Wireshark Network Capture

October 16, 2024

1 Wireshark Network Packet Capture

Owen Kroeger

ITT-305

Prof. Sluiter

1.0.1 1. Wireshark Network Packet Capture Demonstration

This assignment shows how unencrypted network traffic can be intercepted using Wireshark. We explore how sensitive information, such as usernames and passwords, can be exposed in plaintext over HTTP and how switching to HTTPS encryption protects this data during transmission.

1.0.2 2. Building the Login Application

We created a simple Spring Boot application with a login form to demonstrate how HTTP traffic can expose sensitive information. The key steps involved:

- **Creating a new Spring Boot project**: We set up the project with Spring Web, Thymeleaf, and DevTools dependencies.
- **Creating a `UserModel` class**: This class stores user credentials such as `username`, `password`, and `id`.
- **Creating a `UserController` class**: This handles GET and POST requests for displaying the login form and processing user login credentials.
- **Creating two HTML templates**:
  - `loginform.html`: Provides a form for users to input their login credentials.
  - `loginresult.html`: Displays the results of the login attempt, indicating success or failure.

1.0.3 3. Network Packet Sniffing with Wireshark

Wireshark is a network diagnostic tool used to capture network traffic. It can be used for both legitimate network analysis and malicious packet sniffing. We used Wireshark to demonstrate how HTTP traffic exposes sensitive data.

Steps:

- **Start Wireshark**: Select the network interface to monitor (in this case, the loopback interface, since we're running the web server locally).

- **Capture HTTP traffic**: After starting Wireshark, we logged in to the Spring Boot application. Wireshark captured the unencrypted HTTP request, which included the username and password in plain text.

1.0.4 4. Transitioning to Secure HTTPS Communication

To protect user credentials, we transitioned the application from HTTP to HTTPS using SSL/TLS encryption.

- **Generating an SSL Certificate**: Using the `keytool` command, we generated a self-signed SSL certificate and stored it in a keystore.
- **Configuring Spring Boot for HTTPS**: We updated the `application.properties` file to enable HTTPS by specifying the keystore and its password.
- **Running the application with HTTPS**: The application now runs on port `8443` using HTTPS, ensuring encrypted communication.

1.0.5 5. Capturing Encrypted Traffic with Wireshark

With HTTPS enabled, we captured the network traffic using Wireshark again.

- **TLS encryption**: This time, the captured traffic was encrypted using the TLS protocol, making it unreadable. Wireshark showed packets labeled as "TLS" instead of "HTTP," confirming that the data was encrypted and secure.

1.0.6 6. Understanding SSL/TLS and Certificate Authorities

SSL/TLS certificates are used to encrypt data during transmission, ensuring that sensitive information like passwords cannot be intercepted by third parties.

- **Self-signed certificates**: For this demo, we used a self-signed certificate, which is not trusted by browsers but is acceptable for local testing.
- **Certificate Authorities (CAs)**: In production, certificates should be issued by trusted CAs to ensure that browsers and clients trust the website.

1.0.7 7. Security Best Practices

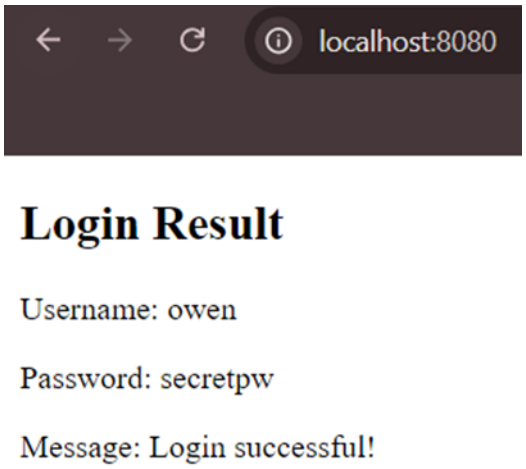
To ensure secure communication:

- Use strong, modern encryption standards like TLS 1.2 or higher.
- Always use HTTPS for web applications that handle sensitive information.
- In production, use SSL/TLS certificates issued by trusted CAs, rather than self-signed certificates.

