Intrusion Detection System Using Machine Learning Algorithms

Team:
R11844821- Usha Raman Adapa
R11849307 - Naga Sai Ganga Bhavani Ghanta
R11840652 - Rishika Rao Takkallapally
R11842967 - Reshma Dudekula
R11850660 - Raja Venkatesh Gurugubelli
R11850821 - Subhash Chandra Bose Goud M
R11840489 - Kalyan Bathula
R11849382 - Banu Pooja Kannikanti

Introduction

The "Network Intrusion Detection System" monitors a network of computers, looking out for potentially risky actions include obtaining confidential information or corrupting/hacking networks. Accurate intrusion detection must be able to decrease the number of false alarms and speed up alert detection. The performance has been elevated using a variety of techniques. For the evaluation of IDS, machine learning techniques like Random Forest Approach is used.

Motivation

Today's world depends on networks, and data security has emerged as a crucial field of research. An intrusion detection system (IDS) is a device that monitors the state of a network's software and hardware and aids in cyber security.

However, several intrusion detection systems continue to have a high false alarm rate, generating numerous warnings for low-threat cases, increasing the workload of security analysts and potentially allowing serious attacks to go undetected.

Many researchers have concentrated on the creation of IDSs using machine learning techniques to address the issues mentioned above. Because of their generalizability, machine learning systems can also detect unknown threats.

Goals

Our First Goal

To Train Machine Learning Algorithm i.e Random Forest using KDD cup 99 Data Set to detect network intrusions.

Our Second Goal

Perform netwok attacks and capture network packets using wire shark. The captured data by wireshark is used to test the trained algorithm

Our Third Goal

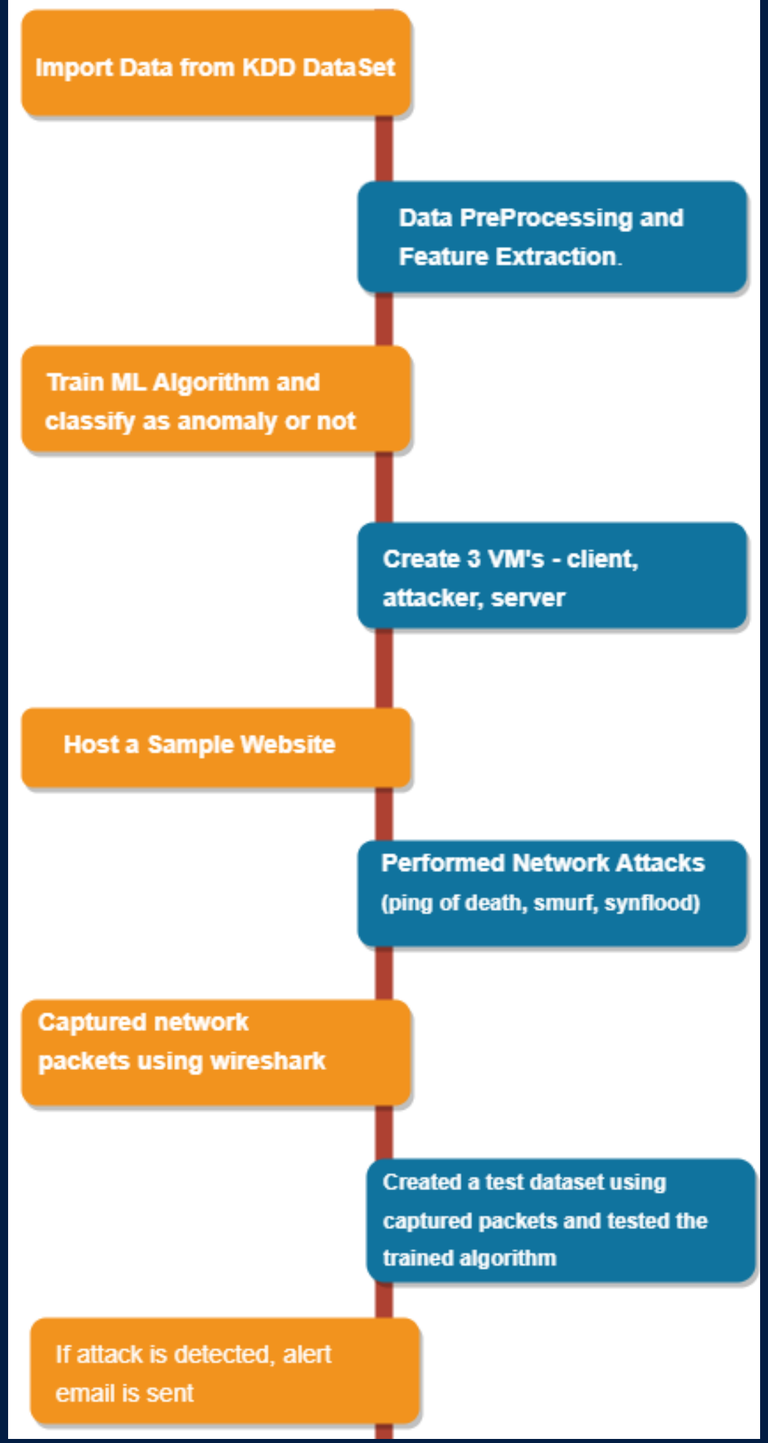
When intrusion is detected by the System, alert e-mail is sent to the network administrator.

