Submission Guidelines

Due Date:

- Final submission deadline: **1 September 2025**

Submission Incentives & Penalties:

Date	Status	Remarks
By **28 August 2025, 2100 hours**	Early Bird Submission	Eligible for up to **5% bonus marks**
1 September 2100 hours	Final Deadline	
After **1 September 2025**	Late Submission	**5% penalty per day** will apply

Presentation & Peer Evaluation:

- Scheduled during **T3W10, T4W1 and T4W2** (Term 4, Weeks 1 and 2)

Total Marks:

- The assignment carries a total of **100 marks**

Assignment Guidelines

1. **Group Work:**

   This assignment must be completed in **groups of four**. Working in a group ensures the workload is manageable and provides every member the opportunity to contribute meaningfully.

2. **Responsibility to Share Knowledge:**

   It is the **duty of each group member** to ensure that others understand the work they are doing. During the assessment, **any group member may be asked to explain or modify the work**, regardless of who originally work on it. This encourages collaboration and shared understanding.

3. **Open-Ended Design:**

   The question descriptions are intentionally brief to encourage **creative thinking** and allow flexibility in your implementation approach.

4. **Use of AI Tools**

   You are permitted to use ChatGPT-4 and other AI tools to assist with this assignment. However, you must:

   - Document all prompts used during development, including any iterations or refinements. This should be included as part of your final submission.
   - Verify the accuracy of any output generated by AI tools. You are responsible for ensuring that the information, code, or suggestions provided by AI are correct, relevant, and appropriate for your assignment.

- Clearly indicate how and where AI tools were used in your workflow (e.g., code generation, debugging, explanation, design ideas).

5. **Testing:**

   Include **test cases** to demonstrate the correctness and reliability of your program or setup. This is essential for validating your solution.

6. **Programming Language:**

   You may use either **Java or Python** for this assignment.

7. **Assumptions:**

   If any part of the question is unclear, clearly **state your assumptions** in your report. This ensures transparency and consistency in your approach.

8. **Peer Review:**

   A **peer review component worth 20 marks** will be included. Each group will evaluate the work of other groups and assign scores based on the quality and clarity of their presentations.

Each question is worth 20 marks.

Question 1: Distributed Multi-User Application

Objective:

Design and implement a distributed client-server application using sockets and other networking classes available in Java or Python. The application should feature a graphical user interface (GUI) and support multi-threading to handle multiple clients concurrently.

Requirements:

- The application must follow the client-server architecture.
- It should be GUI-based for both server and client sides.
- Multi-threading must be implemented to support simultaneous client connections.

Suggested Use Cases:

1. Online Networking Contest Simulator

   Create a competitive quiz platform where at least two clients participate in a networking-themed contest. The server will distribute questions (based on topics covered in CS6132), and clients will submit their answers in real-time.

2. Collaborative Whiteboard Application

   Develop a real-time drawing tool where multiple clients can connect to a central server and draw collaboratively. This project emphasizes real-time data synchronization, network programming, and GUI development.

Submission Requirements:

- Complete source code along with executable files.
- A brief report explaining the system architecture, design decisions, and implementation details.
- A short video tutorial demonstrating how to set up and run the application.

Question 2: Kali Linux Security Project

Instructions:

Choose one of the following security-focused projects to implement using Kali Linux tools only. Each option explores a different aspect of cybersecurity, including password security, malware analysis, network attacks, and social engineering.

Option A: Password Cracking Tool

Objective:

Demonstrate the risks of weak password practices by using Kali Linux tools to crack hashed passwords.

Tools You May Use:

- Hashcat – for dictionary, brute-force, and rule-based attacks
- John the Ripper – for cracking various hash formats
- Crunch – for generating custom wordlists

Requirements:

- Generate a set of hashed passwords using tools like md5sum, sha256sum, or openssl.
- Apply at least **two cracking techniques** (e.g., dictionary attack with rockyou.txt, brute-force with hashcat -a 3).
- Measure and compare cracking times for different methods.
- Analyze password weaknesses (e.g., short length, common patterns).
- Recommend best practices (e.g., long passphrases, use of 2FA).

Option B: Malware Analysis Sandbox

Objective:
Set up a basic sandbox environment to safely analyze suspicious files.

Tools You May Use:

- VirtualBox or QEMU – for virtualization
- inotify-tools – to monitor file changes
- tcpdump – to capture network traffic
- pspy – to observe running processes

Requirements:

- Create an isolated VM environment for malware execution.
- Monitor system behavior during execution (file, network, process).
- Log and summarize observed activities.
- Discuss limitations (e.g., evasion techniques, lack of deep analysis).
- Suggest improvements (e.g., integration with Cuckoo Sandbox).

Option C: Fake Wi-Fi Hotspot (Social Engineering Attack)

Objective:
Simulate a rogue access point to demonstrate risks of unsecured wireless networks.

