Tecnológico de Monterrey, Campus Monterrey
Programación de estructuras de datos y algoritmos fundamentales
Rodrigo Llaguno Cárdenas, A01198067
Grupo 206

Reflexión Act 4.3

Para la entrega de la cuarta evidencia de nuestra actividad integral, teníamos que utilizar la estructura de datos de Grafos para almacenar la información de ataques cibernéticos en forma de Botnet por sus IP, donde el primero representa el nodo y el segundo IP representa las aristas o conexiones. Una vez almacenada la información se debe determinar el número de conexiones salientes, o grado, de cada nodo. Después de esto se debe analizar la información y responder la pregunta de ¿qué nodos tienen el mayor grado? Finalmente se debe concluir en base a este análisis cuál es presumiblemente el bot master, es decir, el nodo con el mayor grado.

Existen diferentes tipos de ciberataques pero el tipo con el cual estamos trabajando en esta evidencia es un botnet. Las botnets vienen de las palabras robot y network y sirven para describir dichos ataques los cuales son conectados por un malware y controlado por un propietario de los robots. Primero, estos robots infectan a las víctimas como computadoras mediante malware o incluso ingeniería social, término que describe la acción de infiltrarse usando técnicas sociales. Después se debe ampliar la botnet para infectar aún más otros equipos donde finalmente se activa la red. Con esto se pueden hacer desde acciones muy simples como leer y escribir datos de un sistema, enviar mensajes, instalar aplicaciones, recopilar información personal, hasta cosas más complejas como espiar en víctimas, crear ataques DDoS o hacer actividades de cripto minería (Belcic, 2021). Es importante identificar una botnet y poder tomar acción contra ella.

Los Grafos son una estructura de datos no lineal la cual consiste de dos elementos principales, nodos o vértices, el cual contiene la información fundamental y ejes, los cuales establecen las conexiones que existen entre nodos. Dichos ejes no están limitadas a solamente una conexión, esto quiere decir que un nodo puede tener n ejes. Existen diferentes maneras de representar un grafo. La primera se le conoce como representación de conjuntos donde se

define al grafo como un conjunto matemático estableciendo la relación entre nodo y vértices, por ejemplo:

$$V = \{V_1, V_2, V_3, V_4\}$$

$E = \{\{V_1, V_2\}, \{V_2, V_3\}, \{V_3, V_4\}, \{V_4, V_1\}\}$

Otro tipo de representación son las listas de adyacencia. De esta manera se muestra la información del grafo de manera que para cada nodo se especifica cuál nodo está conectado con la ayuda de un arreglo o lista enlazada auxiliar (GeeksForGeeks, 2022). Esta estructura de datos se utiliza principalmente para el mapeo de información de diferentes tipos como por ejemplo el calendario de aerolíneas, mapas de gps, resolver rompecabezas, networks, etc. Son útiles pues algunas de las cosas que puedes hacer ya con un grafo son algoritmos de problemas de flujo, de distancia más corta, de detección, etc. Existen algunos algoritmos principales pertenecientes a los grafos los cuales se describirán a continuación (Gupta, 2022):

- addEdge(), agrega un eje a un nodo donde se especifique el destinatario y en caso de ser necesario, el valor de dicho eje. Complejidad O(1)
- isTree(), verifica si el grafo es un árbol. Complejidad O(nodos+ejes)
- DFS(), recorre el grafo desde un nodo raíz del grafo hasta llegar al nodo más profundo de cada rama y luego se regresa. Complejidad O(nodos + ejes)
- BFS(), recorre el grafo empezando desde un nodo raíz y se desplaza nivel por nivel hasta recorrer el grafo por completo. Complejidad O(nodos+ejes)
- topoSort(), ordena los nodos de un grafo de manera que si existe un eje de *u* a *v* entonces *u* debe estar posicionado antes de *v*. Complejidad O(nodos + ejes)

Algunas ventajas de los grafos son que se puede organizar la información de una manera eficiente. Esto como consecuencia permite una visualización de datos también eficiente. Además esto permite la utilización de algoritmos los cuales permiten determinar los caminos más cortos, mínimos, vecinos de nodos, etc. Se debe tomar en consideración que también cuentan con algunas desventajas como la gran cantidad de complejidad de memoria que pueden llegar a ocupar o el hecho de que puede ser un poco más complicado de manejar una vez implementada (GeeksForGeeks, 2022).

En esta evidencia es de gran utilidad los grafos para el análisis de la botnet pues así podremos organizar la información que recibamos en cuanto al número de conexiones que se intentaron hacer para cada IP. Ya una vez con esta información almacenada, será muy fácil obtener información como el mauyor grado o en este caso el bot master.

Para la evidencia 4, decidimos implementar nuestro grafo utilizando 3 arreglos. Para el primer arreglo creamos una clase llamada Nodo la cual contenía un dato ip y otro accesos. Hicimos el primer arreglo de tipo Nodo con el largo de la cantidad de Nodos en la bitácora. El segundo arreglo era de tipo string para guardar el ip de origen y el tercer arreglo era de tipo string para guardar el ip de destino. Después, leímos la bitácora e insertamos los valores relevantes dentro de nuestro grafo. Al finalizar determinamos cuántos accesos tenía cada IP y ordenamos nuestro IP en base al número de accesos. Finalmente, escribimos los IP de mayor a menor accesos y en consola los top 5 y el bootmaster.

Referencias:

Belcic, I. (2021, October 8). *¿Qué es una botnet? | Buscador gratuito de botnets*. Avast. https://www.avast.com/es-es/c-botnet

GeeksForGeeks. (2022, May 25). *Applications, Advantages and Disadvantages of Graph*. GeeksforGeeks. Retrieved November 21, 2022, from https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-graph/

Gupta, S. (2022, May 12). *Time & Space Complexity of Graph Algo - 1*. Coding Ninjas. Retrieved November 21, 2022, from https://www.codingninjas.com/codestudio/library/time-space-complexity-of-graph-algo-1