Methodology

Data Collection: Algoroithm is trained using KDD dataSet. Packets with malicious content are generated for the purpose of classification of normal and anomaly by performing network attacks. This dataset created containing details of packets, is used for testing the algorithm. The features and attributes of the packets are visible in the captured Wireshark.

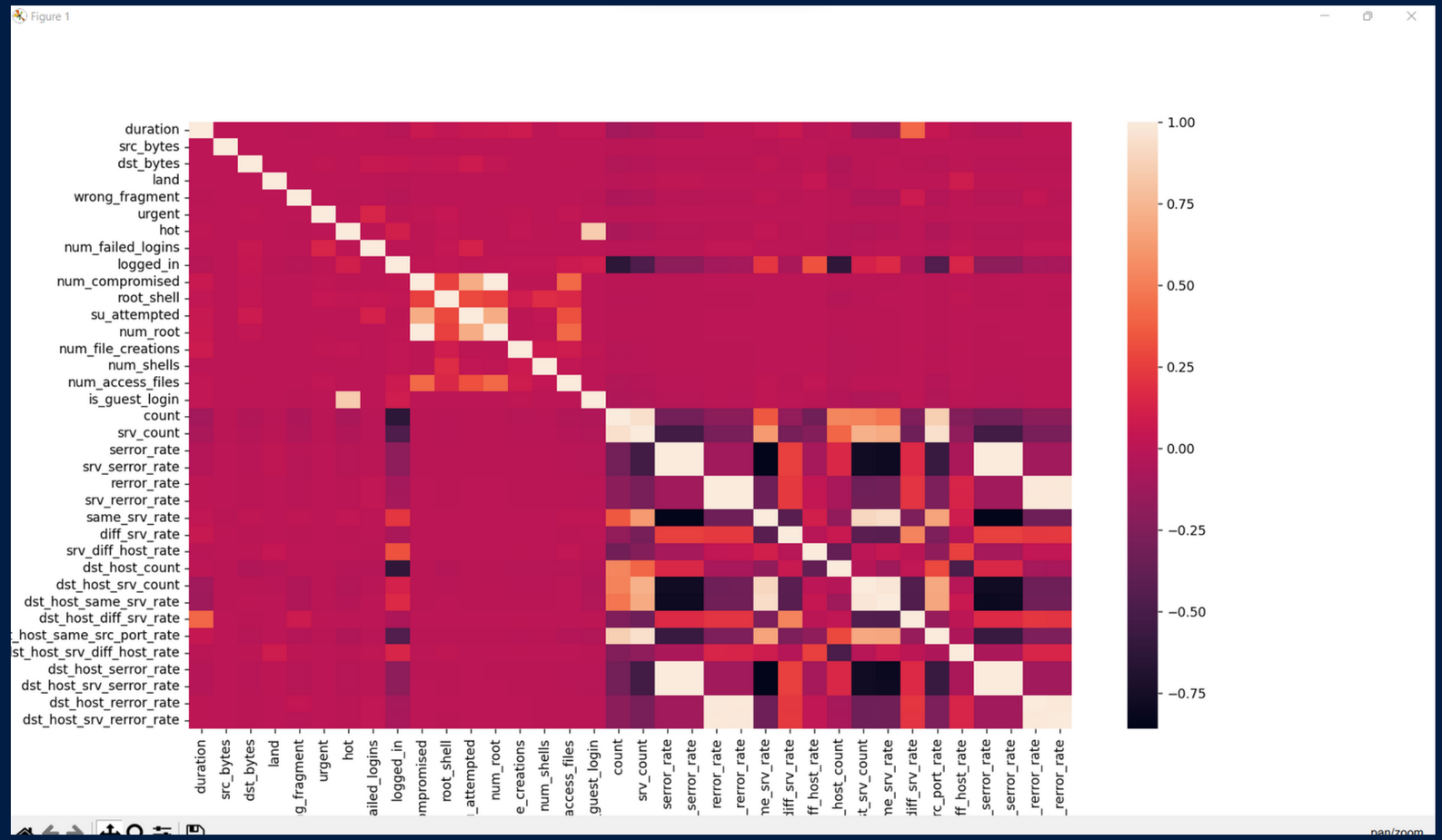
Feature Extraction: It is done on the dataset to select the important features that determine the classification of packets.

