Act 4.3. Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)

Alain Vicencio Arizabalo A01620758

Julian Andres Gonzalez Arguello A00831514

Programación de estructura de datos y algoritmos fundamentales

(Grupo 570)

ITESM

Lunes 6 de febrero del 2023

Reflexión Individual Act 4.3

Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales

Reflexión Julián:

Esta entrega de la situación problema involucró el almacenamiento de la bitácora en una estructura de datos tipo grafo. Los grafos son estructuras que permiten representar múltiples relaciones con nodos, siendo los elementos básicos de información, y arcos, siendo la liga que junta los diferentes nodos de un grafo. Para poder representar un grafo computacionalmente, se utilizan las listas de adyacencia, en las cuales cada vértice del grafo es una entrada en un vector, y cada arista son sus conexiones, que pueden ser implementadas con una lista ligada que vimos anteriormente.

Los grafos son sumamente importantes en situaciones relacionadas con las redes de botnets ya que su estructura permite la representación de las interacciones en una red de este estilo, y su ordenamiento facilita el análisis del sistema infectado. Si el sistema es infiltrado y se tiene la información de la bitácora ordenada en el grafo, en esta situación problema se podría localizar la IP del dispositivo infectado y las computadoras con conexiones directas a este. Esto facilitaría la mitigación del problema y el ataque en contra de el malware.

El código realizado en la entrega 4.3 igualmente permite localizar las 5 IPs con mayor grado de salida con la función gradosIPs() de complejidad O(n) al usar un tipo STL de mapa y un Max Heap para almacenar los datos. Igualmente, el código encuentra el camino más corto entre el Bootmaster, o el elemento con mayor grado de y el resto de los nodos del grafo. Esto fue hecho con la implementación del algoritmo Dijkstra con complejidad O(E Log V). Su complejidad hace al algoritmo muy eficiente en problemas de esta naturaleza. En la situación

problema, este algoritmo puede ser usado para encontrar el camino más corto entre el Bootmaster y un nodo infectado, ayudando a identificar el camino y facilitando la toma de decisiones para combatirlo de una manera más rápida.

Referencias

GeeksForGeeks. (2022). Dijkstra's Shortest Path Algorithm using priority_queue of STL Recuperado el 6 de febrero del 2023 de: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-using-priority_queue-stl/

Rodriguez, E. A. (s.f.). Grafos. Recuperado el 6 de febrero del 2023 de: https://experiencia21.tec.mx/courses/328618/files/123454078?wrap=1