Tecnológico de Monterrey

REFLEXIÓN - ACT 4.3 INTEGRAL

DANTE HERNÁNDEZ RAMÍREZ - A01668070

Programación de estructuras de datos y algoritmos fundamentales

(Gpo 573)

31 de Enero del 2026

Reflexión

Previa a esta actividad, el análisis de los registros estaban principalmente centrados en la frecuencia de apariciones de las diferentes sesiones de inicio de los registros de IP. Esto principalmente gracias al uso de estructuras de datos jerárquicas (Binary Heap), permitiendo así la detección de comportamientos anormales en los volúmenes de registros, siendo una idea clara de la implementación de un sistema de ciberseguridad.

Para la realización de la actividad presente, se contempla un cambio significativo: los datos tomados en cuenta no son únicamente direcciones IP, sino que los registros se empiezan a comprender con una relación entre todos los datos contenidos en ellos. Con esta modificación no solo se identifican las direcciones IP con comportamientos anormales, sino que también permite comprender cómo se relacionan entre si, ampliando el análisis con el uso de grafos.

El código actual se comprende de:

- Lectura de la bitácora y construcción del grafo dirigido y ponderado.
- Representación de las conexiones mediante una lista de adyacencia.
- Cálculo del grado de salida de cada nodo del grafo.
- Identificación de las IPs con mayor grado de salida mediante un Heap.
- Determinación del boot master.
- Cálculo de los caminos más cortos desde el boot master hacia el resto de las IPs.

ESTRUCTURAS DE DATOS Y GRAFOS

En la actividad anterior se utilizó Binary Heap como estructura principal para ordenar y obtener las IPs con mayor frecuencia de accesos. Aunque esta aproximación fue adecuada para cumplir el objetivo planteado, su análisis se limitaba a datos individuales y no contemplaba la interacción entre IPs.

COn las modificaciones implementadas, el uso de grafos permite representar de manera más realista el comportamiento de la red. Esta estructura reduce el consumo de memoria y facilita el recorrido de las conexiones existentes sin necesidad de evaluar relaciones inexistentes.

ANÁLISIS DE COMPLEJIDAD

Con esto, la solución implementada se podría catalogar como:

- Lectura de la bitácora y construcción del grafo: O(n + m)
- Cálculo del grado de salida de cada nodo: O(n + m)
- Inserción y extracción de IPs con mayor grado usando Heap: O(n log n)
- Obtención de las 7 IPs con mayor grado: O(7 log n)
- Algoritmo de Dijkstra para caminos más cortos: O((n + m) log n)

Siendo el algoritmo de Djikstra el término más importante para la resolución del código, se podría definir el código con la complejidad de O((n + m) log n).

CONCLUSIÓN

El uso de heaps y el algoritmo de Dijkstra demuestra la importancia de seleccionar correctamente las estructuras de datos y algoritmos en función del problema a resolver. Más allá del cumplimiento técnico, se refuerza la idea de que un buen análisis computacional puede marcar la diferencia entre una detección superficial y una evaluación integral de posibles amenazas dentro de un sistema.

REFERENCIAS

Cormen, T. H., Leiserson, C. E., Rivest, C. L., & Stein, C. (2022). Introducción a los algoritmos (4th ed.). MIT Press. https://mitpress.mit.edu/9780262046305/introduction-to-algorithms/

Dijkstra's shortest path algorithm. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/dijkstras-shortest-path-algorithm-greedy-algo-7/

Cybersecurity and cyberwar: What everyone needs to know. Oxford University Press. https://global.oup.com/academic/product/cybersecurity-and-cyberwar-9780190654102