**Giải thích quá trình thực hiện phát hiện DDos bằng bộ mã lệnh network\_traffic\_detection.py**

- Sau khi dùng mã lệnh (sudo python3 network\_traffic\_detection.py) thì trong đoạn mã sẽ thực hiện phần def main() sau khi đã import các thư viện cần thiết

* Trong hàm main() bắt đầu với quá trình kiểm tra thông tin file đầu vào có hợp lệ hay không bằng cách dùng hàm check\_input(), trong hàm check\_input() bắt đầu kiểm tra với độ dài thành phần tính sau từ python3 với lần kiểm tra đầu tiên bằng điều kiện "len(sys.argv) != 3" - độ dài thành phần của lệnh nhập có khác 3 hay không?

***Giải thích về độ dài thành phần:***

***Trong python để kiểm tra thành phần trong cấu trúc thì người ta sẽ dùng hàm sys.argv là một mảng lưu trữ các thành phần trong một cấu trúc***

***Mô phỏng:***



***Như trên hình thì sudo python3 là một lệnh thục hiện còn phần còn lại là là đối tượng truy vấn thì nó được gọi là chuỗi truy vấn với chuỗi truy vấn thì trong đó chứa các thành phần được ngăn cách nhau bằng dấu cách. Và mỗi mảng của sys.argv sẽ chứa mỗi thành phần đó.***

***Như ví dụ trên thì argv sẽ có 3 ngăn mảng là sys.argv[0] sẽ chứa chuỗi “network\_traffic\_detection.py”, sys.argv[1] sẽ chứa chuỗi “ens33” và cuối cùng sys.argv[2] sẽ chứa chuỗi “20”. Và lệnh sys.argv dùng để trả về một mạng chứa các thành phần cấu trúc trên.***

* + Nếu điều kiên trên đúng thì sẽ báo lại cú pháp chỉnh sửa là python3 " + sys.argv[0] + " <NIC> <NUMBER\_PACKET\_CAPTURING> và buộc người dùng thoát tệp code và thực hiện lại
* Nếu sai tiếp tục sẽ thực hiện điều kiện ngược lại "len(sys.argv) == 3" - độ dài thành phần của lệnh nhập có bằng 3 không
  + Nếu đúng tiếp tục kiểm tra vị trí của tệp tin đường truyển mạng (ens33) liệu tệp tin đó đang không có ở vị trí '/sys/class/net/' hay không?
    - Nếu đúng thì báo lỗi cổng đường truyền mạng không tìm thấy - ERROR: Interface \'" + sys.argv[1] + "\' not found!!! Và thoát chương trình
    - Ngược lại thì trả về giá trị 1 – đúng tiếp tục chương trình
* Quay lại với hàm main tiếp tục tạo chương trình tạo biến model\_LR và gán giá trị bằng lệnh chạy mô hình – joblib.load('RandomForest model.joblib'). Sauk hi mô hình chạy xong thì sẽ thông báo ra màn hình
* Tiếp tục tạo một biến i gán giá trị bằng 0 để đánh số cho các trong thư mục PCAP

Chú thích PCAP là file lưu trữ thông tin lưu lượng mạng bắt được bằng một phần mềm nào đó

* Tạo vòng lập liên tục và trong đó có những lệnh sau:
  + Tạo biến nic để lưu trữ cổng truy cập mạng
  + Num\_cap để lưu trữ số lượng gói tin được bắt
  + pcap\_file để gán chuỗi tên của .pcap

Chú thích cú pháp đặt tên PCAP file với cú pháp 'capture\_' + str(i) + '.pcap'

Vd: capture\_0.pcap

'capture\_' 🡪 capture\_

str(i) 🡪 0 : chuyển đổi giá trị i thành chuỗi

'.pcap' 🡪 .pcap

* + Tạo một dòng lệnh shell 1 bằng chuỗi sau:

command1 = "sudo tcpdump -c " + num\_cap + " -i " + nic + " -w PCAP/" + pcap\_file

*Giải thích:*

*command1: biến để gán chuỗi lệnh shell*

* *Từng thành phần trong chuỗi shell*

*sudo để dùng quyền cao nhất*

*tcpdump dùng để trích xuất lưu lượng trên đường mạng*

*-c num\_cap để đếm tổng số lượng là num\_cap gói tin*

*-i nic để bắt gói tin từ cổng mạng nic nào*

*-w PCAP/pcap\_file tổng hợp thông tin trên vào ghi dữ liệu vào tệp tin có địa chỉ PCAP/pcap\_file*

Sau khi được tiến hành chạy thì dòng lệnh nguyên vẹn trên shell là

sudo tcpdump -c 20 -i ens33 -w PCAP/capture\_0.pcap

Có nghĩa bắt 20 gói tin từ cổng mạng ens33 và ghi thông tin bắt được lên tập tin ở PCAP/ capture\_0.pcap

