Ignacio Joaquin Moral
A01028470
En este caso, decidimos que la IP denominada A va a ser `172.26.113.22`, el host Destino con nombre anómalo, descubierto el entregable pasado, es `c8dzweh1e382v74a2pn5.xxx`, y el host Destino con conexiones entrantes anómalas, igualmente descubierto el reto pasado, es freemailserver.com
Para poder responder la pregunta número 1, primero generé un vector de grafos, precisamente para poder guardar los grafos diarios de conexiones. y un vector de fechas para poder guardar todas las fechas y automatizar el proceso de mostrado. Después, generé un header que me permita generar grafos de manera automática. Este grafo recibía los valores de un vector con todos los registros, y guardaba en cada grafo los valores en caso de que la fecha indique. Guardaba dos cosas principales, los vecinos de Entrada, y los vecinos de Salida. Los vecinos de Entrada eran guardados cuando el destino IP coincidía con el IP otorgado, y los vecinos de Salida eran guardados cuando el host IP coincidía con el IP otorgado. Esto era para que se puedan guardar los índices de los grafos, para que finalmente se puedan unir los ejes dependiendo si fuera de Entrada o Salida. Debido a que es un grafo dirigido, había una diferencia entre ellos.

```cpp
#include <iostream>
#include <vector>
#include <string>
#include "graph.hpp"
#include "read_csv_2.hpp"
#include "clase_archivo.hpp"
#include "reto_5.hpp"

int main(){
    std::vector<class Registros<std::string>> registros;
    registros = readRecords();
    std::vector<std::string> direccionesIP;
    for(int i=0; i<registros.size(); i++)
    {
        direccionesIP.push_back(registros.at(i).fuenteIP());
    }
    std::vector<int> idx;
    std::string ip, val, date;
    ip="172.26.113.";
    std::cout << "Escribe un numero del 1 al 150 para encontrar su IP:" << std::endl;
    std::cin >> val;
    ip.append(val);
    std::vector<std::string> dates;
    dates.push_back("10-8-2020");
    dates.push_back("11-8-2020");
    dates.push_back("12-8-2020");
    dates.push_back("13-8-2020");
    dates.push_back("14-8-2020");
    dates.push_back("15-8-2020");
    dates.push_back("16-8-2020");
    dates.push_back("17-8-2020");
    dates.push_back("18-8-2020");
    dates.push_back("19-8-2020");
    dates.push_back("20-8-2020");
    dates.push_back("21-8-2020");
```

Imagen 1.1, Guardado de fechas y selección de IP

```cpp
std::vector<class Graph<std::string>> internosPorDias;
for(int i=0; i<dates.size(); i++)
{
    std::vector<int> vecinosSalientes, vecinosEntrantes;
    Graph<std::string> internoPorDia;
    internoPorDia = checkGraphs(ip, idx, dates.at(i), i, vecinosEntrantes, vecinosSalientes);
    internosPorDias.push_back(internoPorDia);
    std::cout << "El dia " << dates.at(i) << " la computadora escogida tuvo estas conexiones: ";
    internosPorDias.at(i).BFS(idx.at(i));
    //Para la pregunta 2, de conexiones entrantes por dia
    // if(vecinosEntrantes.size()>0)
    // {
    //     std::cout  << "\t\tLas conexiones entrantes a esta ip en este dia fueron " << vecinosEntrantes.size() << std::endl;
    // }
}
std::cout << std::endl;
```

Imagen 1.2, Automatización de mostrado y guardado

```cpp
#ifndef RETO_5_HPP
#define RETO_5_HPP
#include <vector>
#include <iostream>
#include "graph.hpp"
#include "clase_archivo.hpp"
#include "read_csv_2.hpp"
Graph<std::string> checkGraphs(std::string ip, std::vector<int> &idx, std::string date, int index, std::vector<int> &vecinosSalientes, std::vector<int> &vecinosEntrantes)
{
    std::vector<class Registros<std::string>> registros;
    registros = readRecords();
    Graph<std::string> internoDia1;
    int first = internoDia1.add_node(ip);
    idx.push_back(first);
    int i=0;
    while(i<registros.size())
    {
        if (registros.at(i).date() != date)
        {
            i++;
        } else
        {
            while(i<registros.size() && registros.at(i).date()==date)
            {
                if(registros.at(i).fuenteIP()==ip)
                {
                    std::string vecinoSalida, dummy;
                    vecinoSalida=registros.at(i).destinoIP();
                    dummy=vecinoSalida;
                    if(dummy.size()>10 && dummy.erase(10, dummy.size()) == "172.26.113")
                    {
                        int idSalida=internoDia1.add_node(vecinoSalida);
                        vecinosSalientes.push_back(idSalida);
                    }
```

