WIRELEAK CHALLENGE WALKTHROUGH

WRITTEN BY:
TALEEN SKAFI

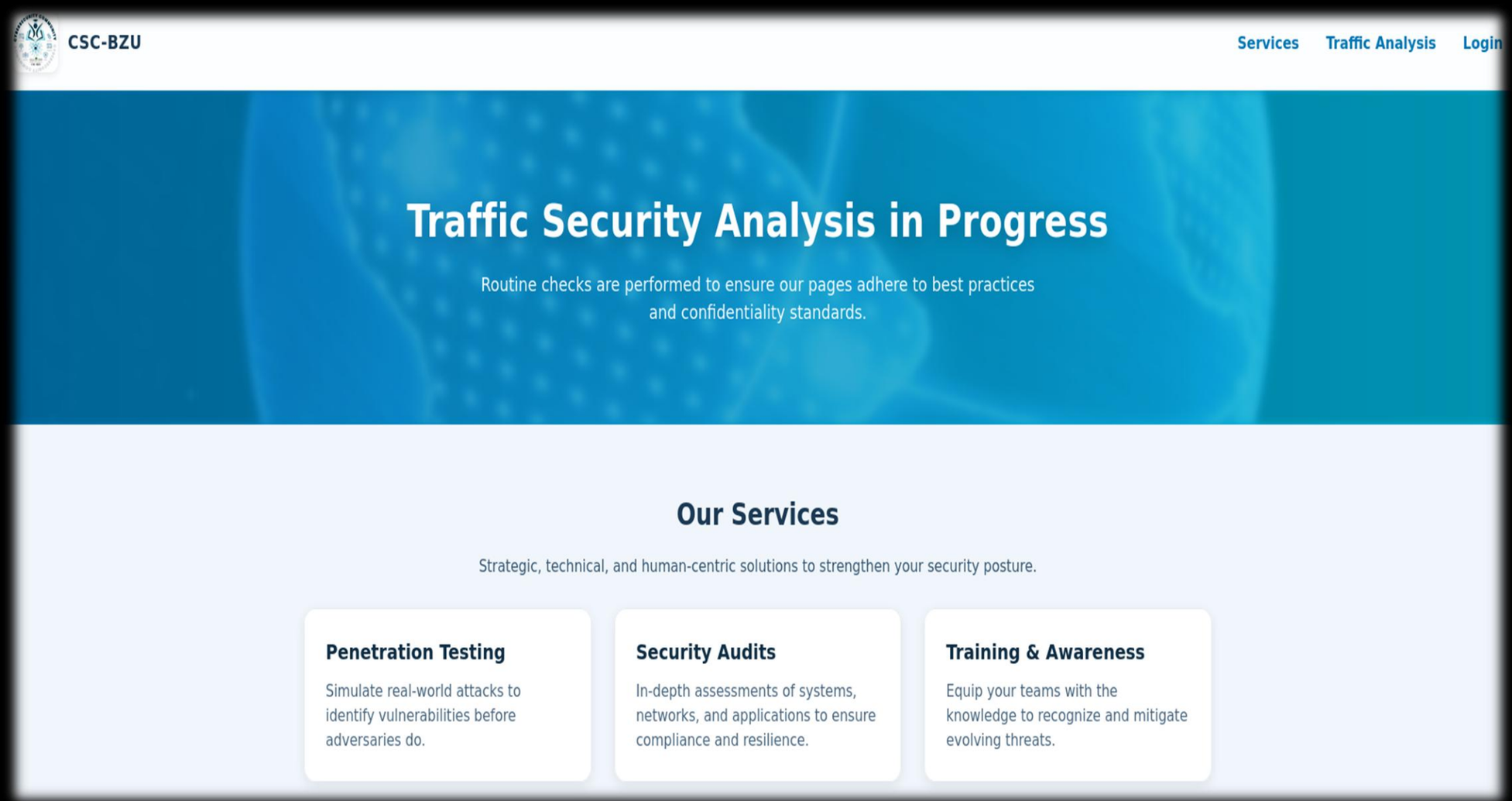
BZU - CSC

Welcome to **WireLeak**, a challenge where a web page uses plain HTTP to transmit data with **no encryption**. Because of this, sensitive information travels in cleartext and can be captured by anyone watching the network. Use a packet sniffer to uncover the leaked credentials and capture the flag.

