

C++ Data structures to understand botnet

Infographic with the structures used (vectors, linked lists, trees, graphs and hash tables)

YOUR LOGO

Guerra de los bots: Ataque cibernético

Algoritmos fundamentales:
La realización de algoritmos de programación es un elemento común en la detección de patrones, resaltando tres operaciones que son el ordenamiento, búsqueda y mezcla. Además, el uso de estas técnicas en lenguajes de programación facilita diversas acciones, ya que por medio de una clasificación se pueden organizar los datos con ciertos criterios. Con el uso de estos algoritmos para ordenar nuestros datos y filtrar por fechas lo cual nos ayuda a identificar de mejor manera la fecha de los posibles ataques

Uso de listas ligadas:
Utilizamos una lista doblemente ligada determinando el orden a partir de la ip destino para así poder clasificarlas de menor a mayor y agruparlas, también nos ayuda a tener un registro de cómo fluye la información y ver si se tiene alguna anomalía en el instante que se está utilizando.

Árboles:
En esta situación con el uso de árboles utilizamos un heap con las ip destino determinando la prioridad por su número de accesos, pueden identificar los que atacan a nuestro sistema también, además podríamos analizar cuáles computadoras han accedido desde el momento donde se identificó una botnet, porque la infección tiene que empezar en cierto tiempo. Con esto encontramos la boot master: 62.231.113.60

Grafos:
Generamos un grafo dirigido con ip origen e ip destino donde la ip destino generaba arcos hacia los nodos de origen, ya que al hacer acciones en repetidas ocasiones es una actividad sospechosa por lo que logramos encontrar las direcciones ip con más salidas la cual fue: 172.30.142.38 nuestro primer infectado

Códigos hash:
Las tablas hash asocian llaves con valores, para nuestra situación utilizamos tablas hash para encontrar el número de accesos cercanos de una dirección ip en un rango de 30 segundos, utilizamos la dominio destino como llave mientras que el valor fue un par de datos: la hora y los accesos cercanos en el rango dicho anteriormente por lo que al analizar los datos de los dominios que recibieron más acceso cercanos obtenemos las posibles víctimas, forbes.com con 50 accesos cercanos fue víctima de un ataque DDOS. Con esta implementación comprobamos que la boot master es: zlpfkkkbaskodzalpmll.xxx ya que tiene un grado alto de entrada por lo que las redes se conectaban a pedir instrucciones constantemente

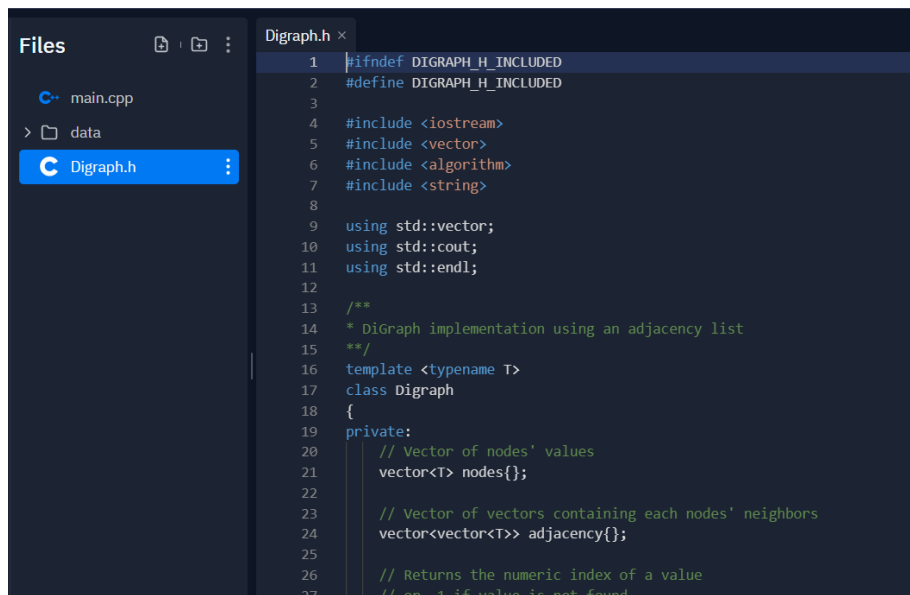
Alan Josué Melgar Fuentes A01752228
Jorge Isidro Blanco Martínez A01745907
José Luis Madrigal Sánchez A01745419

Trees gave us the boot master.

Hash Tables gave us the domain with more close accesses (equal or less than 30 seconds), which is the one who was attacked, like a DDOS.