* + Tạo biến tcpdump để bắt đầu chạy bằng lệnh sp.run([command1], shell=True)
  + Tạo một dòng lệnh shell 2 bằng chuỗi sau:

argus -r PCAP/" + pcap\_file + " -w - -F argus.conf | ra -F ra.conf -L0 -c , -n > traffic.binetflow"

*Giải thích*

*argus dùng để kiểm toán lại bản ghi*

*-r dùng bản ghi tcpdump hoặc bản lưu trữ lưu lượng mạng (PCAP)*

*-w - -F argus.conf | ra -F ra.conf -L0 -c , -n > traffic.binetflow ghi sự chỉnh sửa của file pcap theo cấu hình của file argus.conf và ra.conf vào trong file traffic.binetflow*

*-F argus.conf sử dụng config file (file đuôi \*.conf)*

* + Tiếp tục như command1 thì ta dùng

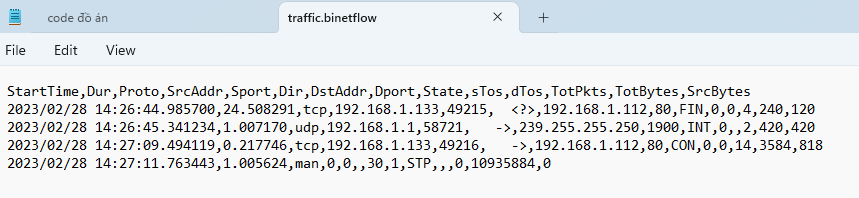
argus\_ra = sp.run([command2], shell=True) - để khởi chạy câu lệnh trên

* + Lệnh cat\_binetflow = sp.run(["cat traffic.binetflow"], shell=True)

Dùng để đọc các thông số bắt được trên đường mạng và bao gồm các thông số sau:

Start Time(Thời gian bắt đầu gói tin), Duration(Khoảng thời gian di chuyển), Protocol, Source Address, Sport, Direct,DstAddr,Dport,State,sTos,dTos,TotPkts,TotBytes,SrcBytes

Tất cả thông số đó được nằm trong tệp tin traffic.binetflow, và dưới là hình ảnh cho biết nội dung bên trong traffic.binetflow



Mỗi giá trị bên dưới được ngăn cách bởi nhau bằng dấu phẩy.

Và lệnh trên dùng lệnh cat để đưa thông tin lưu trong traffic.binetflow ra ngoài màn hình terminal

* + Chuỗi lệnh đọc và ghi file nhật ký

f = open('traffic.binetflow', 'r')

chuoi = f.read()

ff = open('log.txt', 'a')

ff.write(chuoi)

f = mở file traffic.binetflow với quyền là chỉ định quyền đọc ‘r’

chuoi = đưa dữ liệu của file traffic.binetflow vào biến chuoi

ff = mở file log.txt với tất cả quyền ‘a’

ff.ghi dữ liệu từ biến chuoi vào trong file log.txt

* + Lệnh đọc file bằng pd.read.csv()

df = pd.read\_csv('traffic.binetflow')

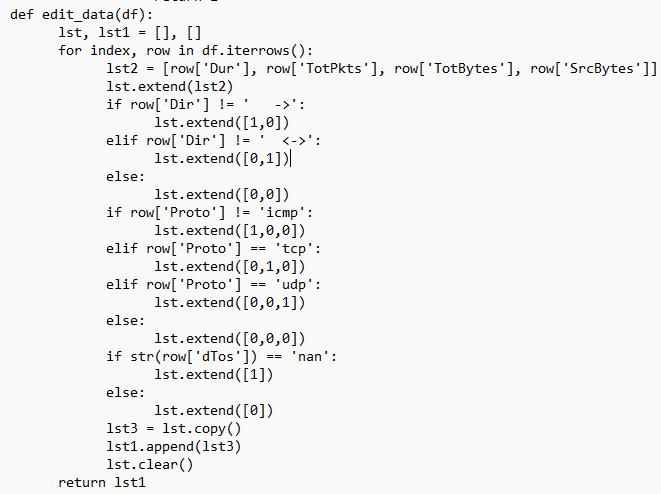
Dùng để đọc dữ liệu theo dạng csv của file traffic.binetflow. CSV là một dạng file thống kê dạng excel để để dễ dàng chuyển đổi thông tin ở dạng thô dữ liệu.

