**DSA DELIVERABLE\_2 REPORT**

**Project Title: Secure Network Assessment Tool**

**1. Initial Implementation of Core Data Structures**

The project implements a simplified Cyber Risk Scorer system that evaluates how secure each computer system is based on its configuration.  
We used core data structures studied in class — Structure, Array, and Linked List — to represent, store, and manage system data efficiently.

* **Structure (struct System)**

struct System {

string name;

int password\_strength;

int open\_ports;

bool firewalls;

int risk\_core;

};

**Functionality:**

* Each System represents one computer or device in a network.
* Stores attributes that affect cyber risk: password strength, open ports, firewall status, and total risk score.
* Helps organize related data into one logical unit — showing the concept of a user-defined data type.
* **Array (int riskWeights[3])**

int risk\_weights[3] = {30, 25, 20};

**Functionality:**

* Holds constant weight values for different risk parameters.
* Demonstrates the use of arrays for fixed-size data storage.
* Allows easy indexing and retrieval of weights for calculations.

**c) Linked List (Dynamic Storage of Systems)**

struct Node {

System\* data;

Node\* next;

};

struct SystemList {

Node\* head;

SystemList() { head = NULL; }

void add\_system(System\* sys) {

Node\* newNode = new Node();

newNode->data = sys;

newNode->next = head;

head = newNode;

}

};

**Functionality:**

* Dynamically stores systems using nodes and pointers.
* Each node contains a pointer to a System object and a pointer to the next node.
* Demonstrates dynamic memory allocation and pointer-based data organization, key topics in DSA.

**2. Algorithms Used (Integration with Data Structures)**

* **Risk Calculation Function**

int calculateRisk(System\* sys) {

int score = 0;

score += (10 - sys->password\_strength) \* 3;

score += sys->open\_ports \* 8;

if(!sys->firewalls) score += 25;

sys->risk\_score = (score > 100) ? 100 : score;

return sys->risk\_score;

}

* Uses simple arithmetic logic to compute each system’s risk score.
* Demonstrates how algorithms operate on structured data.
* **Selection Sort Algorithm**

void sort\_systems(System\* systems[], int n) {

for(int i = 0; i < n-1; i++) {

int max\_index = i;

for(int j = i+1; j < n; j++) {

if(systems[j]->risk\_score > systems[max\_index]->risk\_score)

max\_index = j;

}

if(max\_index != i) {

System\* temp = systems[i];

systems[i] = systems[max\_index];

systems[max\_index] = temp;

}

}

}

* Ranks systems by risk score (highest to lowest).
* Demonstrates sorting techniques studied in class.

**3. Sample Input/Output**

* **Input (Hardcoded Example):**

System sys1 = {"WebServer", 7, 2, true, 0};

System sys2 = {"Database", 3, 5, false, 0};

System sys3 = {"PC", 9, 1, true, 0};

* **Output:**

CYBER RISK SCORER - CORE IMPLEMENTATION

Calculating Risk Scores...

RANKED SYSTEMS

1. Database - 100/100 (HIGH)

2. WebServer - 41/100 (LOW)

3. PC - 17/100 (LOW)

**4. Reflection of Classroom Concepts**

| Concept | Implemented Through | Explanation |
| --- | --- | --- |
| Structure | struct System | Represents complex data as a single logical unit |
| Array | riskWeights[3] and system array | Stores multiple elements sequentially for easy access |
| Linked List | SystemList | Demonstrates dynamic memory and pointer linkage |
| Sorting Algorithm | Selection Sort | Applies algorithmic thinking for ranking |
| Functions | Modular approach (calculateRisk, sortSystems) | Encourages code reusability and abstraction |

**5. Conclusion**

This project successfully demonstrates how Data Structures and Algorithms can be applied to a real-world concept.  
We effectively used structures, arrays, and linked lists to represent and manage system data, while selection sort and risk computation algorithms were used to analyze and rank systems.

The project reflects all the principles taught in class, showing both theoretical understanding and practical implementation.