When you open the web page, you'll notice it has multiple sections. One of them is titled "Traffic Analysis", and it includes a message that says:

"This site is currently undergoing testing to evaluate whether the data it sends and receives is transmitted securely.“

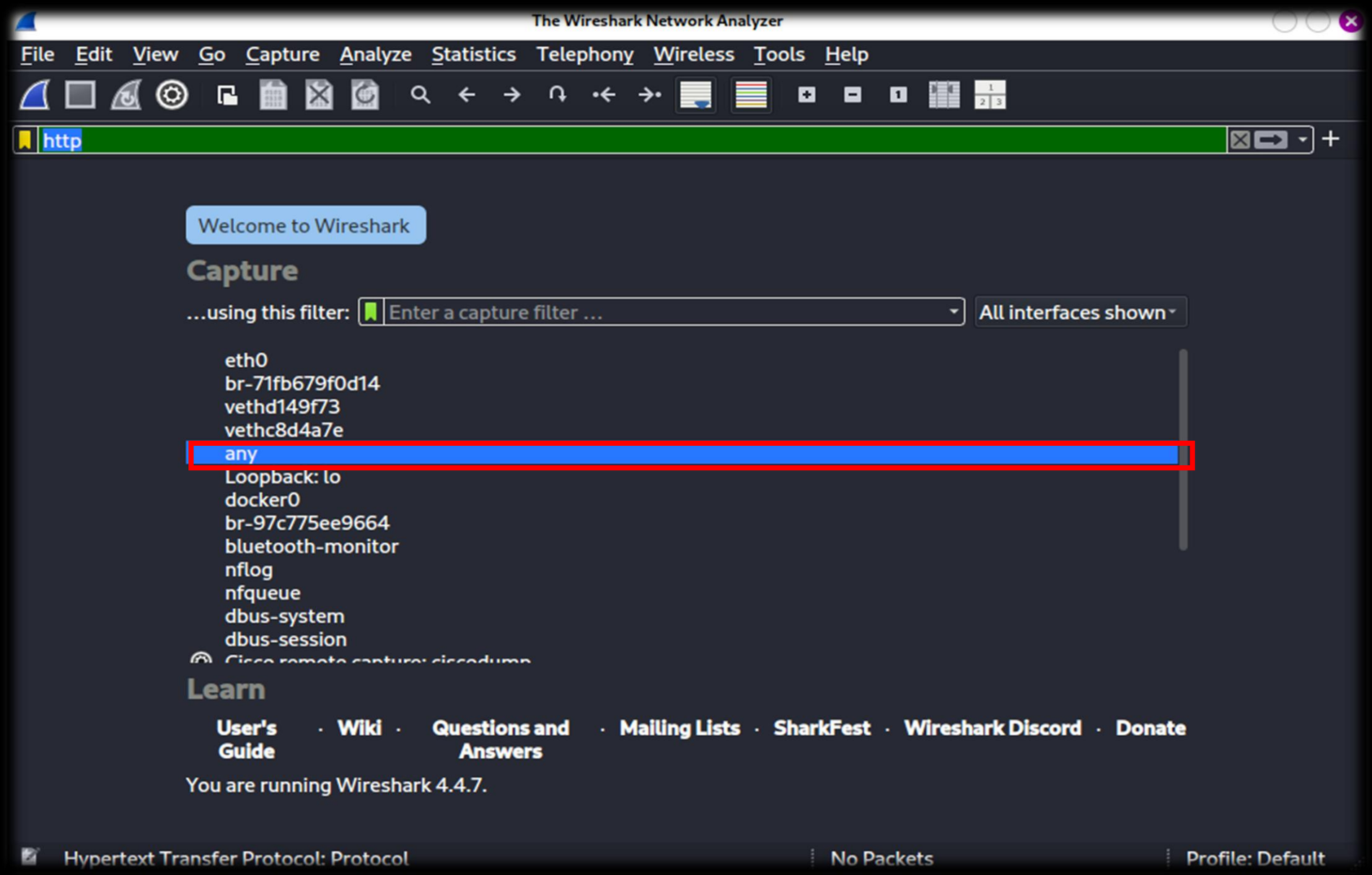
This observation hints that the site is using HTTP, not HTTPS , meaning the data is not encrypted. That could mean sensitive information might be leaking in plain text if you inspect the traffic closely.