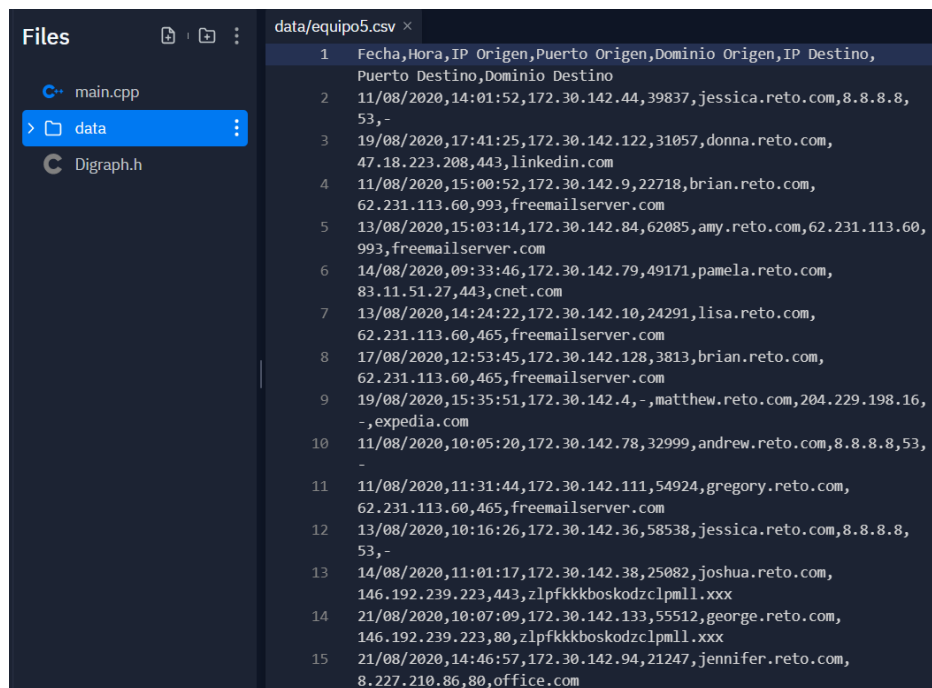
Implementation of Graph (the one which I liked the most and gave us the first infected host)

File of the Digraph Class



```
1 #ifndef DIGRAPH_H_INCLUDED
2 #define DIGRAPH_H_INCLUDED
3
4 #include <iostream>
5 #include <vector>
6 #include <algorithm>
7 #include <string>
8
9 using std::vector;
10 using std::cout;
11 using std::endl;
12
13 /**
14  * Digraph implementation using an adjacency list
15  */
16 template <typename T>
17 class Digraph
18 {
19 private:
20     // Vector of nodes' values
21     vector<T> nodes{};
22
23     // Vector of vectors containing each nodes' neighbors
24     vector<vector<T>> adjacency{};
25
26     // Returns the numeric index of a value
27     // or -1 if value is not found
```

File of the accesses(there are 33261 lines)



```
1 Fecha,Hora,IP Origen,Puerto Origen,Dominio Origen,IP Destino,
2 Puerto Destino,Dominio Destino
3 11/08/2020,14:01:52,172.30.142.44,39837,jessica.reto.com,8.8.8.8,
4 53,-
5 19/08/2020,17:41:25,172.30.142.122,31057,donna.reto.com,
6 47.18.223.208,443,linkedin.com
7 11/08/2020,15:00:52,172.30.142.9,22718,brian.reto.com,
8 62.231.113.60,993,freemailserver.com
9 13/08/2020,15:03:14,172.30.142.84,62085,amy.reto.com,62.231.113.60,
10 993,freemailserver.com
11 14/08/2020,09:33:46,172.30.142.79,49171,pamela.reto.com,
12 83.11.51.27,443,cnet.com
13 13/08/2020,14:24:22,172.30.142.10,24291,lisa.reto.com,
14 62.231.113.60,465,freemailserver.com
15 17/08/2020,12:53:45,172.30.142.128,3813,brian.reto.com,
16 62.231.113.60,465,freemailserver.com
17 19/08/2020,15:35:51,172.30.142.4,-,matthew.reto.com,204.229.198.16,
18 -,expedia.com
19 11/08/2020,10:05:20,172.30.142.78,32999,andrew.reto.com,8.8.8.8,53,
20 -
21 11/08/2020,11:31:44,172.30.142.111,54924,gregory.reto.com,
22 62.231.113.60,465,freemailserver.com
23 13/08/2020,10:16:26,172.30.142.36,58538,jessica.reto.com,8.8.8.8,
24 53,-
25 14/08/2020,11:01:17,172.30.142.38,25082,joshua.reto.com,
26 146.192.239.223,443,zlpfkkkboskodzclpml1.xxx
27 21/08/2020,10:07:09,172.30.142.133,55512,george.reto.com,
28 146.192.239.223,80,zlpfkkkboskodzclpml1.xxx
29 21/08/2020,14:46:57,172.30.142.94,21247,jennifer.reto.com,
30 8.227.210.86,80,office.com
```

Main file

```
main.cpp x
30     return os;
31 }
32
33 int main() {
34     // Leer archivo
35     vector<string> Ip_o{};
36     vector<string> Ip_d{};
37     string datos = "data/resultado";
38     ifstream archivo(datos);
39     string linea;
40     char delimitador = ' ';
41     char delimitador2 = '.';
42     getline(archivo, linea);
43     while (getline(archivo, linea))
44     {
45         stringstream stream(linea);
46         string Fecha, Hora, IP_Origen, Puerto_Origen,
            Dominio_Origen, IP_Destino, Puerto_Destino,
            Dominio_Destino;
47         getline(stream, Fecha, delimitador);
48         getline(stream, Hora, delimitador);
49         getline(stream, IP_Origen, delimitador);
50         getline(stream, Puerto_Origen, delimitador);
51         getline(stream, Dominio_Origen, delimitador);
52         getline(stream, IP_Destino, delimitador);
53         getline(stream, Puerto_Destino, delimitador);
54         getline(stream, Dominio_Destino, delimitador);
55
56         Ip_o.push_back(IP_Origen);
57         Ip_d.push_back(IP_Destino);
58     }
59     Digraph<string> dg{};
```

Running the code (IP adress node with the highest output grade)

```
❖ clang++-7 -pthread -std=c++17 -o main main.cpp
❖ ./main
La ip con el grado de salida mas alto es: 172.30.142.38
```