Imagen 1.3, función checkGraphs para los grafos diaros.

```cpp
                } else if (registros.at(i).destinoIP() == ip)
                {
                    std::string vecinoEntrada, dummy;
                    vecinoEntrada=registros.at(i).fuenteIP();
                    dummy=vecinoEntrada;
                    if(dummy.size()>10 && dummy.erase(10, dummy.size()) == "172.26.113")
                    {
                        int idEntrada=internoDia1.add_node(vecinoEntrada);
                        vecinosEntrantes.push_back(idEntrada);
                    }
                }
                i++;
            }
        }
    }
    for(int j=0; j<vecinosEntrantes.size(); j++)
        {
            internoDia1.add_edge(vecinosEntrantes.at(j), idx.at(index));
        }
    for(int k=0; k<vecinosSalientes.size(); k++)
        {
            internoDia1.add_edge(idx.at(index), vecinosSalientes.at(k));
        }
    std::cout << "Las conexiones salientes de esta compania fueron: " << vecinosSalientes.size() << std::endl;
    return internoDia1;
}
```

Imagen 1.4, continuación de checkGraphs

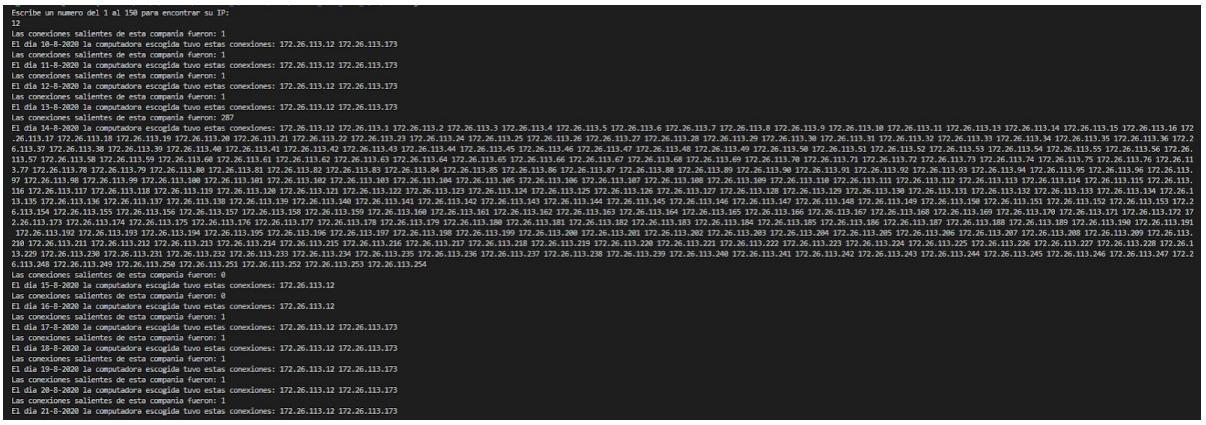

Imagen 1.5, Resultados

Evidentemente, la IP escogida es la que más conexiones salientes internas tuvo, pues tuvo 287 conexiones salientes internas el día 14 de agosto de 2020, como se puede observar en la imagen anterior.

Para la pregunta 2, se aprovechó que se envió los vectores de vecinosEntrada por referencia, y entonces, en el caso de que el tamaño fuera mayor a 0 se imprimían los valores.

```
if(vecinosEntrantes.size()>0)
{
    std::cout  << "\t\tLas conexiones entrantes a esta ip en este dia fueron " << vecinosEntrantes.size() << std::endl;
}
```

Imagen 2.1, Funcion para encontrar los IPs que se conectaron a la IP otorgada.

