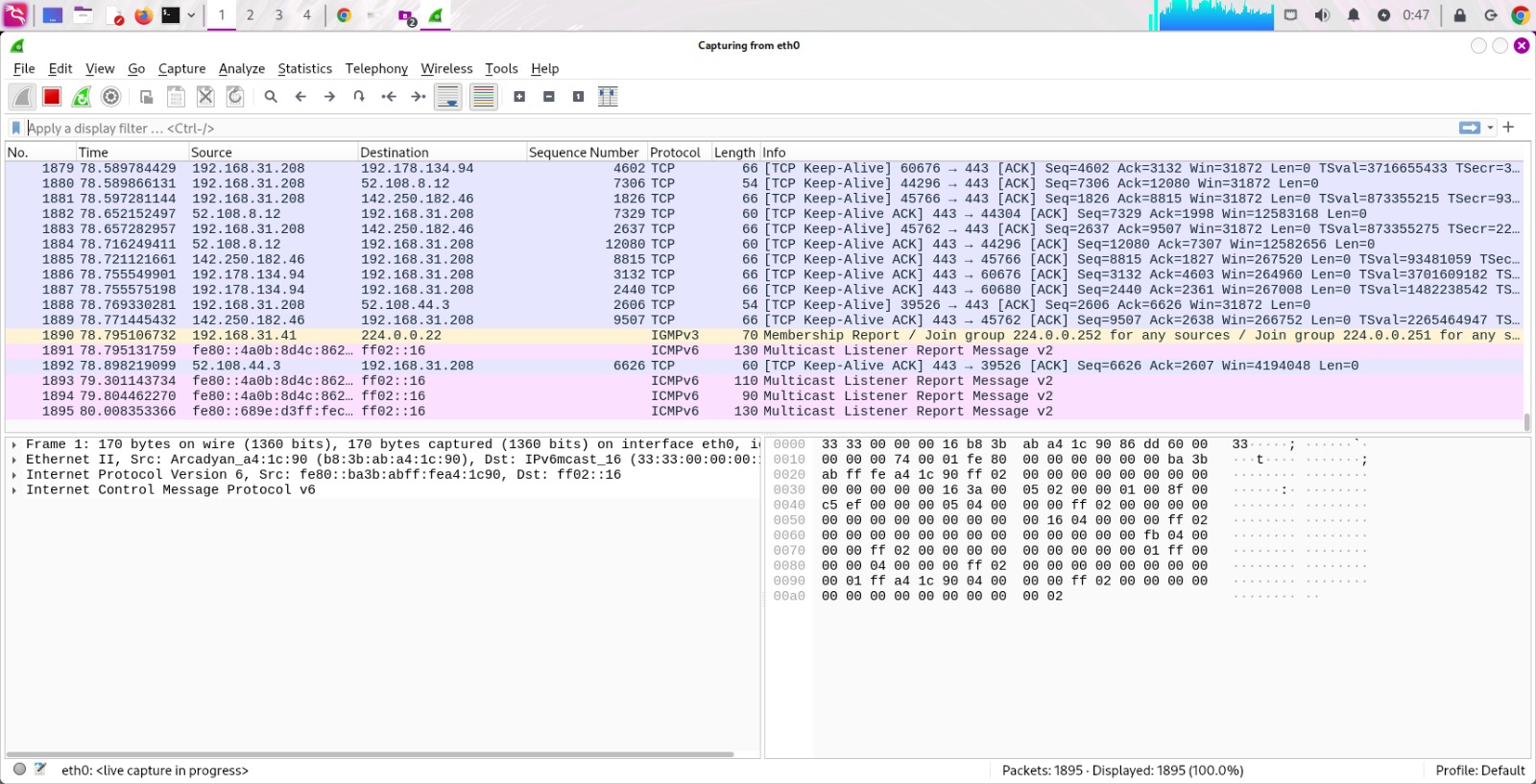
## **One of the most eye-opening features of Wireshark is its ability to reveal how vulnerable plaintext protocols are. In many cases, login credentials such as usernames and passwords can be transmitted without encryption, making them ripe for interception. With Wireshark, you can monitor network traffic, filter for specific login protocols, and even extract sensitive data if it’s being sent unprotected. It’s like having a magnifying glass over the web’s hidden layers.**

## **Step 1: Launch Wireshark**

## **Once installed, fire up Wireshark. You’ll be greeted by a list of available network interfaces these are the different ways your computer is connected to the network, such as Wi-Fi, Ethernet or even virtual network interfaces if you’re running a virtual machine. If you’re not sure which one to choose, look for the one showing the most activity (you’ll see little squiggly lines next to the network interface names). If you’re on Wi-Fi, select the Wi-Fi interface,or Ethernet if you’re using a wired connection**

## **Step 2: Start Capturing Network Traffic**

## **Now that you’ve selected your network interface, it’s time to start capturing traffic! As the capture begins, you’ll see a flood of packets coming in. Don’t panic! At this point, you’re basically watching all the traffic on your network in real time. But with Wireshark’s power comes a** *lot* **of noise, and this is where the fun begins — narrowing down the data to find what matters**