Analysis and Action: over the created dataset, different machine learning algorithms like, Random Forest is implemented to check the accuracy of classification to distinguish different type of attacks.



Result

Correlation map, which is used for feature extraction



Result

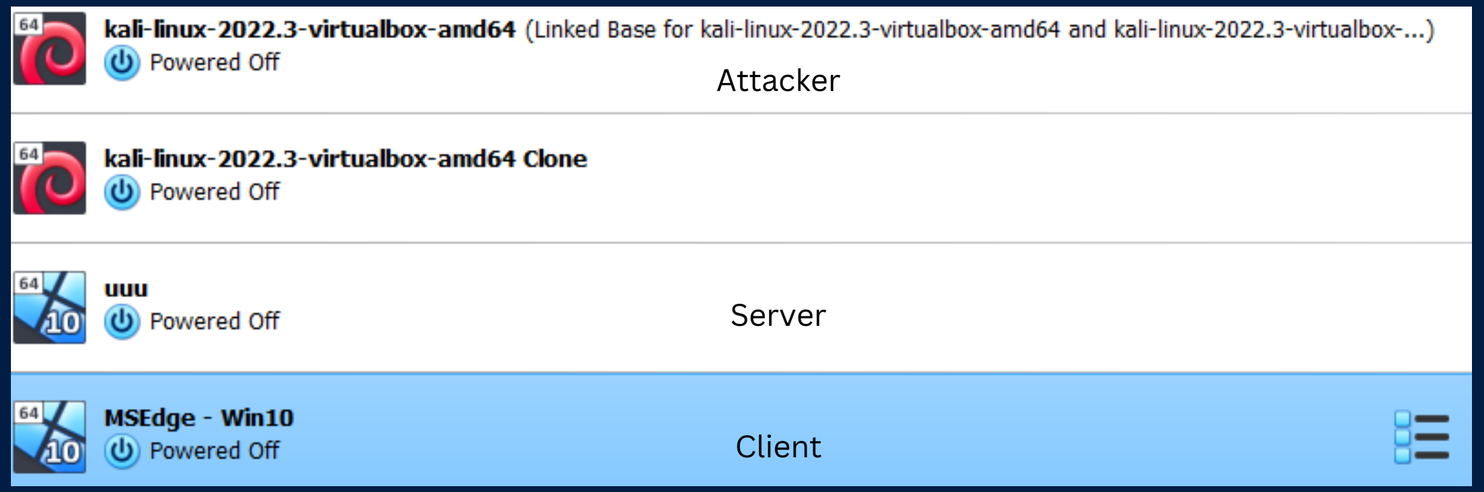
Trained Algorithm

```
# Target variable and train set
Y = df[['typeofattack']]
X = df.drop(['typeofattack',], axis=1)

#sc = MinMaxScaler()
#X = sc.fit_transform(X)

# Split test and train data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
print(X_train.shape, X_test.shape)
print(Y_train.shape, Y_test.shape)


rfc = RandomForestClassifier()
rfe = RFE(rfc, n_features_to_select=30)#30
rfe = rfe.fit(X_train, Y_train.values.ravel())


Y_test_pred = rfe.predict(X_test)
print (Y_test_pred)


print("Train score is:", rfe.score(X_train, Y_train))
print("Test score is:",rfe.score(X_test,Y_test))
```

```
['dos' 'dos' 'dos' ... 'normal' 'dos' 'dos']
Train score is: 0.9999969787971987
```

