

Network System Capstone Final Project

Network monitoring and intrusion detection system Report

1. Introduction

本期末專題的靈感源自我在進行資工專題時的經驗。當時我覺得Wireshark雖然提供了詳盡的封包資訊，但對於新手來說界面有點複雜，不太友善。且有些使用者可能只是想要了解封包基本的內容，過於詳細的資訊反而會導致他們的困擾。因此，我希望開發一個簡單的命令行網路監測工具，可以顯示封包基本的資訊。同時，如果使用者需要更詳細的分析，可以使用專題程式產生的PCAP檔案，再透過Wireshark打開來查看詳細資料，來達到一個程式可以滿足不同程度需求的目標。

在此專題中，我還加入了網路入侵檢測的功能。我的想法是，既然Wireshark都能夠那麼詳細的分析封包了，那麼應該也可以利用它來檢測網路入侵吧？因此，我預計在預計的功能是，我會先預先訓練一個入侵檢測模型。使用者可以輸入他們的封包檔案（以PCAP格式為基礎的CSV檔），然後專題程式將分析數據並計算出網路入侵的機率，以百分比的形式呈現給使用者。這樣的功能將有助於提高網路安全性，讓使用者更容易識別和應對潛在的入侵行為，降低資安疑慮。

2. Functionality

我的期末專題預計包含以下功能，以下我將一一介紹它們：

1. 封包監視：我的程式將提供封包監視功能，可以顯示封包的基本資訊，例如日期時間、封包的起點和目的地，以及所使用的協議類型。這將讓使用者能夠快速瞭解網路流量的概況。
2. 自訂過濾：我們的程式還將提供自訂監控的選項。使用者可以使用預設的監控參數，也自行設定監控的時間長度、網路介面和協議類型，以便根據自己的需求進行網路監測。
3. 檔案儲存和轉換：我的程式將支援監視資料的儲存功能。使用者可以將監測到的資料儲存為PCAP格式，也可以選擇儲存為CSV格式。此外，程式還允許使用者將現有的PCAP檔案轉換為CSV格式，以便進行進一步的分析和處理。
4. 網路入侵偵測：我的程式將提供網路入侵偵測的功能。使用者可以輸入CSV檔案，並使用預先訓練的入侵偵測模型來判斷監控期間是否存在網路入侵的

可能性。這將有助於提高網路安全性，並讓使用者能夠及時識別和應對潛在的入侵行為。

透過這些功能，我的網路監測和入侵偵測系統將能夠提供全面的封包監視、自訂過濾、檔案儲存和轉換，以及網路入侵偵測等功能，從而幫助使用者有效地管理和保護他們的網路環境。

3. Experiment / Development Environment

- 開發語言: Python 3.10
- 作業系統: Ubuntu 22.04
- 網路封包抓取函式庫: Pyshark
- 機器學習函式庫: Scikit-learn

4. Details

網路監測和網路入侵偵測的功能我寫在同一隻程式裡面，我們現在先來看一下功能：

```
~/Desktop/nscap/final
> python3 capture.py -h
usage: capture.py [-h] [-t TIME] [-i INTERFACE] [-f BPF_FILTER] [-d DETECTION]
                  [-p PARSE] [--csv]

options:
  -h, --help            show this help message and exit
  -t TIME, --time TIME  capture time(secs), default is 30 secs
  -i INTERFACE, --interface INTERFACE
                        capture interface, default is lo
  -f BPF_FILTER, --bpf_filter BPF_FILTER
                        packet bpf filter, default is None
  -d DETECTION, --detection DETECTION
                        ddos detection, analyze .pcap file
  -p PARSE, --parse PARSE
                        parse .pcap to .csv file with ddos required arguments
  --csv                also output .csv file with ddos required arguments
```