## **Step 3:**

# **Filtering Traffic for Login Protocols**

## **It’s time to sift through the noise and zoom in on the good stuff — login credentials. Most of the data you’ll capture is random traffic, but if you know what to look for, you can quickly find login attempts and other sensitive data. Wireshark makes this easy with its *filters*** **which allow you to focus on the protocols that matter for sniffing out passwords.**

## **. Focus on Plaintext Protocols**

## **Before we dive into the actual filtering process, let’s talk about *plaintext protocols* these are the key to finding passwords and credentials in Wireshark. Many older or less secure login methods (such as HTTP, FTP and Telnet) send data *Unencrypted* which means things like usernames and passwords are sent across the network in plain text. This makes them extremely vulnerable to anyone listening in (like Wireshark!).**

## **Here are the main protocols:**

* ***HTTP:*** Used for websites that don’t use HTTPS (the secure version of HTTP). This is where you’ll find login credentials from unencrypted web forms.
* ***FTP:*** Used for transferring files between computers, often sending login data unencrypted.
* ***Telnet:*** An old protocol for remote communication, which also sends credentials in plaintext.

## **3.2. Using Filters to Zero in on Login Traffic**

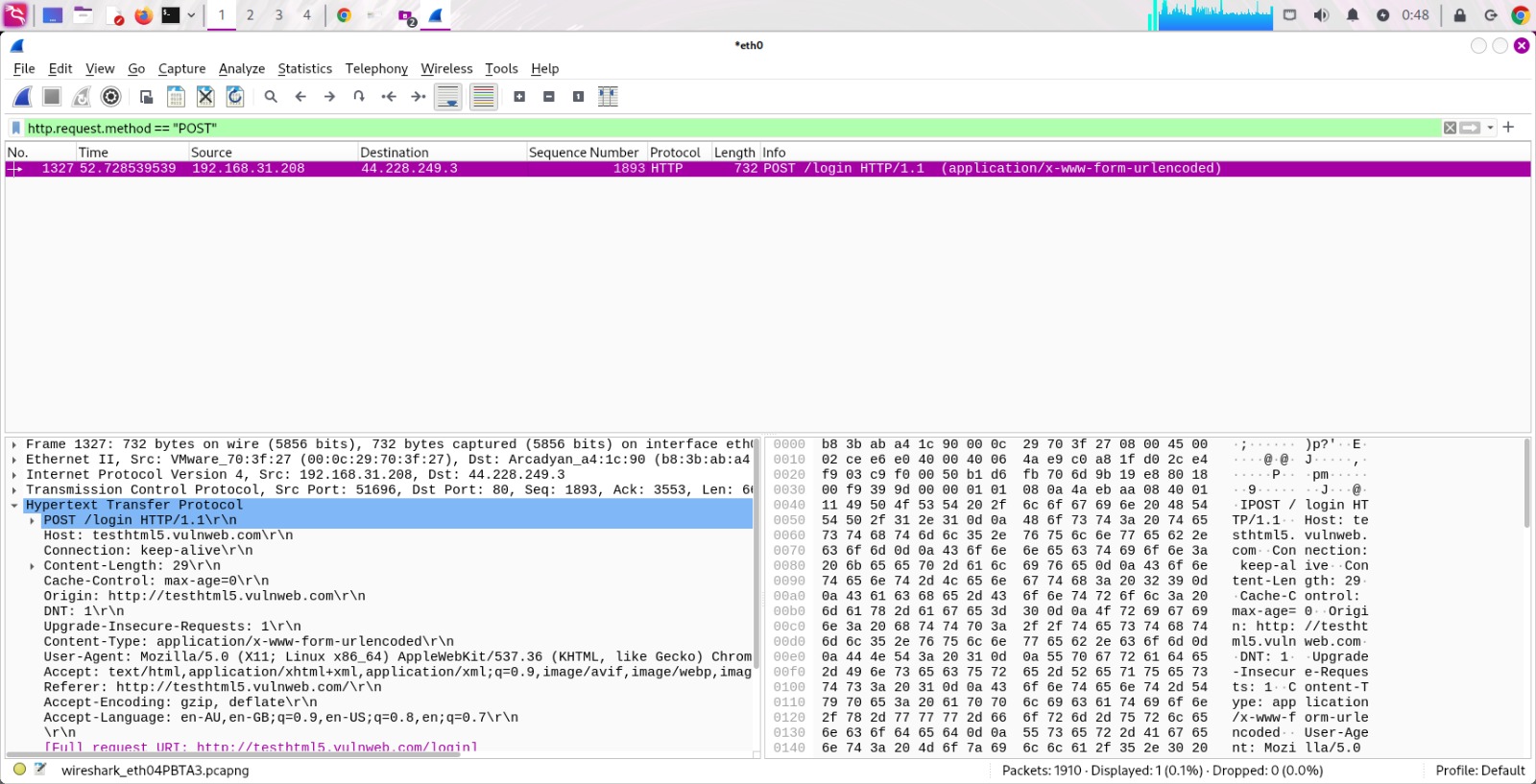
## **Wireshark can capture hundreds or even thousands of packets in just a few minutes, but you don’t want to manually sort through all of them. Instead, we’ll use *display filters* to narrow down the traffic to only what we need. Here are a few filters you’ll use to focus on login-related traffic:**