Result

Environment Set up for Performing Network Attacks

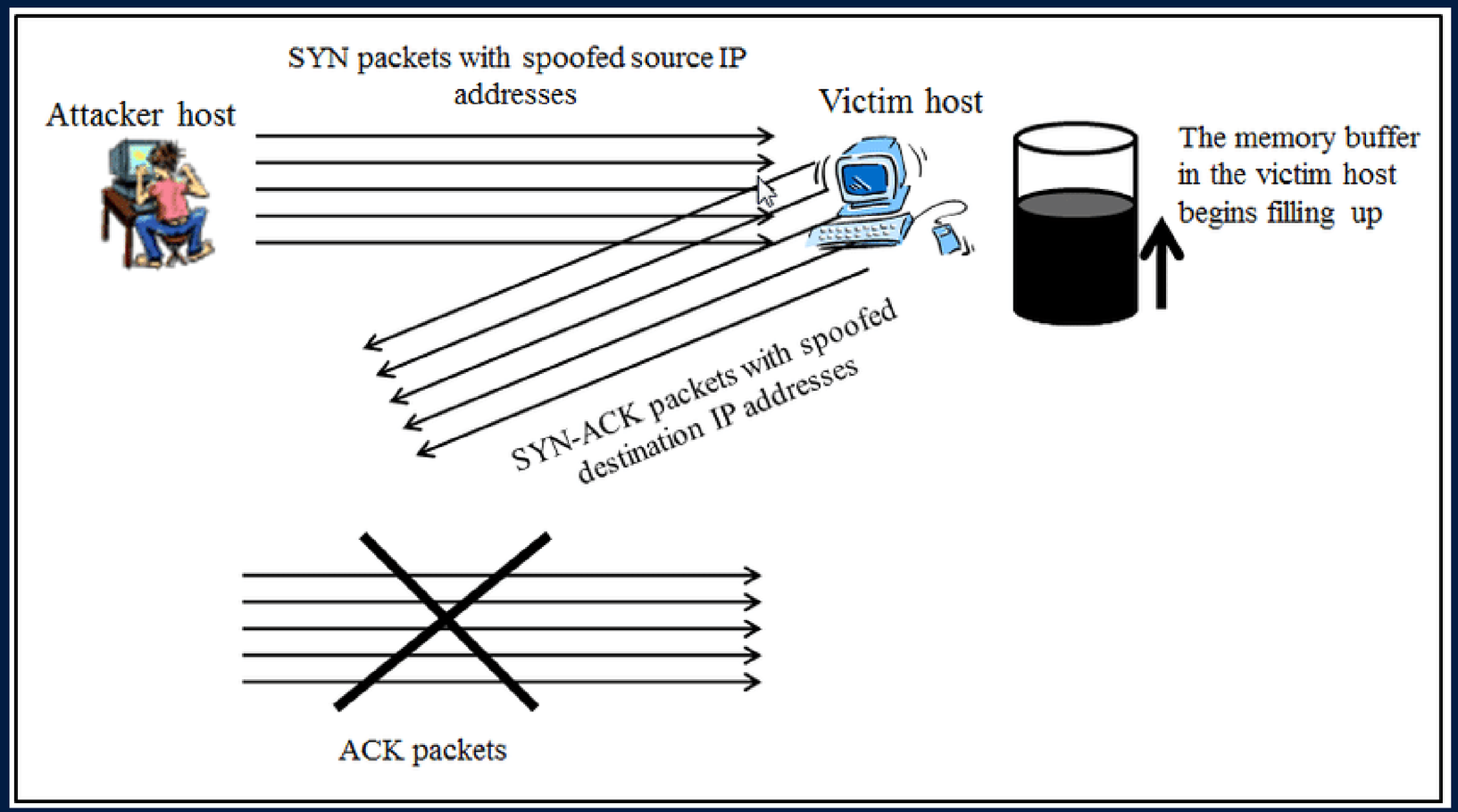


Sample website i.e hosted to check if the attacks are performed

Result

SynFlood Attack

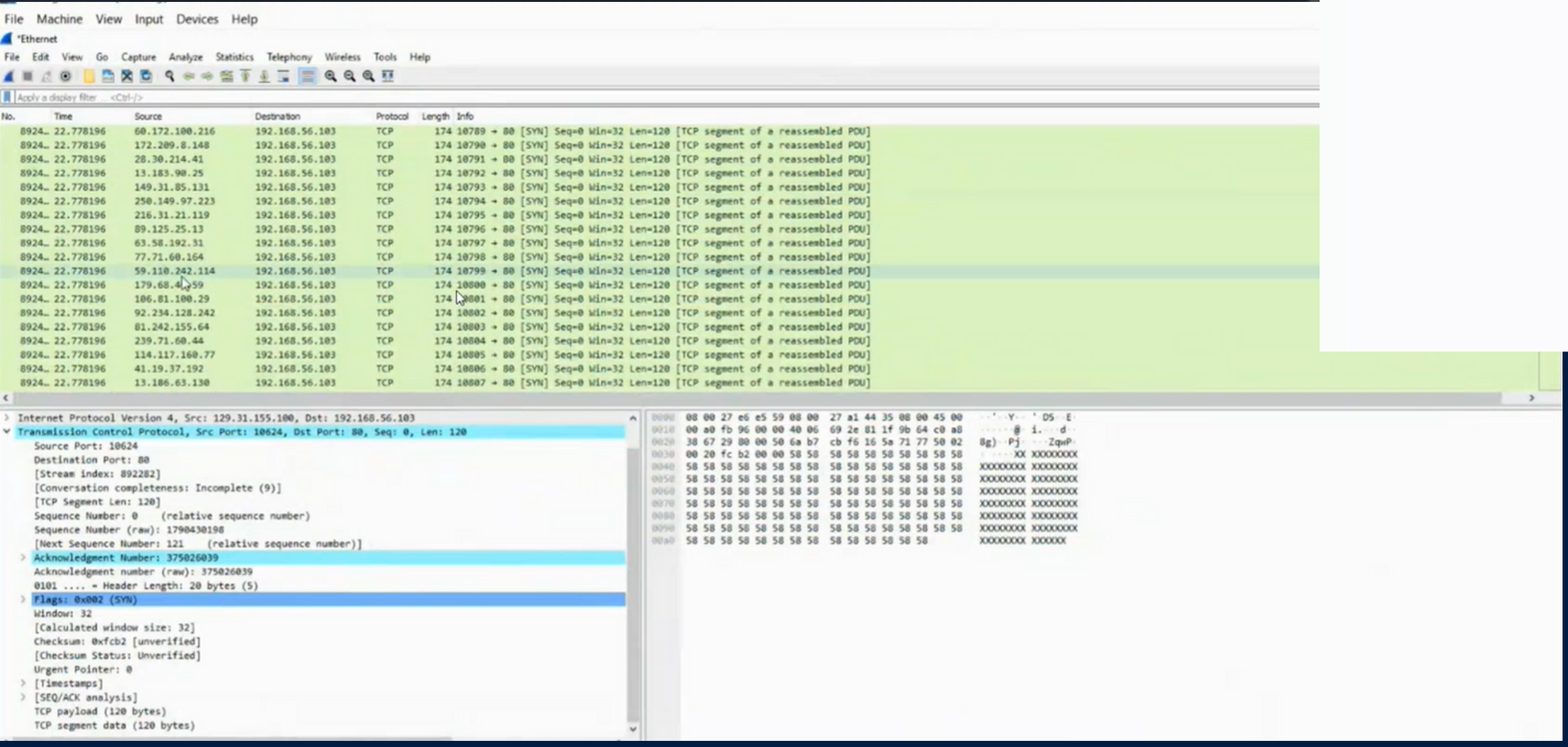
A SYN flood, also known as a TCP SYN flood, is a type of DoS that sends a large number of SYN requests to a server in order to overwhelm it with open connections.