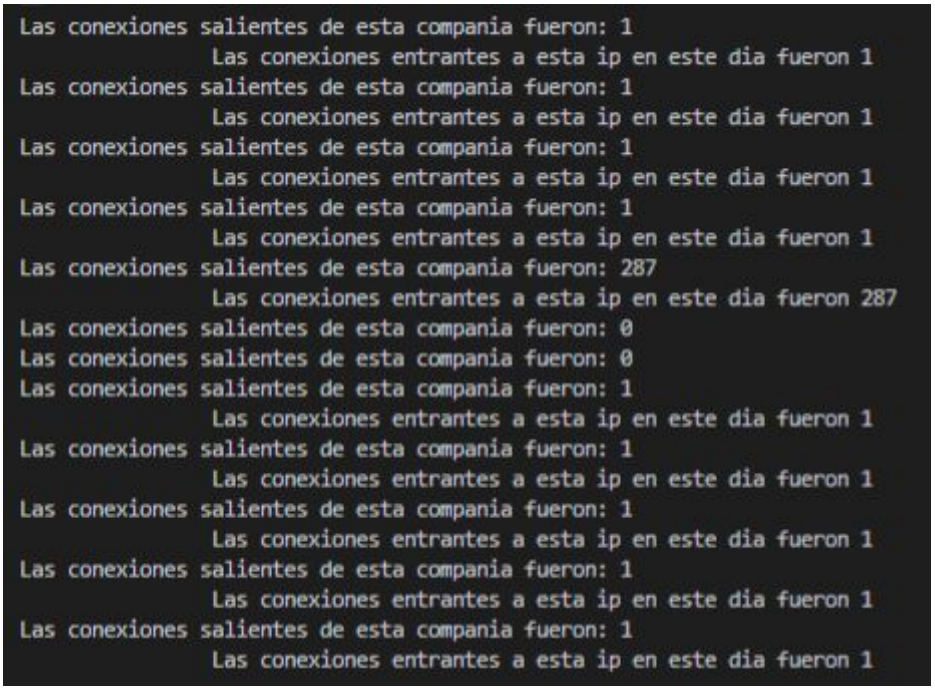

Imagen 2.2, Resultados.

Como se puede observar en la imagen anterior, solo se imprimen las conexiones entrantes en caso de que fueran mayor a 0. Ese “1” semi constante que aparece es una constante en todas las IPs, y posiblemente es la IP de algún superior. Entre la fecha 6 y 7 no se

registraron conexiones salientes o entrantes. Estas fechas representan el 15 y 16 de agosto, un fin de semana, donde probablemente no se trabajó.

Para la pregunta 3, se modificó un poco la función de checkGraphs para que revise los destinos Hosts, en lugar de las IPs, y de igual manera se eliminó el vector de vecinosSalida, pues las conexiones anómalas no se conectaban a otras IPs.

```
std::string ipAnomalo = "c8dzweh1e382v74a2pn5.xxx";
std::vector<class Graph<std::string>> hostsDestinos;
for(int i=0; i<dates.size(); i++)
{
    std::vector<int> vecinosEntrantes;
    Graph<std::string> hostsDestinoAnomalos;
    hostsDestinoAnomalos = checkGraphsHosts(ipAnomalo, idx, dates.at(i), i, vecinosEntrantes);
    hostsDestinos.push_back(hostsDestinoAnomalos);
    if(vecinosEntrantes.size()>0)
    {
        std::cout << "El dia " << dates.at(i) << " se conectaron " << vecinosEntrantes.size() << " computadoras al ip Anomalo." << std::endl;
    }
}
std::cout << std::endl;
```

```
Graph<std::string> checkGraphsHosts(std::string hostDestino, std::vector<int> &idx, std::string date, int index, std::vector<int> &vecinosEntrantes)
{
    std::vector<class Registros<std::string>> registros;
    registros = readRecords();
    Graph<std::string> internoDia1;
    int first = internoDia1.add_node(hostDestino);
    idx.push_back(first);
    int i=0;
    while(i<registros.size())
    {
        if (registros.at(i).date() != date)
        {
            i++;
        } else
        {
            while(i<registros.size() && registros.at(i).date()==date)
            {
                if (registros.at(i).destinoHost() == hostDestino)
                {
                    std::string vecinoEntrada, dummy;
                    vecinoEntrada=registros.at(i).fuenteIP();
                    dummy=vecinoEntrada;
                    if(dummy.size()>10 && dummy.erase(10, dummy.size()) == "172.26.113")
                    {
                        int idEntrada=internoDia1.add_node(vecinoEntrada);
                        vecinosEntrantes.push_back(idEntrada);
                    }
                }
                i++;
            }
        }
    }
    for(int j=0; j<vecinosEntrantes.size(); j++)
        {
            internoDia1.add_edge(vecinosEntrantes.at(j), idx.at(index));
        }
    return internoDia1;
}
```

```
El dia 14-8-2020 se conectaron 350 computadoras al ip Anomalo.
El dia 17-8-2020 se conectaron 612 computadoras al ip Anomalo.
El dia 18-8-2020 se conectaron 599 computadoras al ip Anomalo.
El dia 19-8-2020 se conectaron 614 computadoras al ip Anomalo.
El dia 20-8-2020 se conectaron 597 computadoras al ip Anomalo.
El dia 21-8-2020 se conectaron 571 computadoras al ip Anomalo.
```