* ***HTTP Login Traffic:***Most websites send login information using HTTP POST requests (usually for submitting form data). To filter for those, use:

### http.request.method== "POST"

## **Applying the Filters**

## **To apply a filter, simply type one of the commands (like `http.request.method == “POST”`) into the *Filter Bar* at the top of the Wireshark window & press *Enter* Wireshark will immediately narrow down the packet list to only show the traffic that matches your filter. For example, if you’re looking for HTTP login traffic, typing `http.request.method == “POST”` will filter out everything else, leaving only packets where login forms might be submitted**

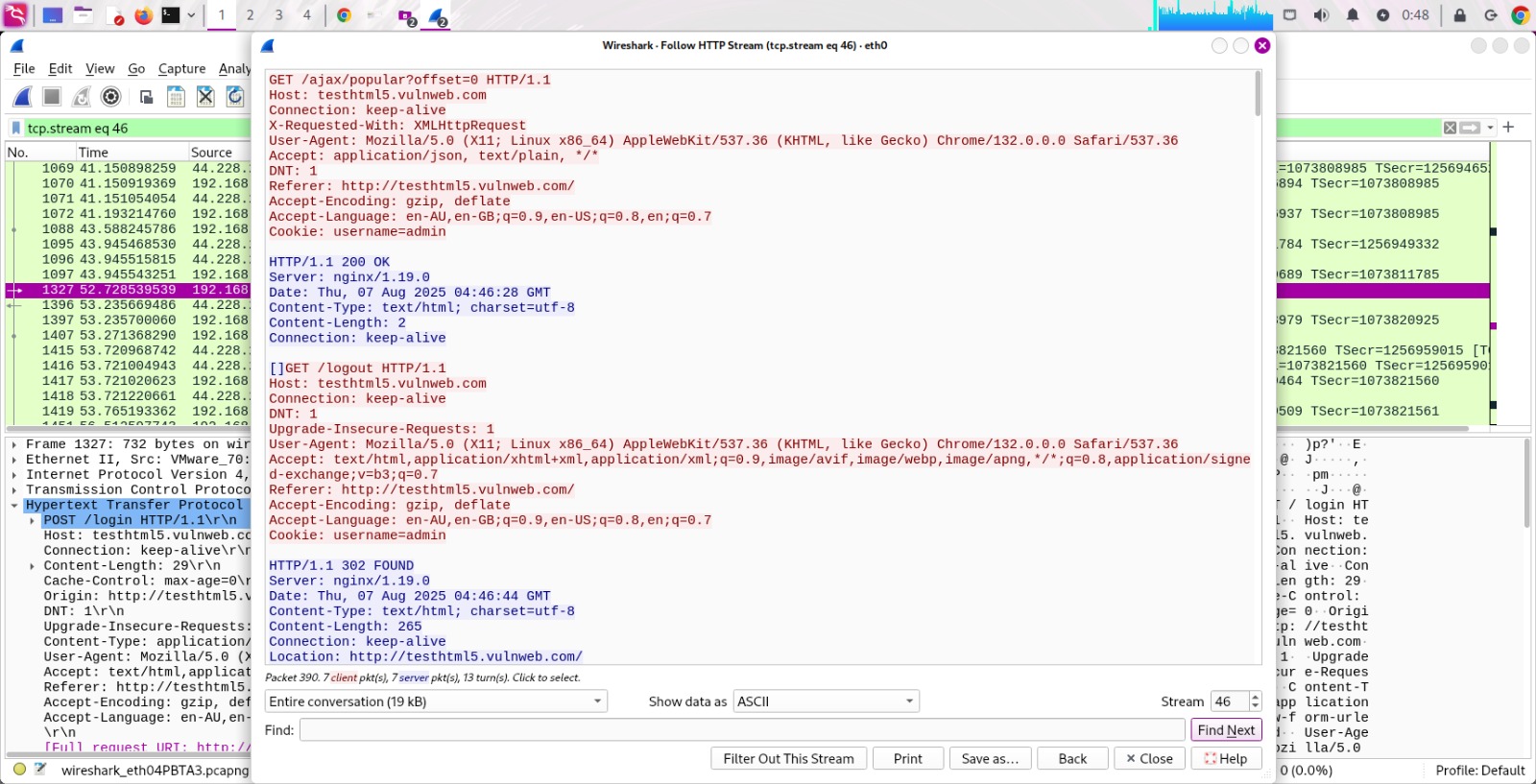


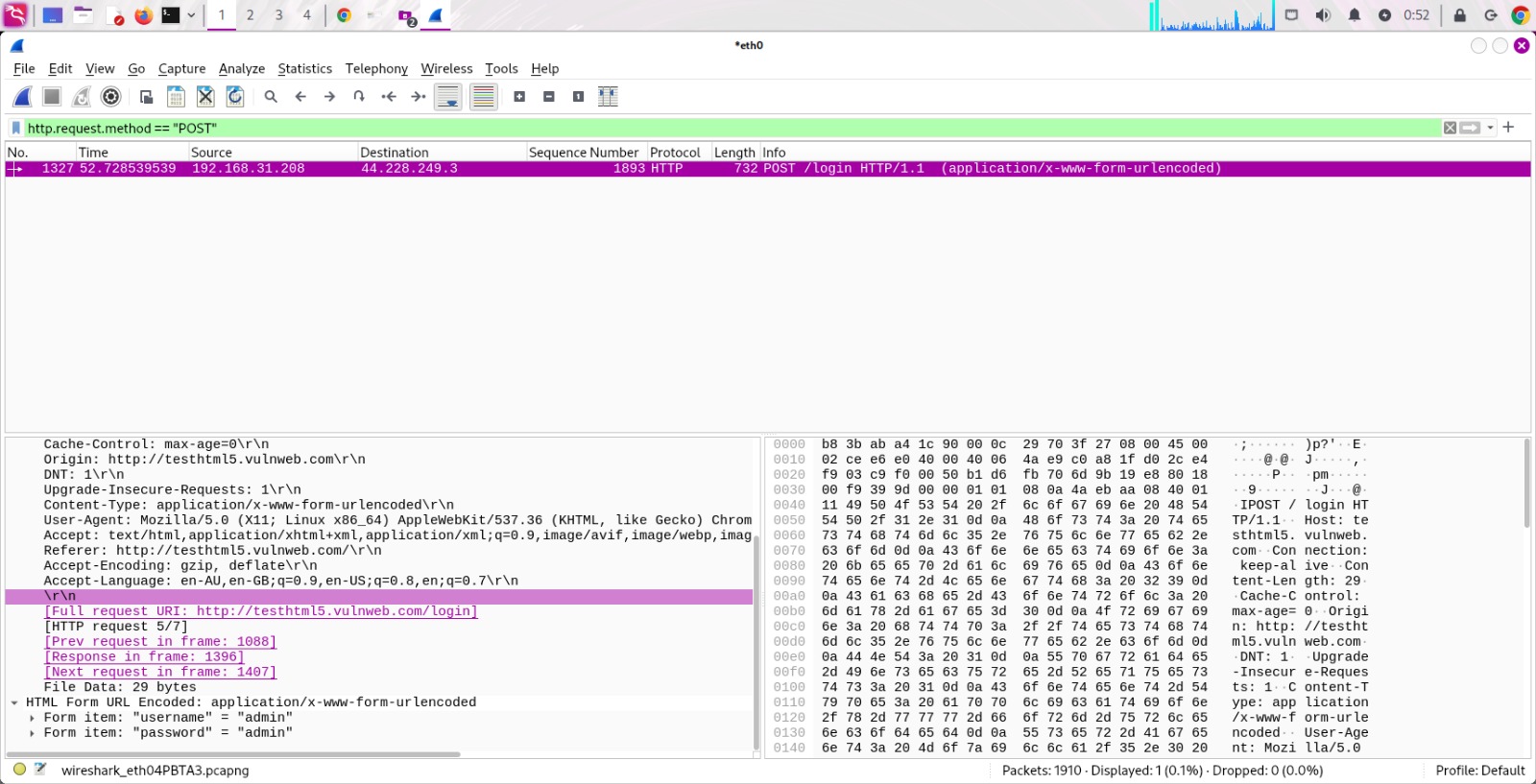
## **Finding Login Data in HTTP Packets**

## **If you’re inspecting an**

## ***HTTP***

## **packet, especially one that matches the POST filter (`http.request.method == “POST”`), you’ll often see form submission data. Expand the *HTTP layer* and scroll down until you see the section**