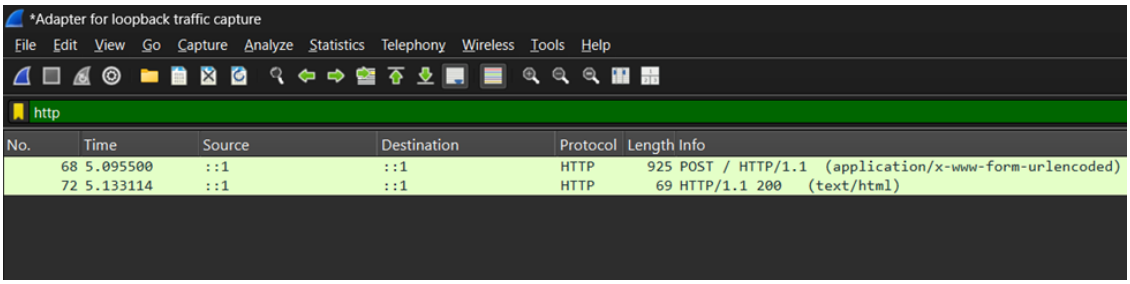
1.0.8 8. Common Security Protocols

Initials	Name	What It Is Used For
HTTP	Hypertext Transfer Protocol	Send hypertext pages for World Wide Web applications.
TCP	Transmission Control Protocol	Provides reliable, ordered, and error-checked delivery of data over IP networks.

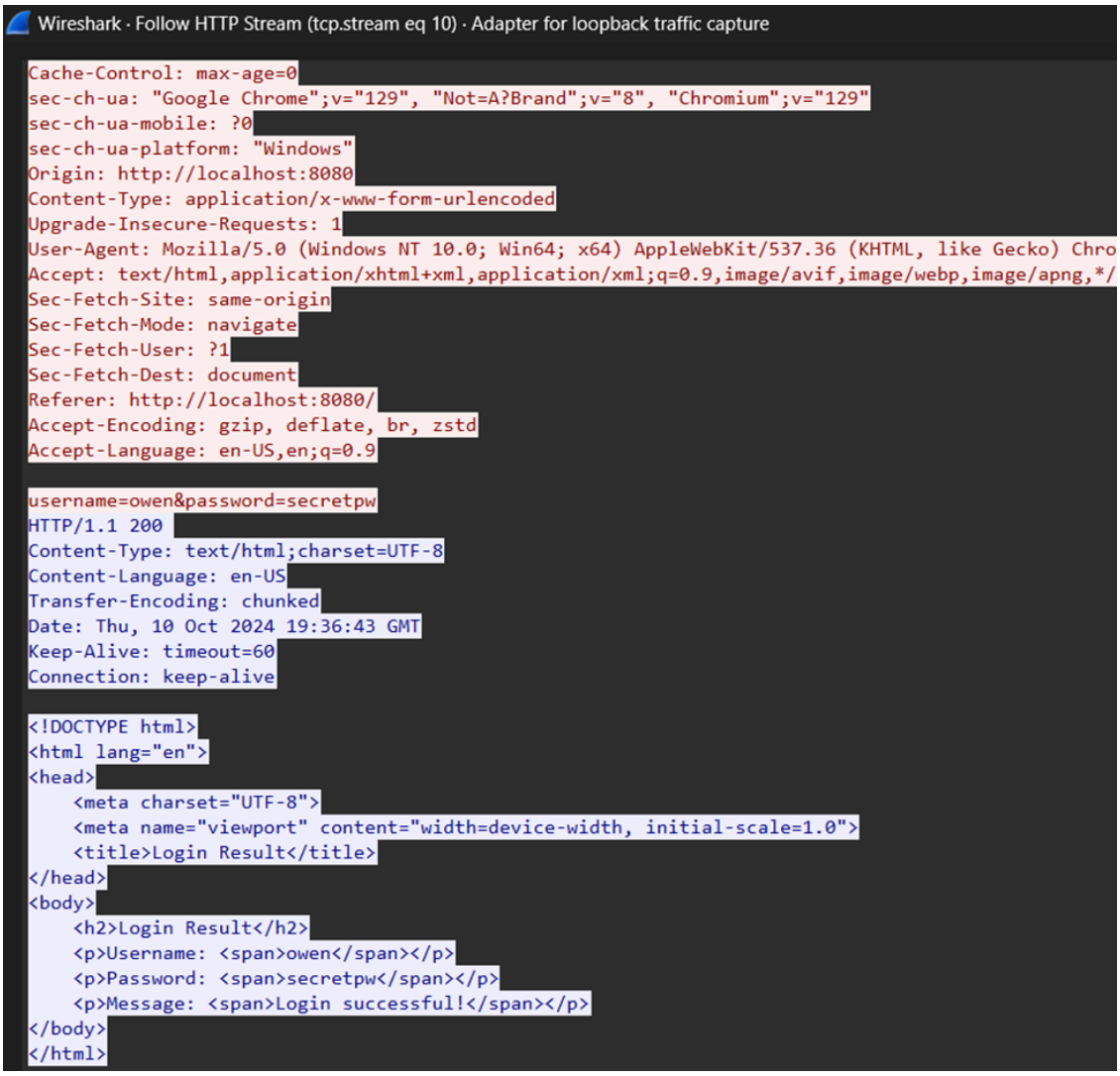
Initials	Name	What It Is Used For
SNMP	Simple Network Management Protocol	Used for network management and monitoring.
FTP	File Transfer Protocol	Transfer files between client and server over a network.
SMTP	Simple Mail Transfer Protocol	Send emails between mail servers and clients.
IMAP	Internet Message Access Protocol	Access and manage emails on a remote server.
POP3	Post Office Protocol 3	Retrieve emails from a mail server.
DNS	Domain Name System	Resolves domain names to IP addresses.
SSH	Secure Shell	Provides secure remote access to systems over an unsecured network.
UDP	User Datagram Protocol	Sends data without ensuring reliable delivery, often used in real-time services.
RDP	Remote Desktop Protocol	Provides remote access to Windows desktops and applications.
VoIP	Voice over Internet Protocol	Delivers voice communications and multimedia sessions over IP networks.
DHCP	Dynamic Host Configuration Protocol	Assigns IP addresses to devices on a network automatically.
LDAP	Lightweight Directory Access Protocol	Manages and accesses distributed directory information services.
TLS	Transport Layer Security	Provides encryption for data transmitted over networks to ensure security.



