Ana Paula Katsuda Zalce, A01025303
Programación de estructuras de datos y algoritmos fundamentales
Prof. Jorge Rodríguez Ruiz
3 de Diciembre del 2021

Reflexión Final

Durante el presente curso, se llevaron a cabo cinco distintas partes del reto, cuyo objetivo era analizar las conexiones de una compañía e identificar el comportamiento de ataques cibernéticos. En dichas partes, se utilizaron los conocimientos adquiridos durante la clase sobre estructuras de datos y su implementación. A continuación se muestra un breve resumen de lo realizado en cada parte del reto:

- Parte 1: en este caso, se utilizaron algoritmos de búsqueda (binaria) y ordenamiento (quicksort) para tener una visualización y un entendimiento de los datos con los que se trabajaría durante todo el reto. Se encontraron datos generales tales como la cantidad de registros de conexiones de computadoras tomadas.
- Parte 2: se utilizaron las estructuras de datos “stack” y “queue” para determinar las conexiones entrantes y salientes de un IP determinado. Esto fue de gran utilidad para indagar en las interacciones realizadas por ciertas computadoras.
- Parte 3: se utilizaron diccionarios y “hash” para identificar dominios anómalos y conexiones extrañas de las computadoras dentro de la empresa analizada. Aquí se descubrió que las computadoras internas tienen conexiones tanto entrantes como salientes.
- Parte 4: para esta parte, se utilizaron diccionarios y árboles de búsqueda binaria con el fin de obtener los sitios más visitados por las computadoras de la compañía en los días registrados.
- Parte 5: en esta parte se utilizaron grafos para determinar la cantidad de conexiones que se hacen a los distintos sitios web y las conexiones entre las computadoras internas de la compañía. A partir de lo anterior, fue posible identificar componentes de un ataque DDoS y comprender mejor lo que sucedía en la red.

Tomando en cuenta las soluciones a las que se llegó para las distintas partes del reto, considero que las estructuras de datos son una herramienta esencial para realizar análisis eficientes y completos. En lo personal, creo que la efectividad y eficiencia de dichas estructuras depende, hasta cierto punto, del problema que se quiere resolver. Sin embargo, pienso que dos de las más eficientes son los diccionarios y los grafos; lo anterior dado que permiten una organización de datos muy completa y el manejo de los mismos me pareció extremadamente más sencilla. Puesto a que los diccionarios utilizan una “llave”, contribuyen en la realización de comparaciones y para acceder fácilmente a cierto elemento. Por otro lado, los grafos permiten relaciones complejas entre nodos, por lo que su uso sirve en más situaciones y en soluciones íntegras.

Retomando las soluciones a las que se llegó, considero que las más eficientes son: la solución de la parte 1, la solución de la parte 3 y la solución de la parte 5. La solución de la parte 1 fue mediante algoritmos con una complejidad temporal bastante buena (se eligieron los algoritmos con ese fin), lo que hizo que el análisis inicial de datos fuera muy eficiente. Por otro lado, como ya se mencionó, el uso de los diccionarios y grafos en las partes 3 y 5 es esencial para una solución más eficiente.

En cuanto a las soluciones que tienen espacio de mejora, sin duda la menos eficiente fue la parte 4. En este caso, se realizaron pasos innecesarios ya que se utilizó la clase ConexionesComputadoras para la resolución; lo anterior implicaba la lectura de todos los datos múltiples veces, por lo que el tiempo de ejecución era muy elevado. La manera de mejorar esto, es mediante la lectura directa de los datos, evitando emplear la clase de ConexionesComputadoras. De esa manera, la lectura de datos no se tiene que hacer tantas veces ya que la clase ConexionesComputadoras incluía la lectura de todos los datos por cada IP. La implementación de lo anterior en código, simplemente se itera en los datos en lugar de usar dicha clase (sí se realizó la implementación de esto y se presentarán ambas maneras de resolver la parte 4).

Otra solución con la posibilidad de mejora, es la parte 2. En este caso, es muy útil tener un “stack” y un “queue” dada la manera en la que se describe el comportamiento de las conexiones. A pesar de eso, es posible que otras interacciones se compliquen si se quisiera acceder a un elemento que no esté al final del “queue” o arriba del “stack” ya que se tienen que hacer “pop” para acceder. En ese caso, podría ser conveniente utilizar vectores en los que se guarden las conexiones entrantes o salientes.

Para finalizar, mi aprendizaje durante todo el semestre ha sido muy extenso y satisfactorio. Observar la manera en la que los temas vistos en clase se pueden aplicar en la realidad y pueden tener un uso tan relevante, me parece impresionante e invaluable.