1. Configuring TLS for a Web Server

Tools Needed:

- Apache/Nginx Web Server
- OpenSSL
- Let's Encrypt (for free SSL certificates)
- A Linux-based server (Ubuntu/Debian recommended)

Steps:

1. **Install the Web Server**

For Apache:
```
sudo apt update && sudo apt install apache2 -y
```

For Nginx:
```
sudo apt update && sudo apt install nginx -y
```

2. **Generate an SSL Certificate Using Let's Encrypt**

Install Certbot:
```
sudo apt install certbot python3-certbot-nginx -y
```

Obtain an SSL certificate (replace `example.com` with your domain):
```
sudo certbot --nginx -d example.com -d www.example.com
```

3. Manually Generate a Self-Signed Certificate (Optional for Testing)
```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/selfsigned.key -out /etc/ssl/certs/selfsigned.crt
```

3.
4. **Configure the Web Server for HTTPS**

For Nginx, edit the config file:
```
sudo nano /etc/nginx/sites-available/default
```

Add these lines inside the `server {}` block:
```
listen 443 ssl;

ssl_certificate /etc/ssl/certs/selfsigned.crt;
```

```
ssl_certificate_key /etc/ssl/private/selfsigned.key;
```

Restart the server:
```
sudo systemctl restart nginx
```

5. **Test TLS Configuration**
   - Open a browser and visit https://example.com.
   - Use `openssl` to check TLS handshake:
     ```
     openssl s_client -connect example.com:443
     ```

---

2. Setting Up SSH with Key-Based Authentication

Tools Needed:

- OpenSSH
- A client machine (Linux/Mac or Windows with PuTTY)

Steps:

1. **Generate SSH Key Pair on Client Machine**
   ```
   ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -C
   "your_email@example.com"
   ```
2. **Copy Public Key to Server**
   ```
   ssh-copy-id username@server-ip
   ```
3. **Manually Add Public Key to Server (if `ssh-copy-id` is unavailable)**
   - Copy the content of `~/.ssh/id_rsa.pub` from the client machine.

Add it to the server's `~/.ssh/authorized_keys`:
```
echo "your-public-key" >> ~/.ssh/authorized_keys

chmod 600 ~/.ssh/authorized_keys
```

2. **Disable Password Authentication (Optional for Security)**

Edit SSH config file on the server:
```
sudo nano /etc/ssh/sshd_config
```

Set the following options:
plaintex
```
PasswordAuthentication no
```

```
PubkeyAuthentication yes
```

3. Restart SSH Service

```
sudo systemctl restart ssh
```

4. Test SSH Login Using Key-Based Authentication

```
ssh username@server-ip
```

---

3. Implementing a Secure Communication Channel

Tools Needed:

- OpenVPN or WireGuard (for VPN setup)
- GnuPG (for encrypted messaging)
- TLS/SSL for HTTPS communication

Option 1: Secure Communication with OpenVPN

Install OpenVPN on the server:

```
sudo apt install openvpn -y
```

1.
2. Set up OpenVPN with a basic configuration (or use WireGuard for modern security).

Option 2: Secure Messaging with GnuPG

Generate GPG key pairs:

```
gpg --full-generate-key
```

1. Encrypt a message:

   ```
   gpg --encrypt --recipient "recipient@example.com" message.txt
   ```

2. Decrypt the message:

```
gpg --decrypt message.txt.gpg
```

---

4. Analyzing Protocol Security Using Wireshark

Tools Needed:

- Wireshark
- tcpdump (for remote packet capture)

Steps:

1. **Install Wireshark**

On Linux:
```
sudo apt install wireshark -y
```

   - On Windows: Download from wireshark.org.
2. **Capture Network Traffic**
   - Run Wireshark and select the network interface (e.g., `eth0` or `wlan0`).
   - Start capturing packets.
3. **Filter Packets for TLS/SSH Traffic**

Use the filter:
nginx
```
tls or ssh
```

To inspect HTTPS, filter:
ini
```
tcp.port == 443
```

To inspect SSH traffic, filter:
ini
```
tcp.port == 22
```

4. **Analyze Encrypted Traffic**
   - Check if any weak ciphers are used.
   - Look for handshake messages to verify TLS negotiation.

Capture Remote Traffic Using `tcpdump`

```
sudo tcpdump -i eth0 port 443 -w capture.pcap
```

5.
    - Transfer `capture.pcap` to your local machine and analyze it in Wireshark.