如上圖所示，我的程式提供了自訂監控選項的功能，-t 可指定監控的時間長短，-i 可以指定監控的網路界面，-f 可以設定要擷取的封包的協議類型，程式也有提供檔案轉換的功能，-p 可以把PCAP檔案轉換為intrusion detection用的csv檔案，當執行程式後，程式就開始在指定的網路界面上擷取封包，沒有特別給參數的話就是一切監控選項都為預設值，執行結束後會產生PCAP檔案供使用者儲存，或是也可以開始時加上--csv選項，這樣結束時就會同時產生intrusion detection用的csv檔案。

接下來我們分為兩部份來詳細講解程式的功能：

a. 網路監控 (Network Monitoring)

首先是網路監測的部份，程式的部份是使用Python的PyShark來實做的，PyShark是一個基於Python的封包分析庫，它提供了Wireshark功能的封裝和使用。透過PyShark，使用者可以以簡潔且易於使用的方式進行封包分析和網路流量監測。它允許使用者對封包進行篩選、監控和分析。PyShark的優勢在於它的易用性和豐富的功能，使得開發人員和網路專業人士能夠更有效地進行封包分析和網路故障排除。

當執行程式後，程式就開始依照給定的監控參數擷取封包：

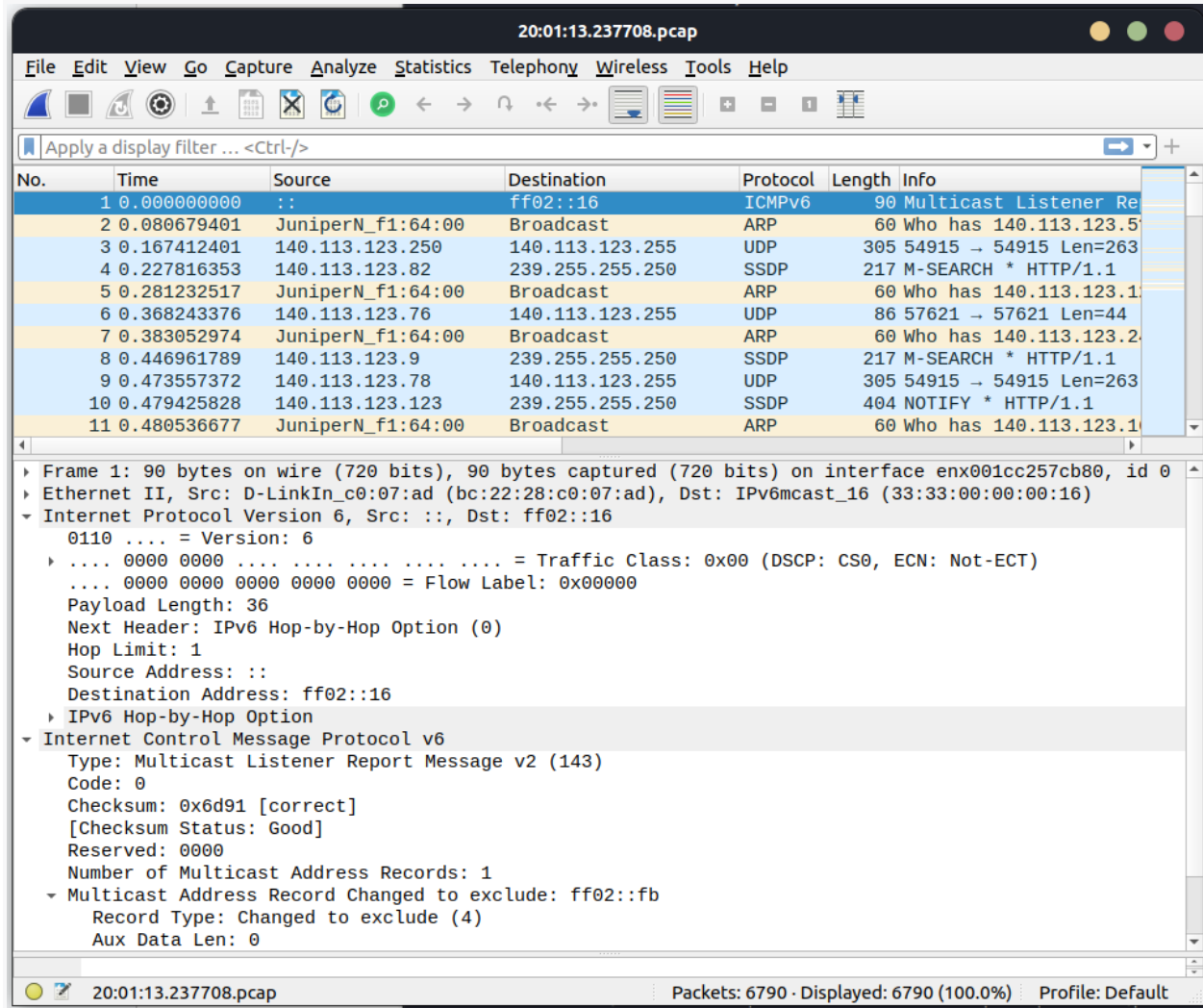
```
UDP SSDP [<ETH Layer>, <IP Layer>, <UDP Layer>, <SSDP Layer>]
Source: 140.113.123.227, 65350
Destination: 239.255.255.250, 1900
Date and Time: 2023-06-07 19:10:25.103689

UDP MDNS [<ETH Layer>, <IP Layer>, <UDP Layer>, <MDNS Layer>]
Source: 140.113.123.32, 45941
Destination: 224.0.0.251, 5353
Date and Time: 2023-06-07 19:10:25.182692

UDP MDNS [<ETH Layer>, <IP Layer>, <UDP Layer>, <MDNS Layer>]
Source: 140.113.123.32, 44439
Destination: 224.0.0.251, 5353
Date and Time: 2023-06-07 19:10:25.198573
```