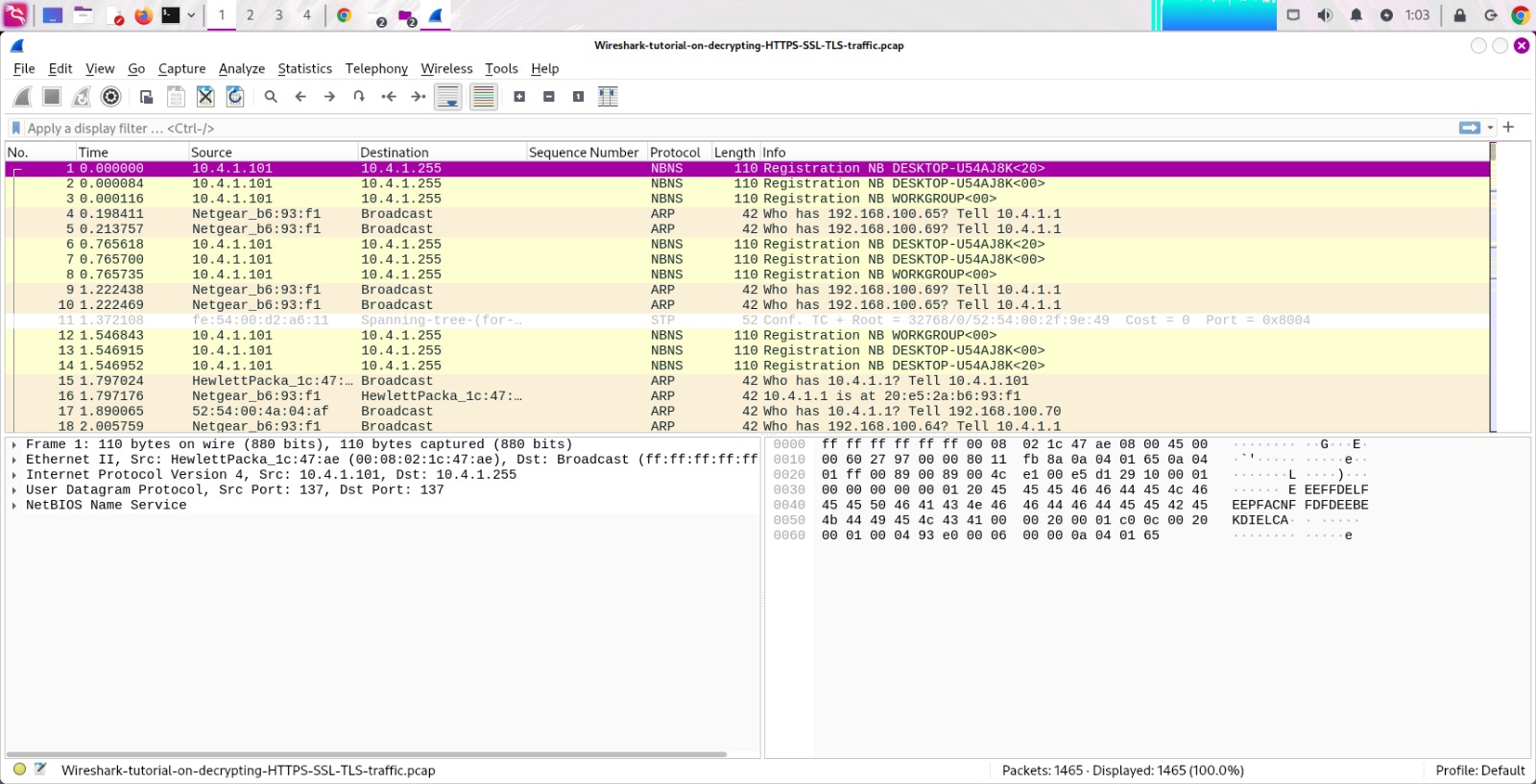
**Form Data:**

## **In older or insecure websites, usernames and passwords can be found in this section. Look for fields like `username=` and `password=` in plaintext.**