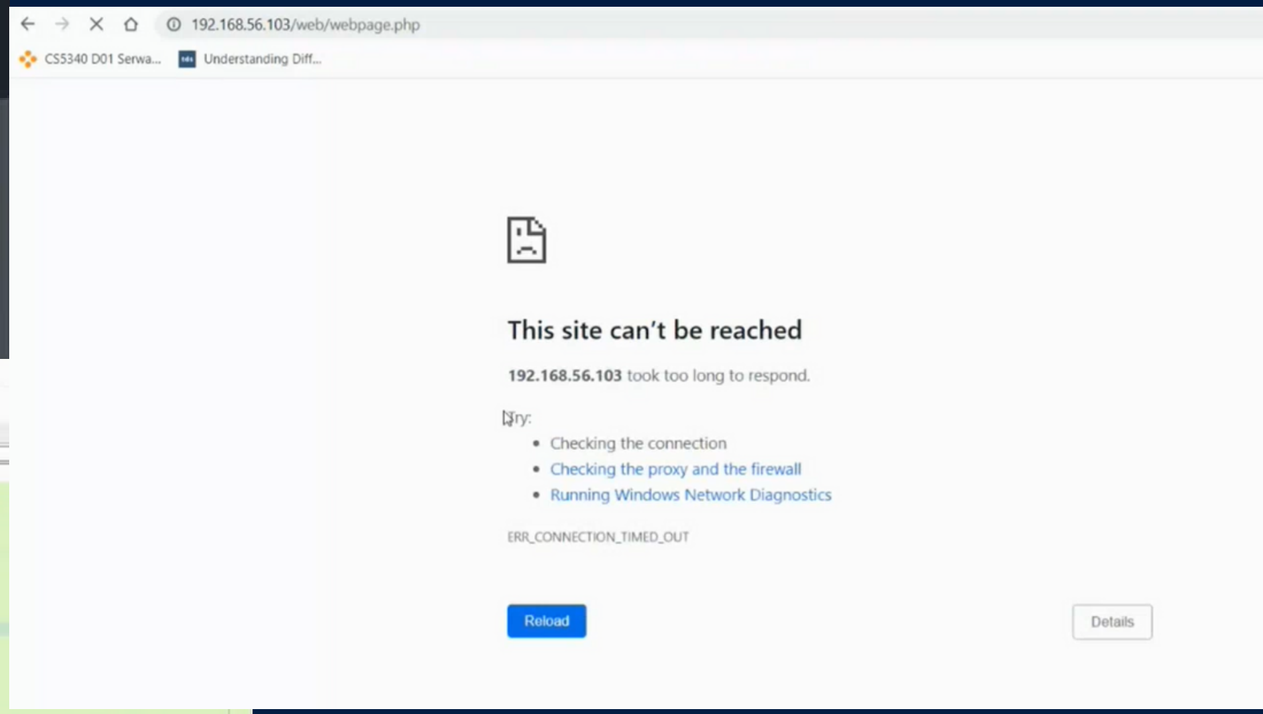


Result

SynFlood Attack

```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 --count 15000 --data 240 --syn --win 64 -p 80 --flood --rand-source 192.168.56.103
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
HPING 192.168.56.103 (eth1 192.168.56.103): S set, 40 headers + 240 data bytes
hping in flood mode, no replies will be shown
```