我們可以從上圖中看到，程式會顯示最基本的封包資訊，包含日期與時間、封包起點和目的地的IP和Port，封包協議和結構結構，而我的程式能夠解構的封包類型有TCP，UDP，ICMP，和ARP這幾種，不論是IPv4又或是IPv6都支援，剩餘的封包種類，程式中只會顯示是哪個協議的封包有進入，但基本資料的部份就不會顯示。這些類型的封包資訊對於初學者而言就已經非常足夠了，除了可以一目了然封包的基本資訊外，也可以對封包的結構有初步的了解。

在擷取時間到了後，程式會將這段時間的封包資料儲存成一個pcap檔案，如果有需要更詳細封包資料的使用者，就可以用Wireshark將這個檔案打開來獲取它需要的資訊：



如果當初有指定也要輸出csv檔的話，那程式結束時也會一併產生可用於我的intrusion detection model的csv檔：

```
2023-06-07 20:01:33,427 - LiveCapture - DEBUG - Cleanup Subprocess (pid 13471)
output pcap to /home/iammrchen/Desktop/nsicap/final/static/20:01:13.237708.pcap
./csv/20:01:13.237708.csv
```

b. Network Intrusion Detection

我的程式提供了可以分析使用者輸入的CSV檔案，分析後會輸出預測有 Network Intrusion的機率，由於Network Intrusion的種類眾多，所以本次期末專題僅僅實做DDoS的部份。

程式中用來判定檔案擷取時間內有沒有DDoS的情況的標準是我預先Train好的Intrusion Detection Model，而Train的Dataset的來源為University of New

Brunswick所提供的Intrusion Detection Dataset, 其中就有他們模擬DDoS時Wireshark所紀錄的PCAP檔案, 還有他們整理過後關於封包資料的CSV檔案, [網頁](#)中都標的十分清楚。我Train Model所使用的csv是使用的們提供的pcap檔案, 從檔案中將網站上標注他們做DDoS時間段的封包提取出來, 我再去處理資料(填掉NaN, Infinity的欄位)並加上label後(benign, ddos)後形成的csv檔案, 不直接使用它所提供的CSV檔案是因為它沒有提供CSV上的features的計算方法, 所以我無法自行從PCAP檔中提取出他的CSV檔中的所有features, 因此我才自己做Data Processing and Labeling。