Tools You May Use:

- airbase-ng or hostapd – to create a rogue AP
- wifiphisher, or manual setup using dnsmasq + nginx – for captive portal
- ettercap or Wireshark – for traffic interception

Requirements:

- Set up a fake Wi-Fi hotspot with a captive portal.
- Demonstrate credential capture via a fake login page.
- Explain associated risks (e.g., session hijacking, data theft).
- Recommend protections (e.g., VPN usage, HTTPS enforcement).

Option D: Phishing Attack (Credential Harvesting)

Objective:
Create and host a phishing page to demonstrate credential harvesting techniques.

Tools You May Use:

- SET (Social Engineering Toolkit) – to generate phishing pages
- Apache2 – to host the page locally
- Ngrok – to expose the page to the internet (optional)
- Wireshark or Ettercap – to monitor traffic (optional)

Requirements:

- Design and deploy a phishing page targeting login credentials.
- Analyze the effectiveness of the attack.
- Discuss ethical considerations and countermeasures (e.g., user education, browser warnings).

What to Submit:

- **Source code/scripts** used in the implementation.
- **Screenshots or screen recordings** demonstrating key steps and outcomes.

- A **short report** explaining your setup, methodology, findings, and recommendations.
- A **brief video tutorial** showing how to set up and run your project.

Question 3: Network Layer Simulation Project

Instructions:

Choose any one of the following programming tasks related to the Network Layer. Your application should include a simple graphical user interface (GUI) to visualize the process or results.

Option A: Router Forwarding Simulation

Develop a program that simulates the forwarding process in a router.

- Input: A routing table and the destination address of a packet.
- Output: The outgoing interface determined by the routing logic.

Option B: Routing Algorithm Simulation

Implement either the Distance-Vector or Link-State routing algorithm.

- The program should simulate the algorithm and display routing tables similar to those shown in class (refer to Appendix 1).
- Output: Step-by-step updates of routing tables for each node.

Option C: Address Block Analysis

Create a program that performs the following:

1. Given any IP address in a block (using classless addressing), determine the first and last address in that block.
2. Identify the range of addresses allocated to each organization and the range of unallocated addresses.

Option D: IP Fragmentation Demonstration

Build a program to demonstrate or animate the IP fragmentation process.

- Input: Packet size, MTU (Maximum Transmission Unit), and header size.
- Output: Fragmented packets with offset and flags, visualized through the GUI.

What to Submit:

- Source code along with the executable file.

- A short report explaining your implementation, with screenshots or a brief video demonstrating the application in action.

Question 4: AI/ML-Based Network Attack Detection

Objective:
Use AI/ML techniques in combination with Kali Linux tools and Python to detect and analyze network-based attacks such as intrusions, phishing, or DDoS.

Choose any one of the following tasks.

Task Options (Choose One):

Option 1: Detect Malicious Network Traffic Using Machine Learning

- Capture network traffic using tools like tcpdump or Wireshark.
- Preprocess the captured data (e.g., extract features such as packet size, protocol, flags).
- Train a machine learning model (e.g., Random Forest, CNN) to classify traffic.
- Test the model using simulated attack traffic (e.g., DDoS, port scans via hping3).
- Evaluate performance (accuracy, false positives/negatives) and discuss results.

Option 2: Phishing URL Detection Using AI

- Collect datasets of phishing and legitimate URLs (e.g., from PhishTank and Alexa).
- Extract relevant features (e.g., URL length, domain age, special characters).
- Train a classifier (e.g., Logistic Regression, LSTM) to detect phishing URLs.
- Optionally deploy the model as a real-time filter (e.g., block URLs using iptables).

Option 3: Anomaly Detection in Network Logs

- Parse system or application logs (e.g., Apache, SSH logs from /var/log/).
- Apply unsupervised learning techniques (e.g., Isolation Forest, Autoencoders).
- Detect anomalies such as brute-force login attempts or unusual access patterns.
- Analyze and visualize the results, highlighting flagged events.

Tools/Frameworks (All Available in Kali Linux):

- **Scapy** – Packet analysis
- **Wireshark/Tshark** – Traffic capture

- **Scikit-learn** – Classical ML
- **TensorFlow/Keras** – Deep learning
- **Jupyter Notebook** – Data analysis and visualization

What to Submit:

- **Python code or Jupyter Notebook** used for implementation.
- **Dataset(s)** used (e.g., PCAP files, logs, or URL lists).
- A **short report** including:
    - Model performance metrics (e.g., accuracy, ROC curves)
    - Comparison with rule-based detection methods
    - Discussion of ethical implications (e.g., bias in training data)
- **Screenshots or a short video** demonstrating your setup and results.

Appendix

A1

Initial distance vector for node A

Destination node	Next node	Cost
A	-	-
B	B	4
C	-	∞
D	-	∞
E	E	2
F	F	6

For distance-vector algorithm

T	B	C	D	E	F
{A}	4 A→B	∞ -	∞ -	2 A→ E	6 A→F
{A, E}	4 A→B	∞ -	∞ -	2 A→ E	5 A→E→ F

For link-state algorithm.