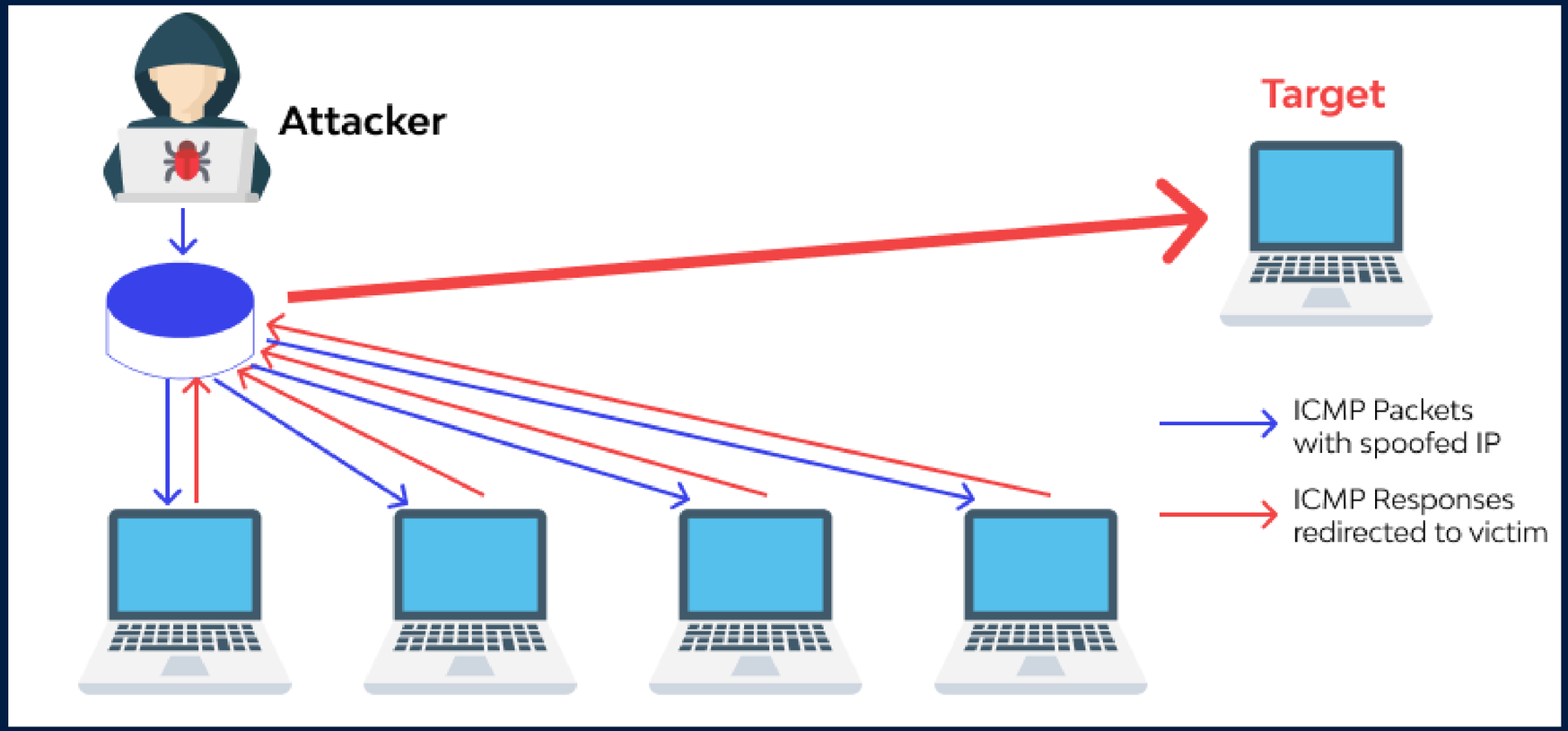




Result

Smurf Attack

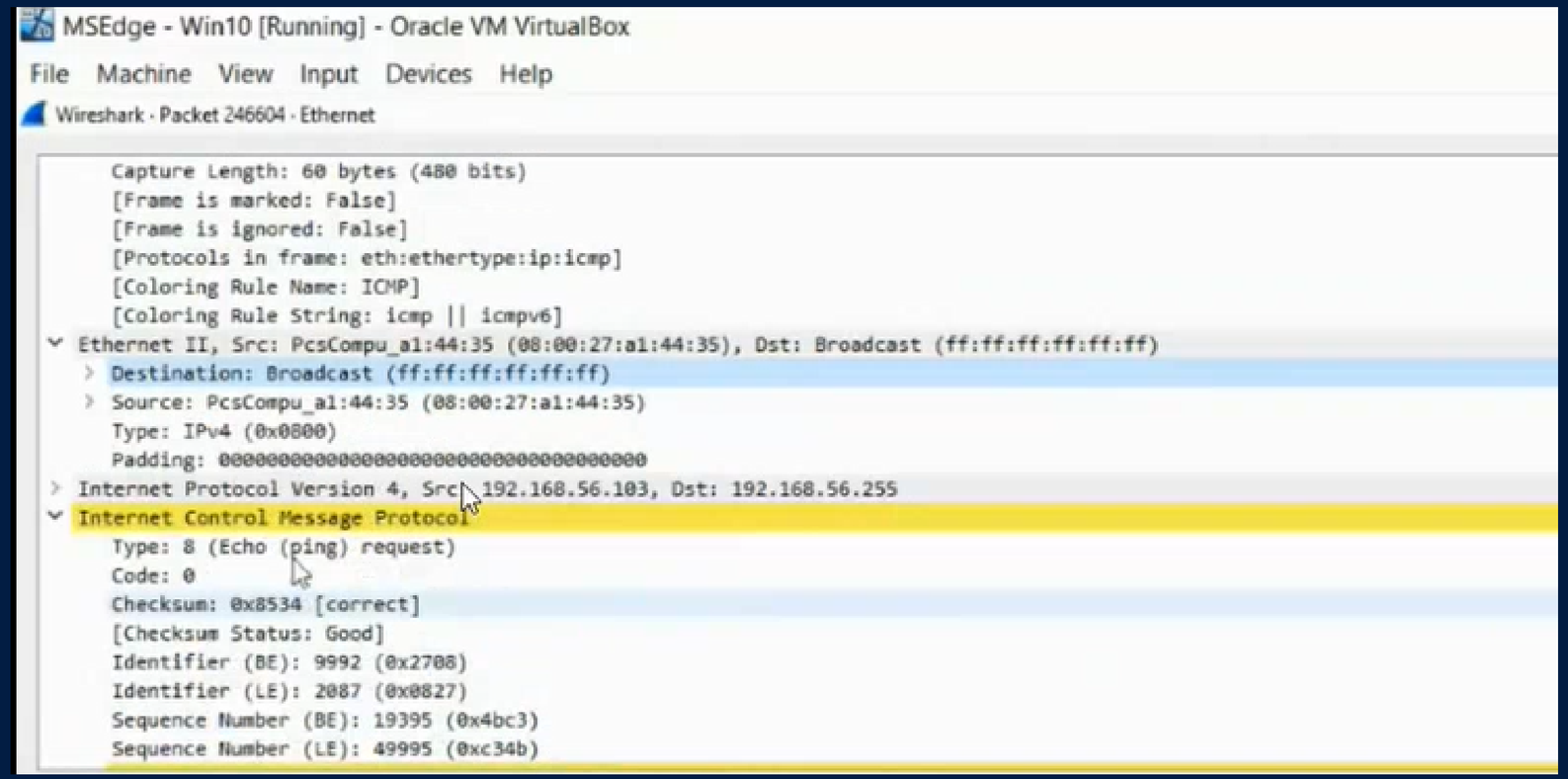
A Smurf attack is a distributed denial-of-service attack in which a large number of Internet Control Message Protocol (ICMP) packets are broadcast to a computer network using an IP broadcast address with the source IP of the intended victim's spoofing. By default, the majority of networked devices will reply to this by sending a response to the source IP address. The victim's computer will experience traffic overload if there are many machines on the network that are receiving and responding to these packets. This may cause the victim's computer to become so slow that using it is impossible.