Afternoon

DDoS LOIT (15:56 – 16:16)












Attackers: Three Win 8.1, 205.174.165.69 - 71

Victim: Ubuntu16, 205.174.165.68 (Local IP: 192.168.10.50)

NAT Process on Firewall:

Attackers: 205.174.165.69, 70, 71 -> 205.174.165.80 (Valid IP of the Firewall) -> 172.16.0.1

Index of /CICDataset/CIC-IDS-2017/Dataset/PCAPs

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 Friday-WorkingHours.md5	2019-09-10 11:22	59	
 Friday-WorkingHours.pcap	2019-09-10 17:04	8.2G	
 Monday-WorkingHours.md5	2019-09-10 11:22	59	
 Monday-WorkingHours.pcap	2019-09-10 18:06	10G	
 Thursday-WorkingHours.md5	2019-09-10 11:24	61	
 Thursday-WorkingHours.pcap	2019-09-10 16:46	7.7G	
 Tuesday-WorkingHours.md5	2019-09-10 11:24	60	
 Tuesday-WorkingHours.pcap	2019-09-10 18:18	10G	
 Wednesday-WorkingHours.md5	2019-09-10 11:23	62	
 Wednesday-WorkingHours.pcap	2019-09-10 20:33	12G	

[illegible]

提取出來CSV檔後，我們就開始做Model的Training，我選擇用Python的Scikit-Learn套件來實做，選用的方法是Random Forest Classifier加上K-Neighbors Classifier來訓練出我的模型，RandomForestClassifier用於分析資料，決定出哪些features是最重要的，接下來再用資料中那些features丟入K-NeighborsClassifier來做模型的訓練：

```
rfc = RandomForestClassifier(verbose=True)
rfe = RFE(rfc, n_features_to_select=10)
print('randomForestClassifying.....')
rfe = rfe.fit(X_train, Y_train)

filename = f'model/my_RFE_model.pkl'
pickle.dump(rfe, open(filename, 'wb'))
feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), X_train.columns)]
selected_features = [v for i, v in feature_map if i==True]
# print(selected_features)
# print('-----')
X_train = X_train[selected_features]
X_test = X_test[selected_features]
# print('end')

scale = StandardScaler()
X_train = scale.fit_transform(X_train)
X_test = scale.fit_transform(X_test)

KNN_model = KNeighborsClassifier()
KNN_model.fit(X_train, Y_train)
```

Random Forest Classifier是一種基於Decision Tree的機器學習模型。它通常用於分類和回歸問題。隨機森林由多個決策樹組成，每棵樹都在隨機選擇的子樣本上進行訓練，並根據投票或平均的方式進行預測。隨機森林能夠處理大量的特徵和樣本，具有良好的泛化能力和抗過擬合能力。它還能夠計算出特徵的重要性，這對於理解數據中哪些特徵對於預測結果更為關鍵是非常有用的。

K-Neighbors Classifier是一種基於實例的機器學習模型。它基於訓練數據中的鄰近樣本進行分類。當需要對一個新的未標籤樣本進行分類時，KNN會比較該樣本與訓練數據中各個樣本的距離，並根據最接近的K個鄰近樣本的類別進行投票或加權平均來進行預測。KNN簡單而直觀，且在處理非線性問題和多類別問題時效果良好。它具有自適應能力，能夠適應不同的數據分佈和特徵空間。

綜合使用Random Forest Classifier和K-Neighbors Classifier來訓練模型的做法可以充分利用它們各自的優勢。Random Forest Classifier可以分析資料，確定重要的特徵，提供初步的預測能力。然後，這些重要的特徵將被提供給K-Neighbors Classifier進一步訓練，以提高模型的預測準確性。這種結合兩個模型的方法能夠有效地處理複雜的分類問題並產生具有較高準確性的結果。

