Semester Project — Phase 2 Report

**Cyber Range Simulation: Network Vulnerability and Attack Queue System**

Course: Data Structures and Algorithms

Team Members: Warda Gul (2024663), Asjid Samad (2024119), . Muhammad Abdullah (2024321)

Instructor:Dr. Zubair Ahmed

Date:19 Oct’25

# 1. Introduction

The purpose of this deliverable is to simulate a **basic network defense environment** where systems are represented as nodes with different levels of vulnerability.  
Users can **queue attacks** targeting specific IP addresses, and the program simulates whether these attacks succeed or fail based on predefined vulnerability and patching factors.

This simulation helps demonstrate key data structure concepts such as **arrays, queues, and structures**, while introducing basic **cybersecurity logic** (e.g., vulnerability scoring and patch impact).

# 2. Objectives

The main objectives of this phase were:

1. To model a network of interconnected systems using a structure (struct SystemNode).
2. To implement a manual queue (struct AttackQueue) that stores attack requests in a FIFO (First-In-First-Out) order.
3. To simulate attack processing based on system vulnerability and patching status.
4. To demonstrate queue operations such as enqueue, dequeue, and status checking (isEmpty, isFull).
5. To provide an interactive menu-driven interface for the user to interact with the system.

# 3. System Design

## 3.1 Data Structures Used

**a) SystemNode Structure**

Represents each system (or node) in the simulated network.  
It stores:

* name: Name of the system (e.g., “Main Server”)
* ip: IP address
* compromised: Boolean indicating if the system is hacked
* vuln: Vulnerability level (0.0 – 1.0)
* patched: Indicates whether the system has a security patch

struct SystemNode {

string name

string ip;

bool compromised;

double vuln;

bool patched;

};

**b) AttackQueue Structure**

Implements a fixed-size queue to manage attack requests:

* queue [MAX\_QUEUE]: Array to hold attack targets (IP addresses)
* front and rear: Track the queue ends
* count: Number of items currently in queue

struct AttackQueue {

string queue[MAX\_QUEUE]; // array to hold IP strings

int front;

int rear;

int count;

};

## 3.3 Constants

* const int MAX\_SYSTEMS = 5 : Total systems in the network
* const int MAX\_QUEUE = 10 : Maximum attack queue length

These constants define network and queue limits, ensuring controlled simulation.

# 4. Functional Modules

| **Function Name** | **Purpose / Description** |
| --- | --- |
| initQueue() | Initializes queue counters to default values. |
| isEmpty() / isFull() | Checks queue status before enqueue/dequeue operations. |
| enqueue() | Adds a new attack request (target IP) to the queue. |
| dequeue() | Removes and returns the next attack to process. |
| setupNetwork() | Initializes 5 systems with varying vulnerability and patch status. |
| showNetwork() | Displays the status of all systems (secure or compromised). |
| attemptExploit() | Determines if an attack is successful based on effective vulnerability. |
| processAttack() | Retrieves a system by IP, attempts an exploit, and updates status. |

# 5. Program Flow

1. **Network Initialization:**  
   The system automatically creates 5 nodes (Main Server, Admin PC, Database, Firewall, Backup Server) with different vulnerability scores and patch statuses.
2. **Queue Initialization:**  
   The attack queue is created and set to empty.
3. **User Interaction (Main Menu):**
   * **Option 1:** View network status
   * **Option 2:** Add an attack to the queue
   * **Option 3:** Process next queued attack
   * **Option 4:** Exit program
4. **Attack Simulation:**  
   Each attack dequeued is matched to a system by IP.
   * If the system is unpatched and has vuln ≥ 0.6, it becomes compromised.
   * If patched, the effective vulnerability is reduced by 60% (vuln \* 0.4).

# 6. Sample Output

=== CYBER RANGE GAME — PHASE 1 (with vuln) ===

Simulated network defense environment

1. View network status

2. Add attack to queue

3. Process next attack

4. Exit

Enter choice: 1

=== SYSTEM STATUS ===

Main Server (192.168.0.1) - [SECURE] vuln=0.3

Admin PC (192.168.0.2) - [SECURE] vuln=0.45 [PATCHED]

Database (192.168.0.3) - [SECURE] vuln=0.85

Firewall (192.168.0.4) - [SECURE] vuln=0.2 [PATCHED]

Backup Server (192.168.0.5) - [SECURE] vuln=0.55

After queueing and processing an attack:

Enter target IP to attack: 192.168.0.3

[+] Attack added to queue: 192.168.0.3

Processing...

[!] ALERT: Database (192.168.0.3) has been hacked!

# 7. Key Concepts Demonstrated

1. **Queue Implementation** — Manual circular queue using array indexing and modulo operations.
2. **Structured Data Representation** — Use of struct for organizing system attributes.
3. **Decision Logic** — Conditional attack success using vulnerability threshold.
4. **Menu-Driven Interface** — Continuous program loop with user input handling.
5. **Simulation Design** — Integration of data structures into a simple cybersecurity-themed model.

# 8. Limitations and Future Improvements

* The current model is single-threaded and deterministic (no randomness in exploit success).
* Future phases could include:
  + Dynamic vulnerability updates (e.g., after multiple attacks).
  + Random probability-based exploit success.
  + Linked list–based dynamic queue (instead of fixed array).
  + Visualization of network connections.
  + Logging and report generation of attacks.

# 9. Conclusion

This project segment successfully demonstrates how fundamental data structures (queues and structs) can be used to build a **realistic cyberattack simulation**.  
It showcases practical understanding of queue operations, structured data handling, and modular program design.

Through this simulation, students can better understand how systems with varying vulnerabilities respond to attacks and how defensive mechanisms like patching affect overall network security.