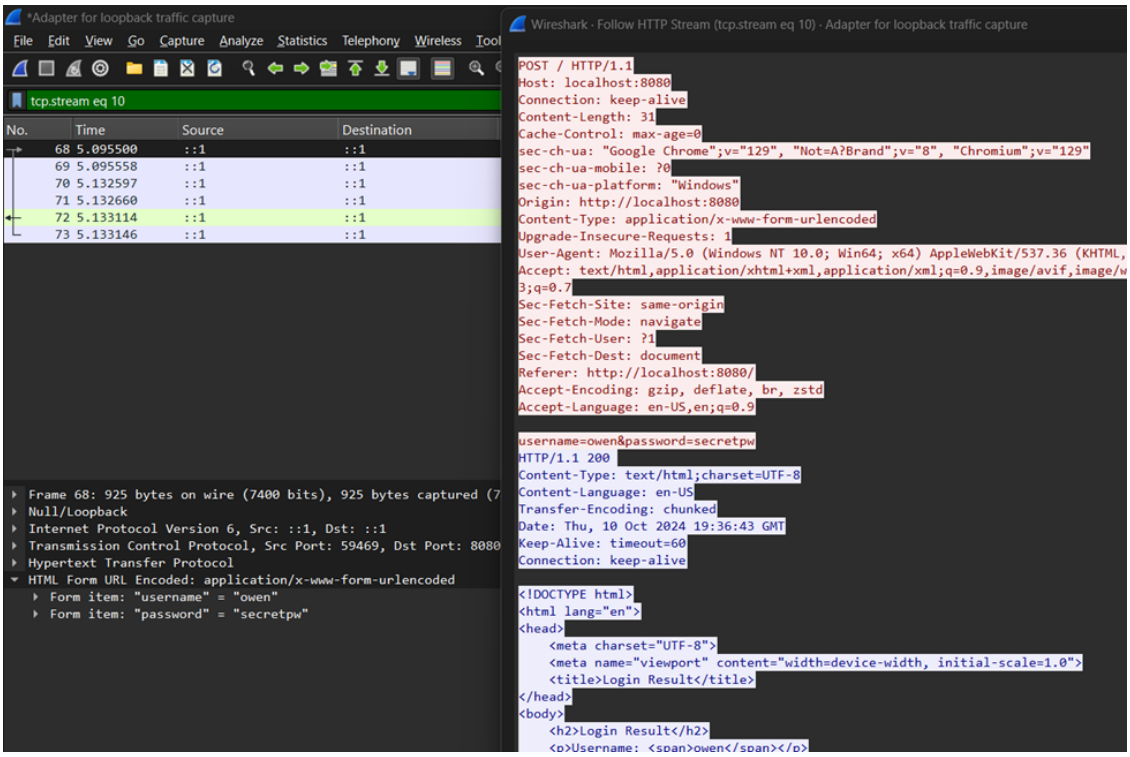
Caption: ↑ This first picture shows my initial login page set up and running on local host with a successful login.