訓練好後分別把Random Forest Classifier和K-Neighbors Classifier存成pickle檔，這樣到時候驗證使用者丟入的CSV有DDoS的機率時，就可以用這兩個預先訓練好的Model來判斷機率：

```
with open('model/my_RFE_model.pkl', 'rb') as file:
    rfe = pickle.load(file)
    feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), X_test.columns)]
    selected_features = [v for i, v in feature_map if i==True]
    print(selected_features)
    X_test = X_test[selected_features]
filename = 'model/my_KNN_model_0.9997728821258233.pkl'
with open(filename, 'rb') as file:
    KNN_model = pickle.load(file)
    prediction = KNN_model.predict(X_test)
    ddos_num = 0
    for result in prediction:
        if result == 1:
            ddos_num += 1
    print(prediction)
    print(f"DDoS Likelyhood: {ddos_num/len(prediction)}")
```

使用者丟入我們程式轉換好的CSV檔後，在加上參數-d後，程式就開始打開檔案並運算可能的百分比：

```
label encoding for test data.....
['ip.src', 'ip.dst', 'ip.flags.df', 'ip.fragments', 'ip.ttl', 'tcp.window_size', 'tcp.ack', 'tcp.seq', 'tcp.stream', 'frame.time_relative']
[1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0
1 0 1 0 0 0 0 0 1 0 1 0 1 0 1 1 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 1 0
0 0 0 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 1 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0
0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0
1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 0 0 0
0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0
0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0
0 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1]
DDoS Likelyhood: 0.28940568475452194
```

5. Conclusion

網路總整是一個重要且廣泛應用的領域，在本次專題中，我致力於開發一個網路監測和入侵檢測系統。我的目標是提供一個使用友好且功能強大的工具，讓使用者能夠有效地監控網路流量、識別入侵行為並保護網路安全。

首先，我介紹了專題的背景和動機，即Wireshark作為一個強大的封包分析工具，但對於新手來說可能有些複雜，因此我希望開發一個簡單且易於使用的命令行網路監測工具。透過封包監視功能，使用者能夠查看封包的基本資訊，如時間、封包的起點和目的地、協議類型，以獲取整體的網路流量概況。

為了提供更多自訂化的監控選項，我引入了自訂過濾功能。使用者可以根據自己的需求設定監控的時間長度、網路界面和協議類型，從而更精確地監測特定區域或特定類型的網路流量。

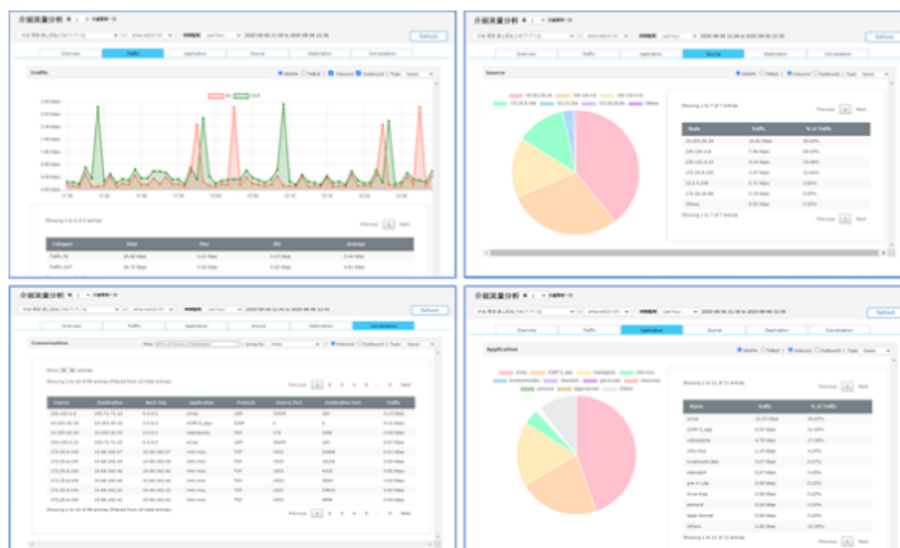
另一項重要功能是檔案儲存和轉換。我的程式允許使用者將監控的資料儲存為PCAP或CSV格式，這樣使用者可以根據自己的需求進一步分析和處理監測數據。同時，使用者還可以將現有的PCAP檔案轉換為CSV格式，提供更靈活的數據處理選項。