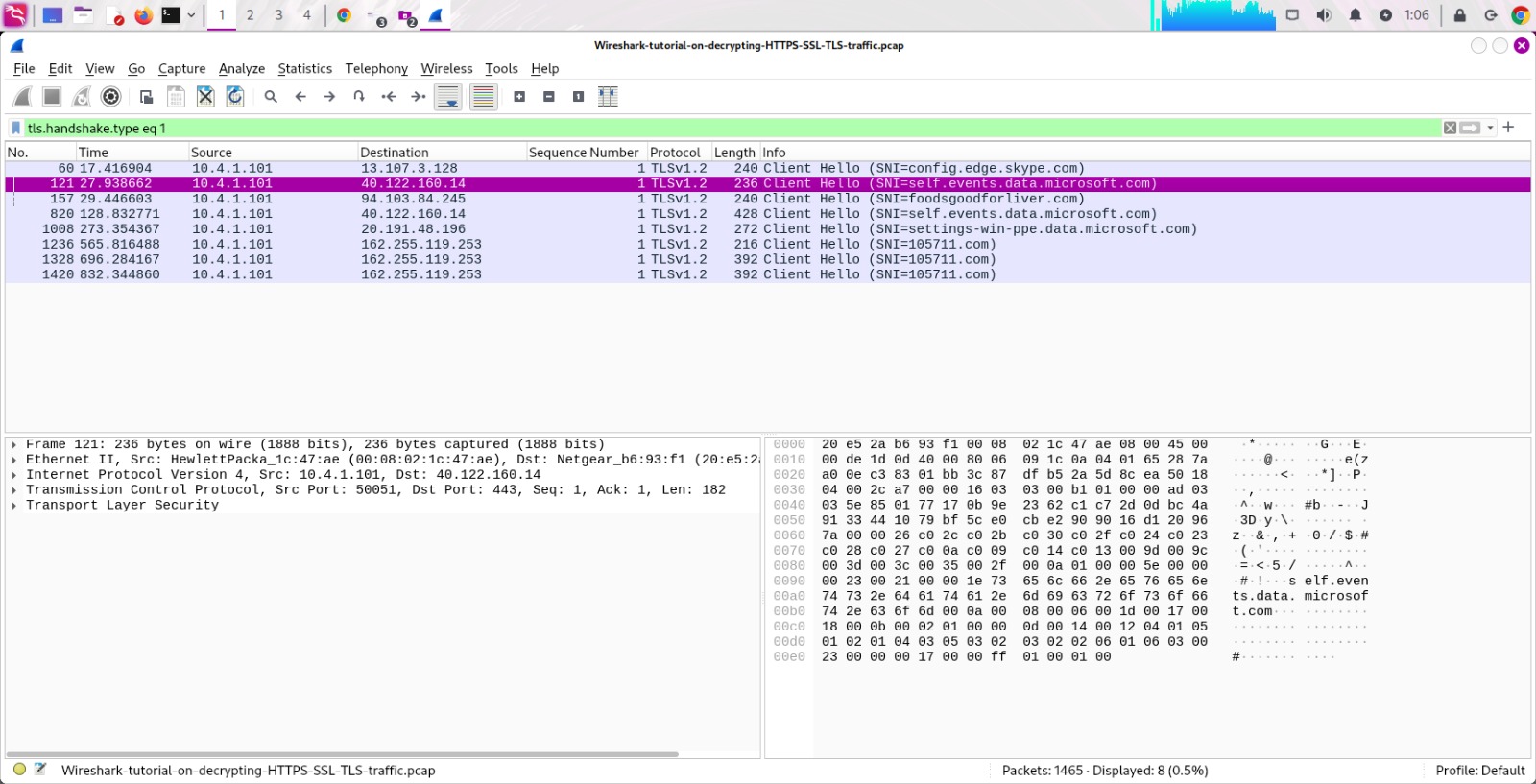
. Capture HTTPS packet using Wireshark and try to decrypt it.

**HTTPS Traffic Decryption**

## ***HTTPS traffic is encrypted by design, making it difficult to analyze directly in network captures. Wireshark provides tools to decrypt this traffic, given the appropriate encryption keys or configuration.***

