🔍 Wireshark Credential Capture – HTTP Login Analysis

📌 Project Overview

This project demonstrates how insecure login forms that transmit credentials over HTTP can be intercepted using Wireshark. Using the intentionally vulnerable site `http://testphp.vulnweb.com`, I captured login credentials during a simulated authentication process. The goal was to observe how unencrypted web traffic exposes sensitive data and to highlight the critical need for HTTPS.

🛠️ Tools Used

- **Wireshark** – Network traffic analysis
- **Browser** – For submitting login credentials
- **testphp.vulnweb.com** – Publicly available vulnerable web application by Acunetix

🎯 Objective

To analyze and capture plaintext login credentials transmitted via HTTP using Wireshark in a legal and ethical lab environment.

🧪 Methodology

1. **Environment Setup**
    - Connected to a monitored network interface (Wi-Fi/Ethernet).
    - Opened Wireshark and started packet capture on the active interface.

2. **Login Attempt**
    - Navigated to `http://testphp.vulnweb.com/login.php`.
    - Entered dummy credentials (e.g., `admin / 123456`) and submitted the form.
3. **Packet Analysis**
    - Stopped the capture after submission.

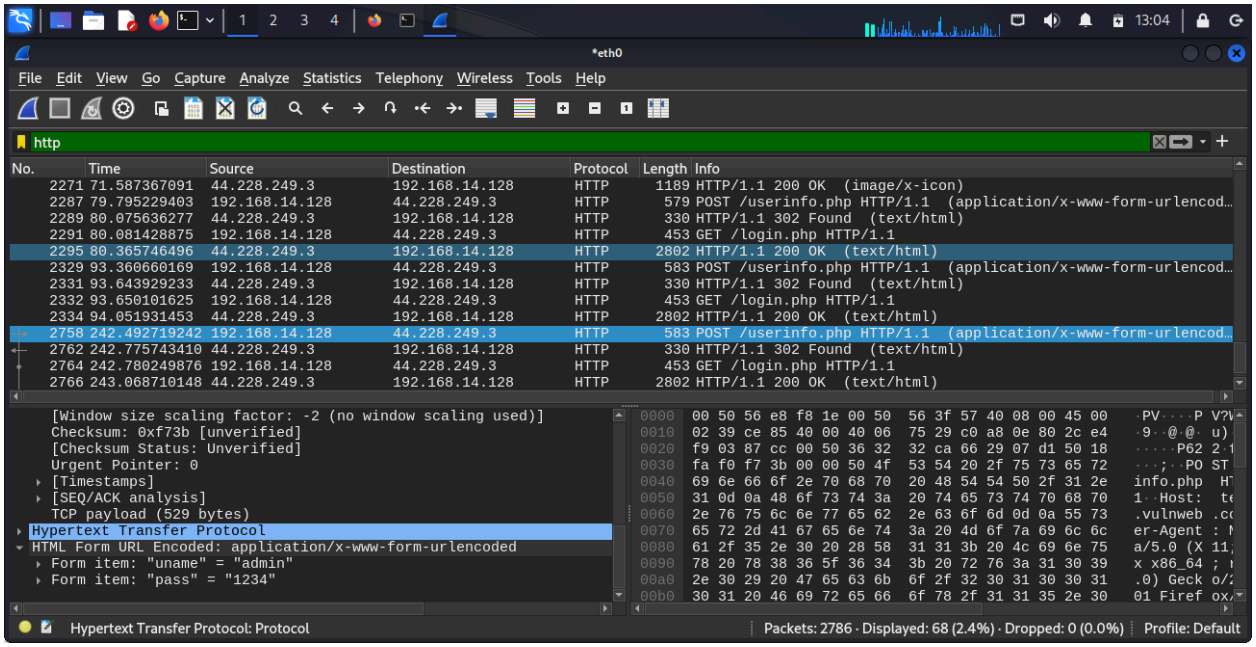
Used Wireshark filters to locate the HTTP POST request:

http.request.method == "POST"

Located the credentials in the payload:

username=admin&password=123456

📸 Screenshots & Evidence



🔍 Findings

- Credentials are transmitted **in plaintext** over HTTP.
- Anyone intercepting the network traffic can read them directly.
- No encryption or secure tokenization is used on this form.

🔐 Security Implications

- **Risk**: High – Sensitive data exposure.
- **Threats**: Man-in-the-middle attacks, session hijacking, credential theft.
- **Real-World Example**: Open Wi-Fi hotspots where attackers can sniff traffic using tools like Wireshark.

✅ Recommendations

- **Always use HTTPS** with valid SSL/TLS certificates to encrypt login forms.
- Implement **secure authentication practices** like hashing, multi-factor authentication (MFA), and token-based sessions.
- Enforce **HSTS (HTTP Strict Transport Security)** headers to prevent protocol downgrading.

📁 Files Included

- `testphp-traffic.pcap` – Captured network traffic (sanitized).
- `credentials-capture.png` – Screenshot of intercepted POST request.
- `README.md` – This report.

🧠 What I Learned

- How Wireshark can be used for real-time credential interception.
- The difference between secure (HTTPS) vs. insecure (HTTP) web traffic.
- The critical importance of encryption in web application security.

📚 References

- Wireshark Official Documentation
- OWASP Transport Layer Protection Cheat Sheet
- Acunetix Vulnerable Web App