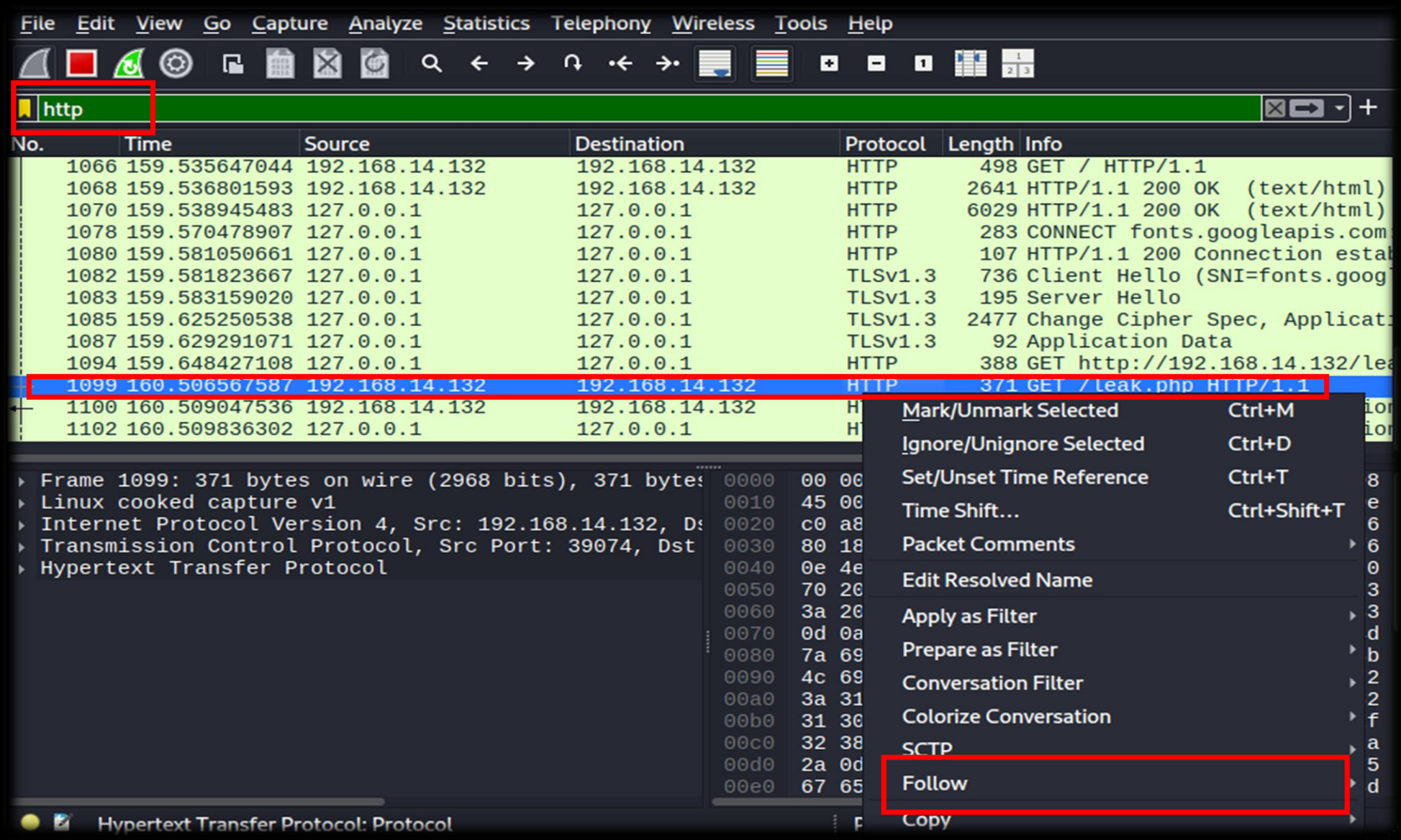


Since the website uses HTTP, the data it sends and receives is not encrypted. That means anyone on the same network can potentially intercept the traffic and read it in plain text.

In our case, we'll use Wireshark, a packet sniffer tool, to analyze the network traffic and see what kind of sensitive information might be leaking.

First, we'll open Wireshark and choose the network interface we want to monitor. In our case, we'll select "**any**", which is a special option that lets Wireshark capture traffic from all available network interfaces on the system. This is useful when you're not sure which specific interface the traffic is going through, or when multiple interfaces are active (like Ethernet, Wi-Fi, loopback, etc.).


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Welcome to Wireshark
Capture
...using this filter: Enter a capture filter ...
All interfaces shown
eth0
br-71fb679f0d14
vethd149f73
vethc8d4a7e
any
Loopback: lo
docker0
br-97c775ee9664
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate
You are running Wireshark 4.4.7.
Hypertext Transfer Protocol: Protocol
No Packets
Profile: Default



Once we refresh the website, Wireshark will start capturing all the network traffic related to that action. To narrow it down and focus only on the web traffic, we can type **http** into the Wireshark display filter bar. This will filter out everything except HTTP packets, making it easier to spot any unencrypted data being exchanged between the browser and the server.

As we inspect the filtered traffic, we'll notice that a file called **leak.php** is being requested by the web page. This is suspicious. If we right-click on that packet and choose “Follow → TCP Stream”, we’ll see something interesting, possibly some plaintext sensitive data being leaked in the response.