* + Lệnh chỉnh sửa dữ liệu

df = df.drop(['StartTime', 'SrcAddr', 'Sport','DstAddr','Dport','State','sTos'], axis=1)

df\_new = edit\_data(df)

* df với dữ liệu bỏ đi các cột không cần thiết bao gồm: các thành phần trong ngoặc 'StartTime', 'SrcAddr', 'Sport','DstAddr','Dport','State','sTos'. Và lưu nó vào trong biến df\_new
* Trước khi lưu thì dữ liệu df dược sàn lọc chỉnh sửa qua bằng hàm edit\_data
  + Trong hàm edit\_data được giải thích như sau:



* + - Ban đầu tạo ra 2 mảng lst, lst1
    - Típ theo cho index đi dò từng cột df.iterrow() – dò cột của đối tượng df
    - Cho các cột row['Dur'], row['TotPkts'], row['TotBytes'], row['SrcBytes'] lưu vào trong mảng lst2
    - Cho mảng 0 (lst) mở rộng với lst 2 bằng lệnh lst.extend(lst2)

Ví dụ về hàm extend:

Với lst có chứa cái giá trị như [1,2,3] và lst2 chứa [4,5,6]. Thì khi dùng lst.extend(lst2)

Thì python sẽ đứa lst2 vào lst thì ta có được một mảng như thế này [1,2,3,4,5,6]

Đối với trường hợp trên thì lst là 1 mảng ko chứa phần tử và còn lst2 chứa các cột trong lst2

Sau khi lst.extend(lst2) thì kết quả nó sẽ ra 1 mảng như sau

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Duration | |  | | |  | | --- | | TotPkts | |  | | |  | | --- | | TotBytes | |  | | |  | | --- | | SrcBytes | |  | |

* + - Sau khi qua hết các giai đoạn xét điều kiện sẽ đến bước lượt sơ bằng cách sau:

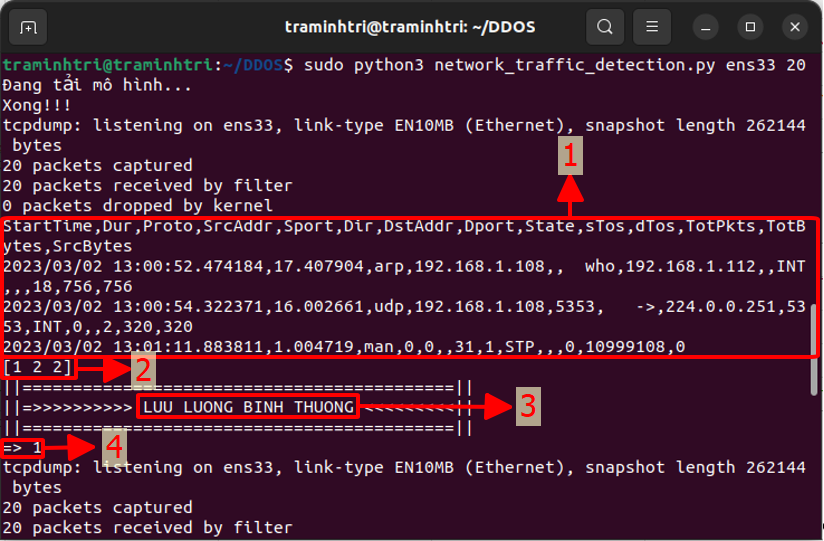
lst3 = lst.copy() – sao chép dữ liệu từ list qua lst3

lst1.append(lst3) – đưa list 3 vào list 1 nhưng đưa toàn vẹn dữ liệu

lst.clear() – xóa sạch dữ liệu trong lst

return lst1 – trả về dữ liệu lst1

* + Sau khi hoàn tất công việc chỉnh sửa dữ liệu thì bộ mã sẽ dùng mô hình dựng sẵn phân tích bộ dữ liệu sau khi được chỉnh sửa và cuối cùng lưu giá trị trả về vào biến y\_predict



Là cái số 2 trong hình hình trên ấy

* + Từ kết quả thông qua mô hình ta sẽ xử lý thêm bước để đưa ra thông báo

for j in y\_predict, trước tiên tạo một biến s1=0:

if j == 1:

s1 = s1 + 1

* + - Cho j đi trong mảng y\_predict vd mảng trên [1 2 2]
    - Xét điều kiện nếu như giá trị của biến chạy (j) == 1
    - Thì s1 ta + thêm 1

Với ví dụ mảng [1 2 2] thì sẽ được kết quả vòng chạy là 1

Đi hết vòng lập thì ta có kết quả xét:

* + Nếu kết quả >= độ dài của mảng y\_predict -1 thì lúc này lưu lượng mạng sẽ là bất thường
  + Còn ngược lại thì lưu lượng mạng bình thường

Với ví dụ trên thì độ dài của y\_predict là 3 vì có 3 phần tử, tiếp tục -1 là còn 2 thế là 1>=2 là sai thì ta được kết quả là lưu lượng mạng bình thường