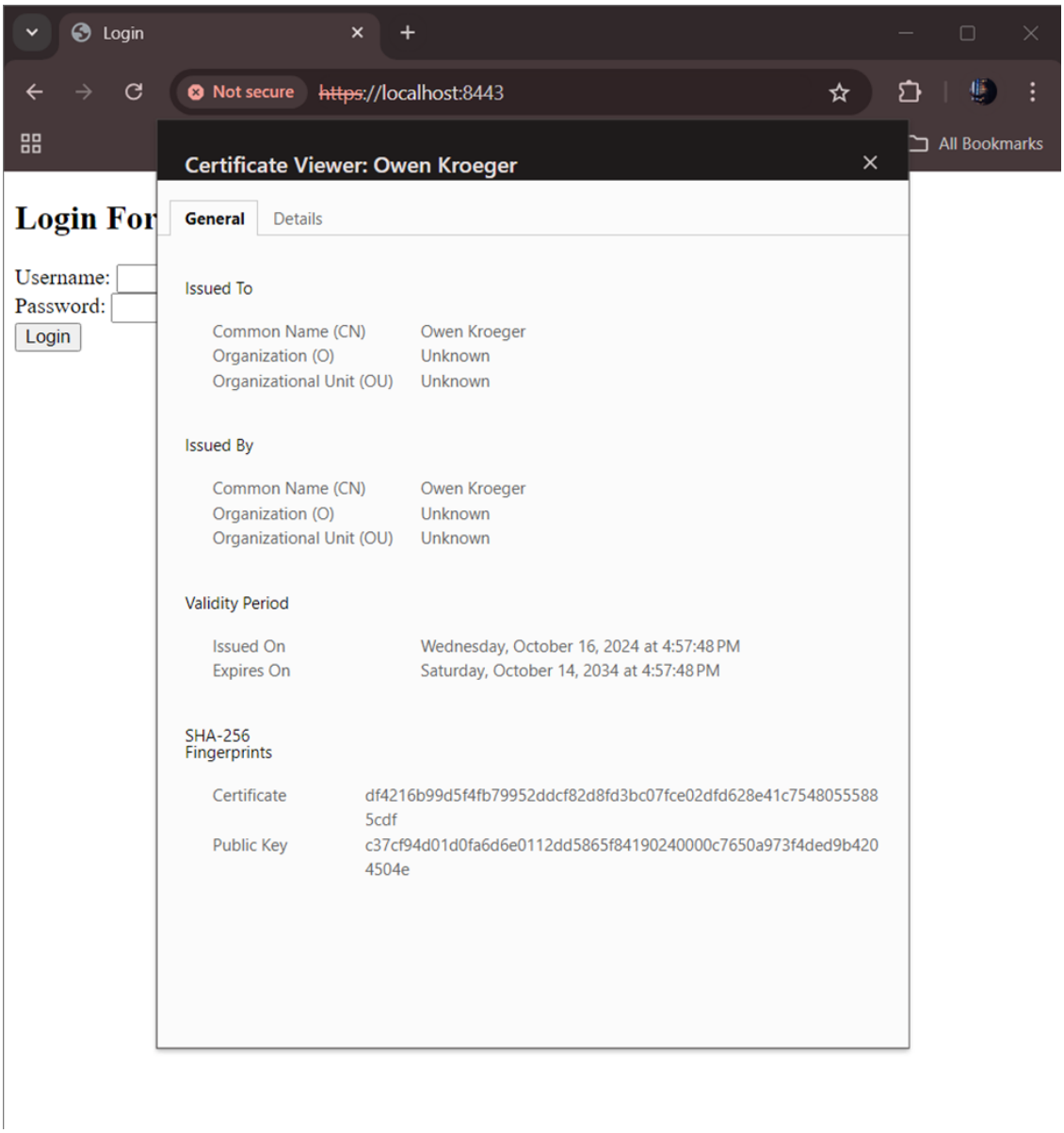
Caption: ↑ This shows wireshark capturing the unencrypted HTTP request.