Como se ve en la imagen anterior, se conectaron a partir del 14, con excepción del 15 y 16, al host anómalo.

Para la pregunta 4, utilizando el mismo grafo anterior, con el cambio de que ahora buscaban las conexiones al ip con número de conexiones anómalas.

```cpp
std::cout << std::endl;
std::string ipTranscurrido = "freemailserver.com";
std::vector<class Graph<std::string>> hostsDestinosTranscurridos;
for(int i=0; i<dates.size(); i++)
{
    std::vector<int> vecinosEntrantes;
    Graph<std::string> hostsDestinoTranscurrido;
    hostsDestinoTranscurrido = checkGraphsHosts(ipTranscurrido, idx, dates.at(i), i, vecinosEntrantes);
    hostsDestinosTranscurridos.push_back(hostsDestinoTranscurrido);
    if(vecinosEntrantes.size()>0)
    {
        std::cout << "El dia " << dates.at(i) << " se conectaron " << vecinosEntrantes.size() << " computadoras al ip Transcurrido." << std::endl;
    }
}
```

4.1, Función para buscar IP transcurrido.

```
El dia 10-8-2020 se conectaron 1373 computadoras al ip Transcurrido.
El dia 11-8-2020 se conectaron 1428 computadoras al ip Transcurrido.
El dia 12-8-2020 se conectaron 1432 computadoras al ip Transcurrido.
El dia 13-8-2020 se conectaron 1335 computadoras al ip Transcurrido.
El dia 14-8-2020 se conectaron 1105 computadoras al ip Transcurrido.
El dia 17-8-2020 se conectaron 1091 computadoras al ip Transcurrido.
El dia 18-8-2020 se conectaron 1146 computadoras al ip Transcurrido.
El dia 19-8-2020 se conectaron 1128 computadoras al ip Transcurrido.
El dia 20-8-2020 se conectaron 1055 computadoras al ip Transcurrido.
El dia 21-8-2020 se conectaron 1078 computadoras al ip Transcurrido.
```

4.2, Resultados Pregunta 4

Como se puede observar, efectivamente hubo muchas conexiones al IP Transcurrido.

Para la pregunta 5, se hicieron unas preguntas de investigación.
Un ping sweep es un ataque que envía peticiones de eco ICMP, "pings", a un rango de direcciones IP, con el objetivo de encontrar anfitriones que se pueden probar en busca de vulnerabilidades. Podría ser lo que está ocurriendo antes del 14 de agosto, pues el IP 172.26.113.173, pues siempre se conectan a este.

Un DDoS es un ataque generado por bots denominado “ataque distribuido denegación de servicio”, esto ocurre cuando un servidor envía varias solicitudes al recurso web atacado, con la intención de desbordar la capacidad del sitio web para administrar varias solicitudes y de evitar que este funcione correctamente. Esto parece ocurrir el 14 de agosto, cuando al IP seleccionado se conectaron 287 computadoras y se conectó igualmente a 287 IPS. Esto podría ocurrir por el servidor anormal al que se empezaron a conectar en la misma fecha.

Un servidor de comando y control es un elemento que se utiliza para intentar controlar los malware. Esto podría ser el mismo IP utilizado, pues aunque las conexiones al host anómalo siguen, pero no ocurren tantas conexiones a otros IPs una vez pasado el fin de semana.

Un botmaster es una persona encargada de mantener los bots de una empresa. Es probable que freemailserver.com sea el botmaster, pues se encarga de la comunicación.