A01713396 - Santiago Martin del Campo Soler

Reflexión de la Actividad 4.3 – Grafos.

En esta actividad 4.3, se trabajó con una bitácora de tráfico de red que registra las conexiones entre distintas direcciones IP. A diferencia de las actividades anteriores, donde la información se analizaba de forma cronológica, en este caso el enfoque se centró en estudiar las relaciones existentes entre las IPs, ya que una misma dirección podía comunicarse con múltiples destinos.

Para analizar esta conexión entre las distintas direcciones IP leímos el archivo "bitacoraGrafos.txt", la bitácora mencionada anteriormente, posteriormente almacenamos los datos en una estructura de grafos, esta misma basada en listas de adyacencia.

Cada dirección IP fue representada mediante un Nodo, una conexión como una arista dirigida desde la dirección IP de origen hasta la IP de destino.

Esta forma de representación permitió organizar y analizar la información de manera más clara y eficiente, facilitando el estudio de las relaciones entre los nodos, algo que no habría sido posible de forma óptima utilizando únicamente arreglos o listas lineales.

Métodos Utilizados en esta actividad

- **Uso de Heap para identificar las IPs con mayor grado:**

  Para determinar las siete direcciones IP con el mayor número de conexiones hacia otras IPs, se utilizó la estructura de datos Heap que ya hemos implementado en actividades previas. La elección fue fundamental, ya que ordenar todas las direcciones IP mediante otros algoritmos tradicionales como Bubble Sort implicaría una complejidad de O(n^2), que resultaría en una poca eficiencia al contar con un gran número de datos.

Por otro lado, Heap nos permite insertar y extraer elementos con una complejidad de O(log n), lo cual nos facilitó la obtención de las direcciones IP con mayores conexiones y permite identificar de manera eficiente a la dirección posible boot master.

- **Cálculo del Camino Más Corto**

  Una vez que identificamos el boot master, se evaluó qué dirección IP implicaba un mayor nivel de procesamiento al analizar las rutas dentro del grafo. El algoritmo utilizado para este análisis (Dijkstra) tiene una complejidad aproximada de O(V+E) log V), donde V representa el número total de nodos y E el número de conexiones existentes. Este enfoque nos permite explorar las conexiones de la red de forma ordenada y eficiente, evitando recorridos innecesarios o revisar todas las trayectorias posible, lo que incrementa considerablemente el costo computacional

Importancia del Uso de Grafos en estos casos

El uso de grafos en el análisis de tráfico de red es fundamental, ya que permite representar de una manera más natural las relaciones entre múltiples direcciones IP y sus conexiones. A diferencia de algunas estructuras lineales como vectores o listas simples, los grafos nos permiten representar de una forma natural las múltiples conexiones que puede tener una dirección IP dentro de una red.

En el contexto de la ciberseguridad, está estructura de datos puede resultar clave para analizar posibles amenazas, debido a que puede identificar direcciones IP que cuentan con actividad actividad inusual.

Reflexión final

El uso de grafos ha sido una novedad dentro de esta actividad, ya que permite visualizar de manera más clara las relaciones y posibles problemas dentro de una red, facilitando la toma de decisiones. Gracias a esta estructura, es posible aplicar algoritmos eficientes que ayudan a mejorar la detección de ataques y a realizar análisis más precisos basados en el comportamiento del tráfico de red.

En mi caso se me dificulto implementar y entender del todo los grafos, pero con la ayuda de mis compañeros y una pequeña investigación pude comprender de una manera sencilla el uso de grafos y su importancia.

Referencias Bibliográficas:

Bose, A., & Bose, A. (2025, 14 febrero). *Malware Detection: How to detect and remove malware ?* Seceon Inc. https://seceon.com/malware-detection-how-to-detect-and-remove-malware/

GeeksforGeeks. (2025, 23 julio). Implementation of Graph in C++. GeeksforGeeks. https://www.geeksforgeeks.org/cpp/implementation-of-graph-in-cpp/

GeeksforGeeks. (2026, 21 enero). Dijkstra's algorithm. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/dijkstras-shortest-path-algorithm-greedy-algo-7/

Linkurious. (2026, 8 enero). Cybersecurity graph visualization: Assessing vulnerabilities. Linkurious. https://linkurious.com/blog/graph-data-visualisation-cyber-security-threats-analysis/

PuppyGraph. (2025, 21 septiembre). PuppyGraph | Query your relational data as a graph. no ETL. https://www.puppygraph.com/blog/graphs-for-cybersecurity#why-do-you-need-graphs-for-cybersecurity

GeeksforGeeks. (2025c, diciembre 22). *Heap sort*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/heap-sort/