**Task 1 (Basic Network Sniffer)**

**Objective**: It shows how data flows on a network and how network packets are structured.

**Step 1**: Create file named “network\_sniffer.py” and paste the following code.

**Step 2**: pip install pcapy

**Step 3**: python.exe -m pip install --upgrade pip

**Step 4**: Run the code by- python network\_sniffer.py

**Procedures**:

1. Creating socket which listens to the network
2. Infinite loop that listens to any data that comes across
3. Our socket when receives the data, stores it into 2 variables
   1. TCP
   2. UDP
4. Unpack ethernet frame
5. Return properly formatted MAC address (AA:BB:CC:DD:EE:FF)
6. Unpacks TCP segment
7. Unpacks UDP segment
8. Formats multiline data

**Task 2 (Phishing Awareness Training)**

**Objective**: Create a presentation or online training module about phishing attacks. Educate others about recognizing and avoiding phishing emails, websites, and social engineering tactics.

**Task 3 (Secure Code Review)**

**Objective**: Choose a programming language and application. Review the code for security vulnerabilities and provide recommendations for secure coding practices. Use tools like static code analyzers or manual code review.

**Let's break down the provided Python code step by step**:

1. **Flask Initialization**: The Flask application is initialized using Flask(\_\_name\_\_).
2. **Hardcoded Credentials**: For authentication, VALID\_USERNAME and VALID\_PASSWORD—hardcoded username and password—are defined.
3. **Login Route**: @app.route("/") is used to build a route for the root URL (/). For this route, GET and POST requests are handled by the login() function.
4. **Login Functionality**: The password and username given in the login form are extracted from the form data upon submission (POST request).
5. **Authentication**: The hardcoded credentials and the inputted ones are compared. The user is sent to the dashboard page (/dashboard) if they match. If not, the login page displays an error message.
6. **Rendering Login Page**: The login page (login.html) is produced if the authentication fails or the request method is GET.
7. **Dashboard Route**: @app.route("/dashboard").create is used to build a route for the dashboard page (/dashboard). Requests for this route are handled by the dashboard() function.
8. **Dashboard Functionality**: The dashboard page (dashboard.html) is rendered by the dashboard() function.
9. **Application Execution**: app.run(debug=True) is used to launch the Flask application. For development purposes, debug mode is activated, which displays thorough error messages.
10. **Static Code Analysis**: Use static code analysis tools to automatically scan the codebase for security vulnerabilities and coding best practices.

**Task 4 (Network Intrusion Detection System)**

**Objective**: Develop a network-based intrusion detection system using tools like Snort or Suricata. Set up rules and alerts to identify and respond to suspicious network activity.