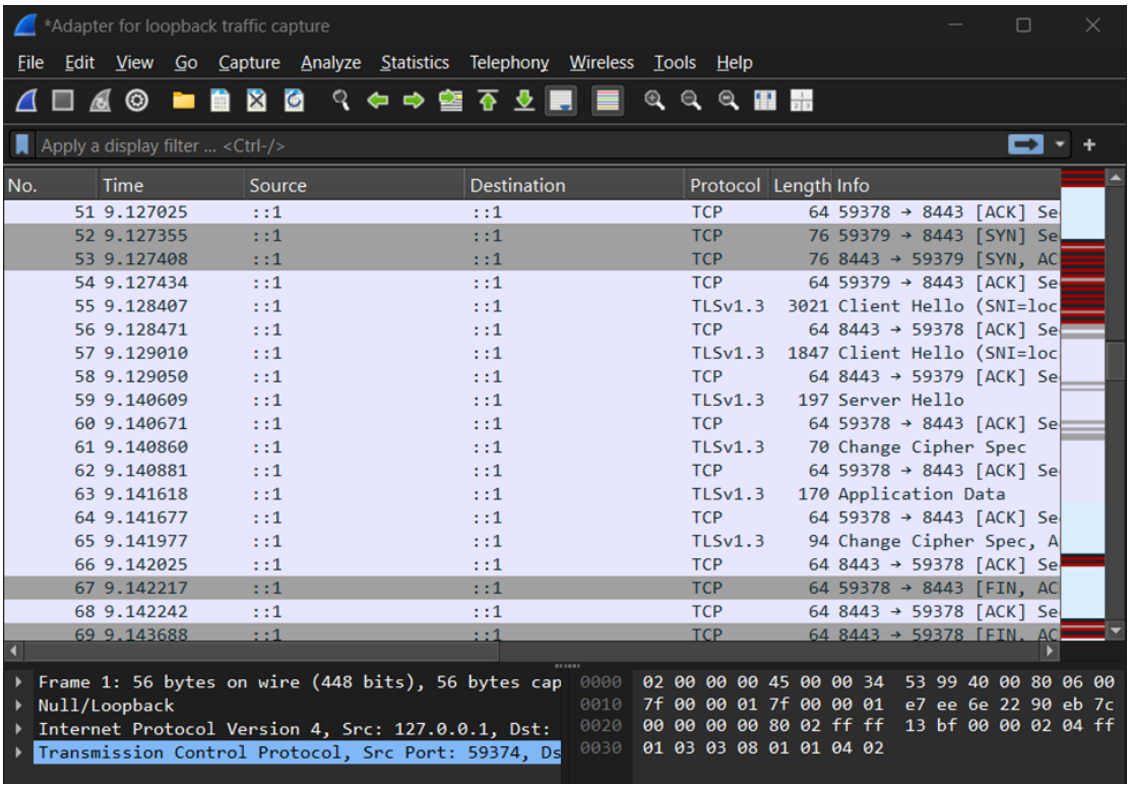
Caption: ↑ Here we are looking at the HTTP stream to find information about the request.



Caption: ↑ This shows the form item contents alongside the HTTP stream, directly showing the intercepted username and password.



Caption: ↑ Here we see our new HTTPS form's invalid certificate warning along with the certificate information.



Caption: ↑ Now when we use wireshark, we see that the request is hidden with the TLS protocol.