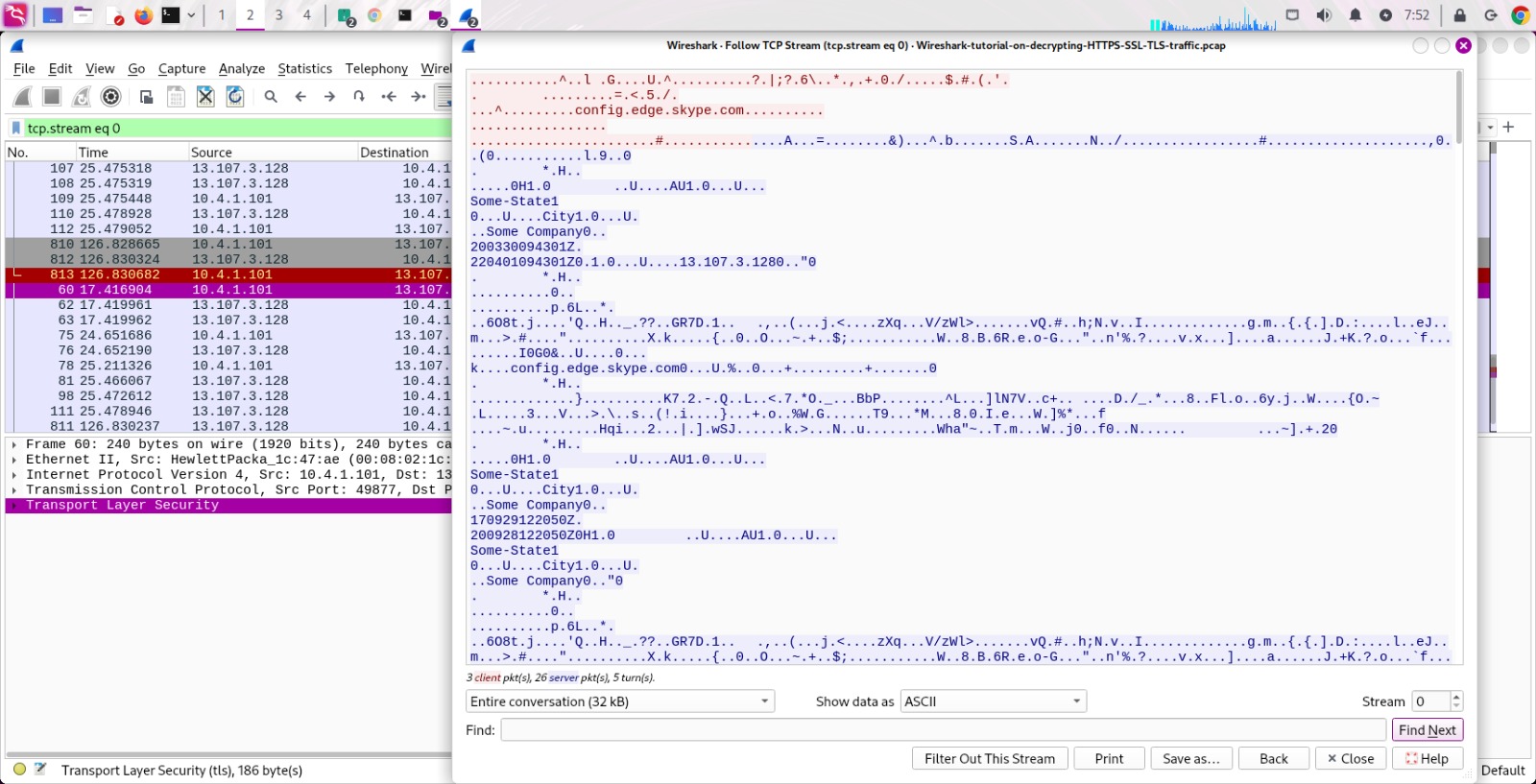


**Interpretation:**  
 This is a capture of **encrypted HTTPS/TLS traffic** where the filter isolates handshake initiation packets. It reveals which domains the local machine attempted to connect to, even without decrypting the traffic.