在入侵檢測方面，我訓練了一個結合Random Forest Classifier和K-Neighbors Classifier的模型。Random Forest Classifier用於分析資料，確定重要的特徵，提供初步的預測能力。然後，這些重要的特徵被提供給K-Neighbors Classifier進一步訓練，以提高模型的預測準確性。這樣的結合能夠充分利用兩個模型的優勢，處理複雜的分類問題並產生更準確的結果，經過我的測試，在輸入ddos的CSV檔後程式所顯示出的ddos的機率百分比的確比正常的CSV檔輸入後顯示出的機率百分比要高，但輸入ddos的CSV檔後顯示的機率百分比只有55%左右，實在說不上非常準確。

因此，我認為這個專案還有能夠進步的空間，以下讓我列點來一一講解：

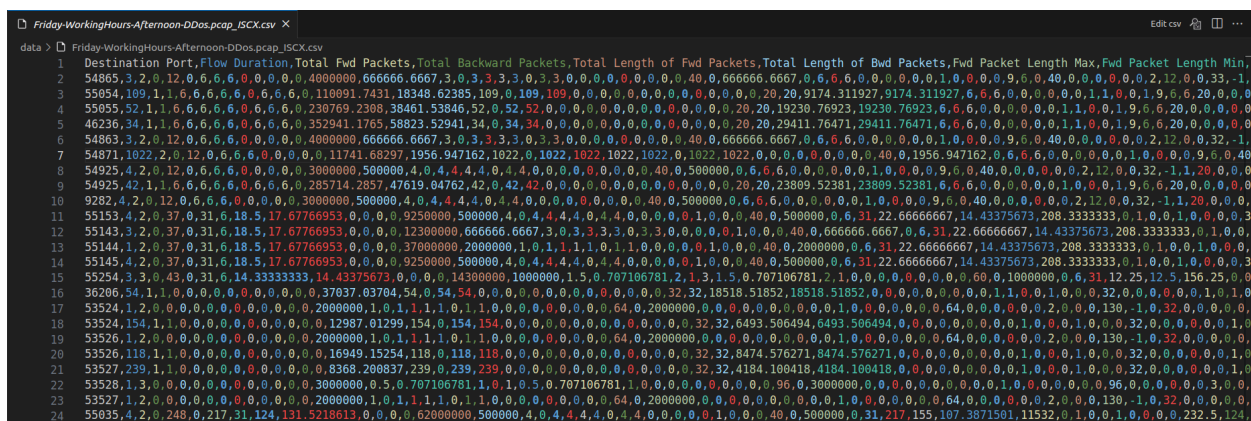
a. 圖形化的封包統計資料

若能夠在專題中加入圖形化的封包統計資料，能讓使用者對於傳進來的封包資料一目了然，此外，如果可以為圖形化後的封包圖表提供下載的功能，也可以為使用者節省不少需要自己分析封包並製表的時間。



b. Intrusion Detection Model的優化

本次專題的Intrusion Detection Model使用的是我自行提取的每個封包的資料訓練得來的，然而參考Dataset網頁中提供的他們整理的csv後，發現它們整理的CSV檔案中關於封包的資料是以flow為單位來整理的，而不是像我是直接把每個packet的資料記錄下來，因此我認為若在封包資料上有更加完整且整體的分析，應該可以大幅提昇Intrusion Detection Model的準確性。



總之，透過本次專題，我成功開發了一個功能豐富且使用友好的網路監測和入侵檢測系統。這個系統不僅提供了基本的封包監視和自訂過濾功能，還具有檔案儲存和轉換、入侵檢測等功能。我相信這個系統將對網路安全和監控領域的學習和應用能產生正面的影響，並能夠幫助使用者更好地管理和保護他們的網路環境。在未來也可以進一步優化系統性能、擴展功能，以及提供更廣泛的支援和應用。