1. Wireshark Traffic Analysis – Capture and interpret basic network packets; identify protocols like HTTP, DNS, and TCP handshakes.
Goal: Understand network traffic flow and analyze how different protocols work.
End Goal: Students should submit captured .pcap files with screenshots and a report describing at least three identified protocols.

2. Port Scanning with Nmap – Understand open/closed ports, services, and basic network mapping.
Goal: Learn to use Nmap for scanning hosts and identifying running services.
End Goal: Students should provide a scan report highlighting open ports, detected services, and security recommendations.

3. Firewall Rules Demo – Configure and test simple inbound/outbound firewall rules on Linux or Windows.
Goal: Understand how firewall configurations protect a system from unauthorized access.
End Goal: Submit screenshots and a configuration report showing implemented rules and their test results.
4. Simple RSA Encryption Demo – Generate keys and encrypt/decrypt text using the RSA algorithm.
Goal: Learn public-key cryptography fundamentals by implementing RSA manually.
End Goal: Provide working Python/C code with generated keys and an example of encrypted and decrypted text.
5. Hashing Practice – Compare MD5, SHA-1, and SHA-256 hashes for different files.
Goal: Understand how hashing ensures integrity and detects file tampering.
End Goal: Submit a table comparing hash outputs for multiple files and discuss which algorithm is more secure.
6. Password Hashing with Salt – Create a small password manager that hashes and stores passwords securely.
Goal: Implement secure password handling using salting and hashing.
End Goal: Submit the source code and demonstrate storing and verifying user credentials securely.
7. Text Steganography – Hide secret messages inside text or image files and retrieve them manually.
Goal: Explore how hidden communication can be achieved through steganography.
End Goal: Provide original and modified files, along with scripts and screenshots showing hidden message extraction.
8. Linux File Permissions Project – Explore chmod, user groups, and file ownership for secure access control.
Goal: Understand user privilege separation and secure file access.
End Goal: Provide examples of different permission settings and their effects, with screenshots or terminal logs.
9. System Log Monitoring – Use Linux journalctl or Windows Event Viewer to track login attempts.
Goal: Learn to use system logs for basic intrusion detection.
End Goal: Submit a log report highlighting unusual login attempts and how they were detected.
10. Creating a Backup and Recovery Plan – Automate simple file backups with integrity verification (checksums).

Goal: Develop a script-based data backup and verification workflow.
End Goal: Submit the working backup script and checksum validation output demonstrating successful recovery.
11. Understanding User Privileges – Demonstrate least privilege concept with different user accounts on Linux.
Goal: Learn how user privilege management enhances system security.
End Goal: Provide user role configuration steps and a report comparing admin vs. limited user access outcomes.
12. Cookie & Session Security – Show how cookies can be hijacked and how secure cookies prevent it.
Goal: Understand session management and cookie security principles.
End Goal: Demonstrate session hijacking in a test setup and explain how secure flags mitigate the attack.
13. Exploring HTTPS Certificates – Use browser tools to inspect and explain SSL/TLS certificate details.
Goal: Learn about HTTPS encryption and certificate-based trust.
End Goal: Submit screenshots and an explanation of certificate chains, validity, and issuer details.
14. Password Strength Analyzer – Develop a tool that rates password strength based on rules (length, symbols, etc.).
Goal: Implement password evaluation logic for secure user credential creation.
End Goal: Provide tool code and test cases demonstrating weak vs. strong password classifications.
15. Cyber Hygiene Audit – Students analyze their devices for common vulnerabilities (unpatched apps, weak passwords).
Goal: Increase awareness of basic cybersecurity practices and local vulnerabilities.
End Goal: Submit an audit checklist and a summary report of vulnerabilities found and fixes applied.
16. Cybersecurity Policy Draft – Write a simple IT security policy for a small business or school lab.
Goal: Learn how organizations design and enforce security policies.
End Goal: Submit a 2–3 page written policy covering access control, data protection, and password standards.
17. Basic Intrusion Detection – Write a script that monitors network pings or login attempts and alerts on anomalies.
Goal: Implement a simple Intrusion Detection System using Python or shell scripting.
End Goal: Submit the code and a log of alerts triggered by simulated suspicious activity.
18. Ransomware Behavior Simulation – Simulate file encryption/decryption safely (without real malware).
Goal: Understand how ransomware encrypts data and how to recover safely.
End Goal: Submit the encryption/decryption code, test results, and discussion of mitigation techniques.
19. Cloud Data Encryption Layer
Goal: Create a script that encrypts files before uploading to a cloud service (AWS S3 or mock local cloud). - Implement AES encryption/decryption and verify integrity with SHA-256. - Demonstrate client-side encryption for privacy.
20. Secure Access Control for Cloud APIs

Goal: Build a REST API (Flask or FastAPI) with role-based access control (RBAC). - Use JWT authentication with token expiry and role-specific permissions. - Log and handle failed access attempts securely.
21. Android App Data Protection Goal: Develop a simple Android app (e.g., a notes app) that encrypts local data using the Android Keystore. - Implement AES encryption for stored data and test data extraction resistance.
22. Secure Messaging App (Local Network Version) - Build a basic chat app that uses RSA for key exchange and AES for encrypted messaging. - Add message authentication and timestamp validation. - Test it over local Wi-Fi or LAN.
23. IoT Sensor Data Encryption Goal: Learn how to protect sensor data before sending it over Wi-Fi. What students do: Use an ESP32 or Raspberry Pi to collect simple data (e.g., temperature or light). Write a small script (Python/C++) to encrypt the data using AES before sending. Use another device (or laptop) to decrypt and display the received message. Learning Outcome: Basic encryption/decryption workflow and understanding how IoT devices keep data private.
24. Secure Communication Between Two IoT Devices Goal: Demonstrate secure message exchange between two IoT boards. What students do: Use two ESP32 boards or two laptops with Python serial/Wi-Fi communication. Implement a simple shared secret key for encoding/decoding messages. Add message checksums or sequence numbers to prevent tampering. Learning Outcome: Hands-on understanding of authentication, integrity, and safe data transfer in IoT networks.