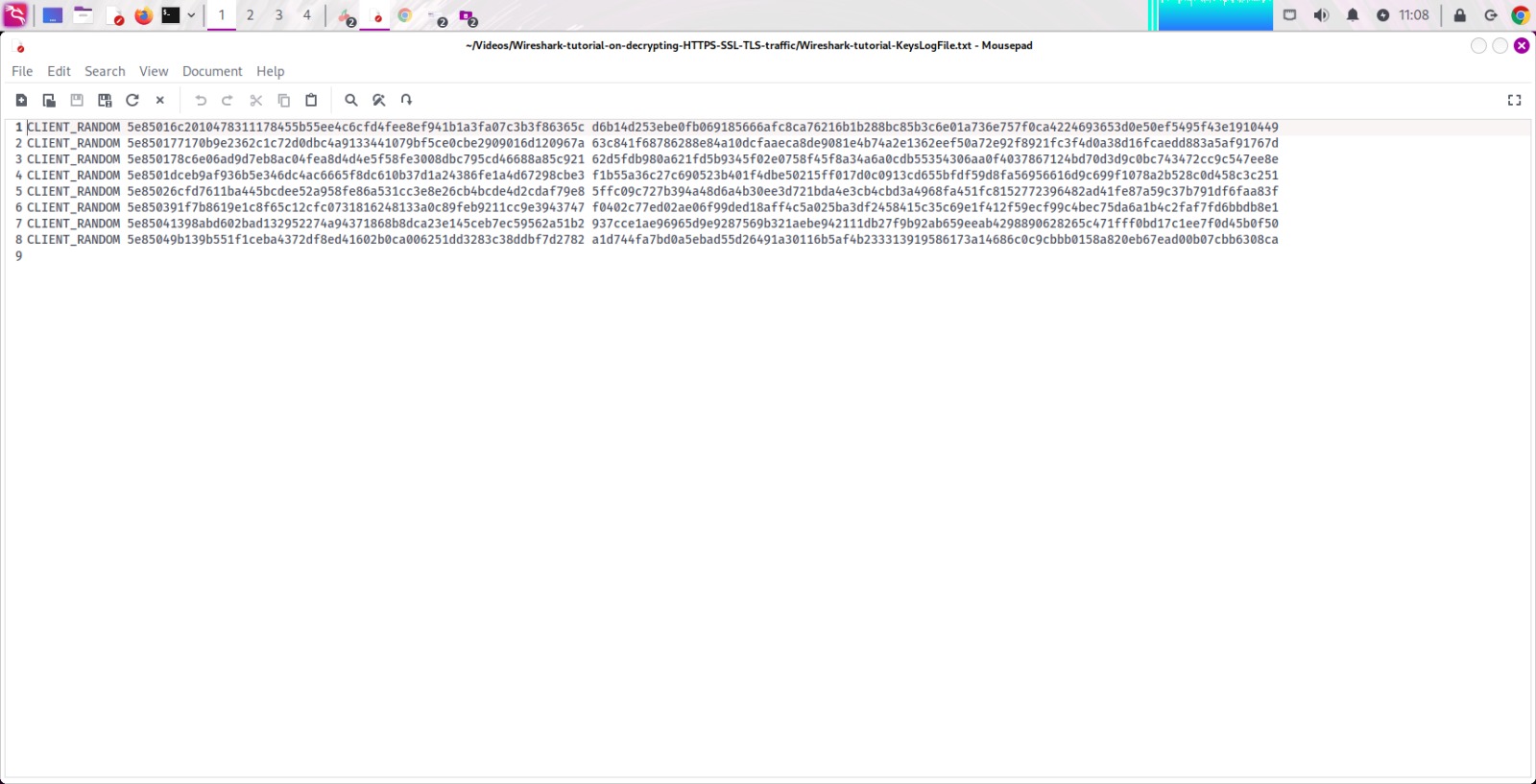


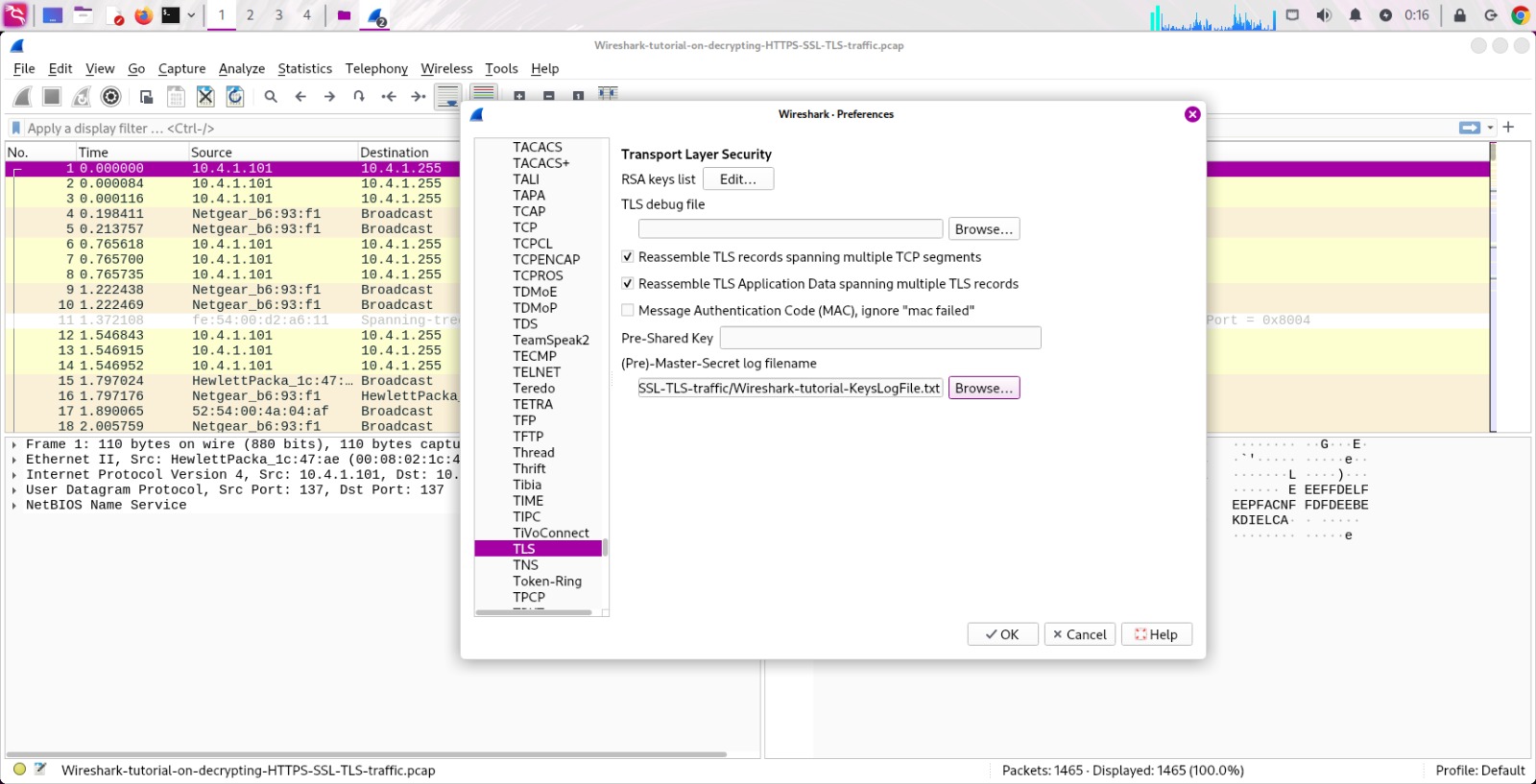
* **Filter Applied:** tls.handshake.type eq 1 – This shows only **TLS Client Hello** handshake packets.
* **Protocol:** All captured packets are **TLSv1.2**.
* **Source IP:** 10.4.1.101 (local machine) is initiating secure

This is a **TLS handshake and session** between a local client and the Skype configuration server. Although the traffic is encrypted, **the domain name and some certificate details are visible**, which can be useful in network forensics to identify the service and potential endpoints without decrypting the payload.



# **Loading the Key Log File**





The **Wireshark "Follow TLS Stream"** view for **stream 1**, and in this case, some of the HTTPS traffic contents are actually visible because the capture appears to include decrypted data.