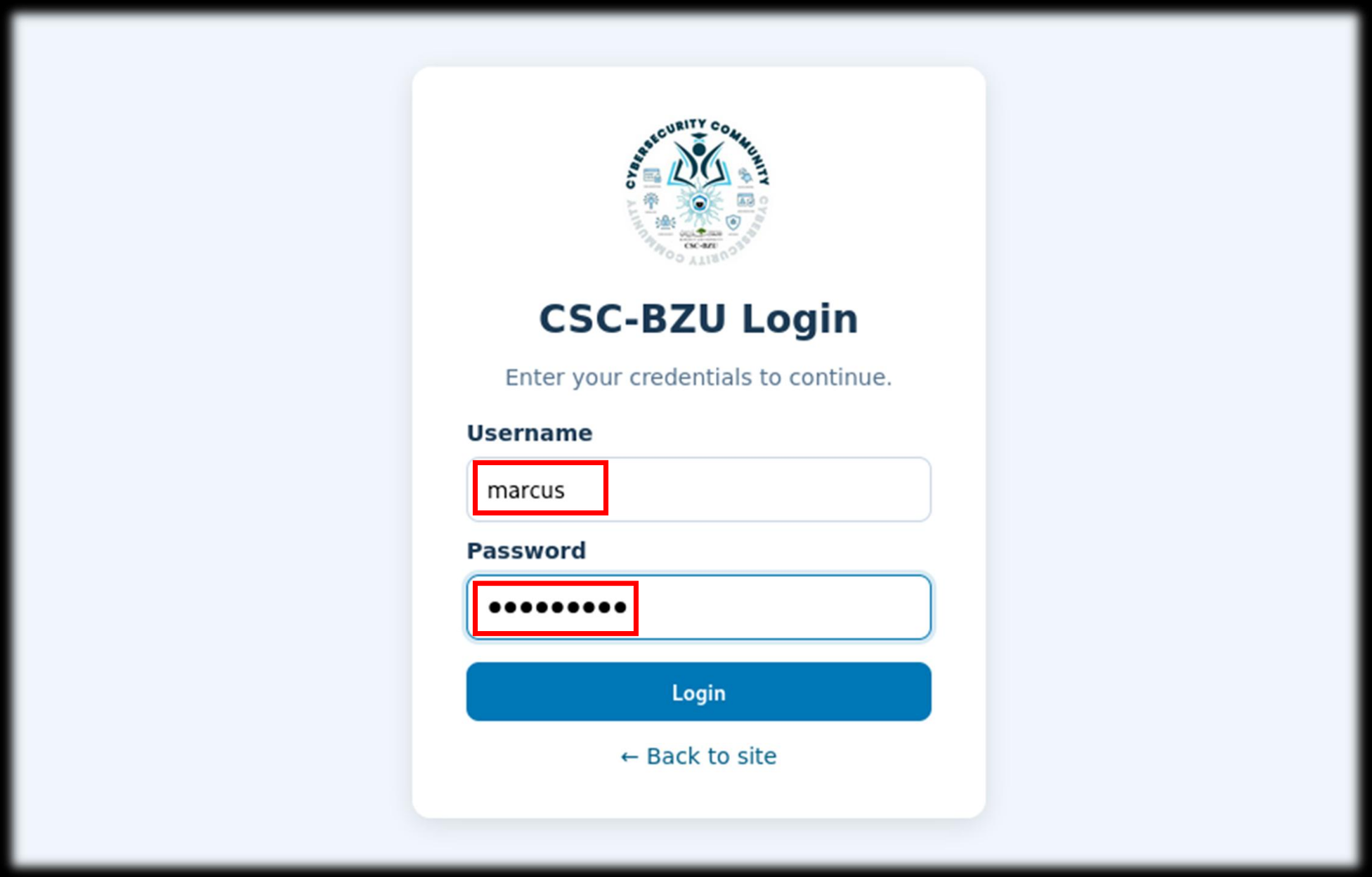
Wireshark · Follow TCP Stream (tcp.stream eq 13) · any

```
Host: 192.168.14.132
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/12
8.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://192.168.14.132/
X-Trigger: true
Connection: keep-alive
Priority: u=4


HTTP/1.1 200 OK
Date: Wed, 13 Aug 2025 20:34:22 GMT
Server: Apache/2.4.65 (Debian)
Content-Length: 59
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: application/javascript

// Internal auth log
// username=marcus password=m@rcu$123
```

Great! We've just uncovered some plaintext credentials for the user **Marcus**. Now, let's take these credentials and try logging into the website through the login page to check if they're actually valid.



Awesome, the credentials worked! We successfully logged in as **Marcus**, and as a result, we got our flag.