Result

Smurf Attack

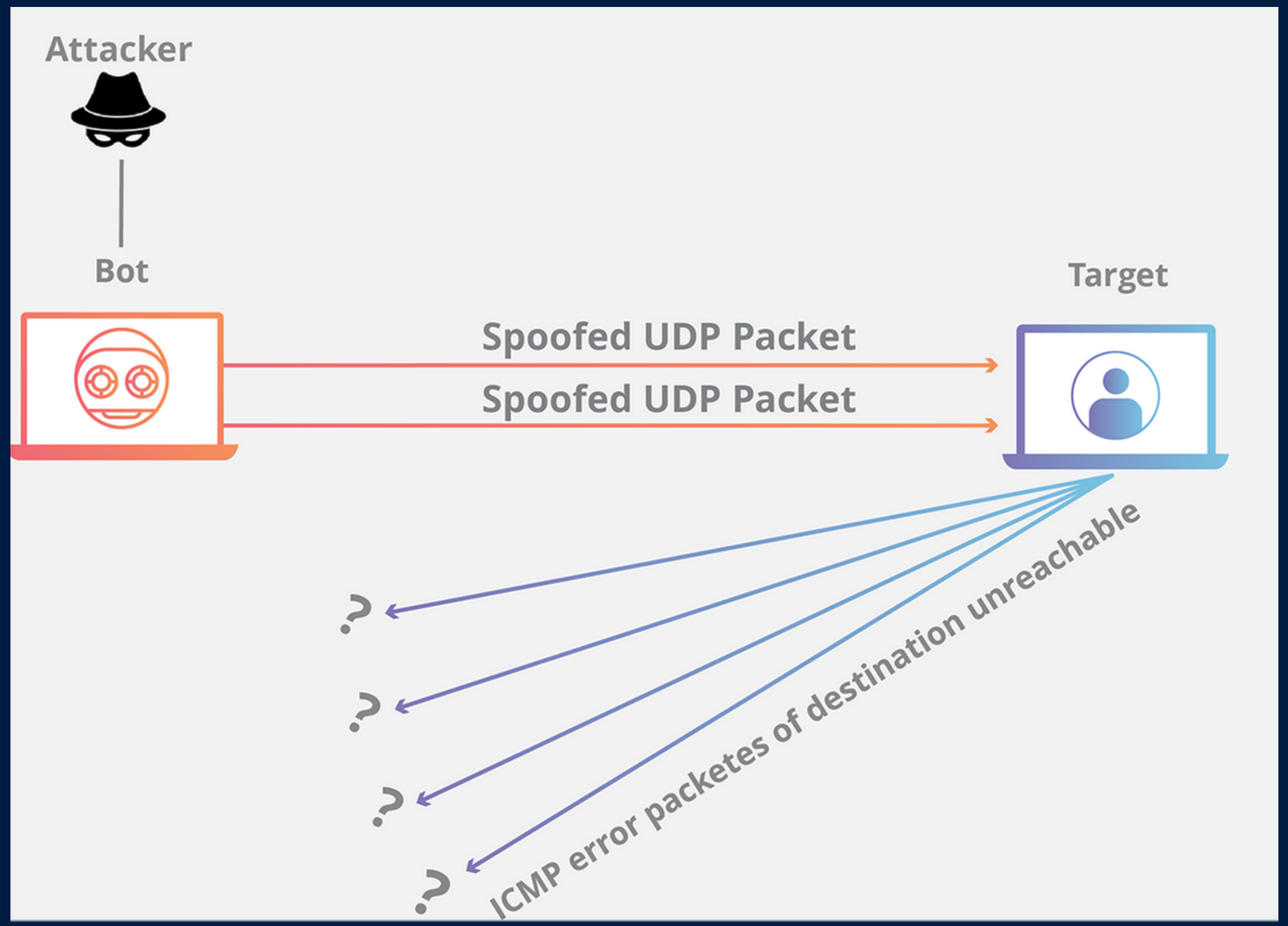
```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 --icmp --count 200000 --spoof 192.168.56.103 192.168.56.255 --flood
[sudo] password for kali:
HPING 192.168.56.255 (eth1 192.168.56.255): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```



Result

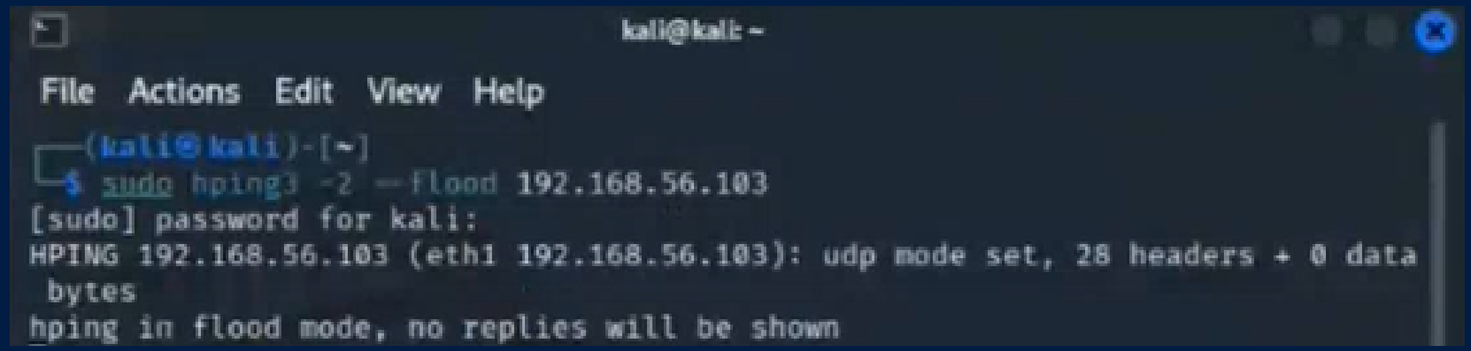
UDP FLOOD ATTACK

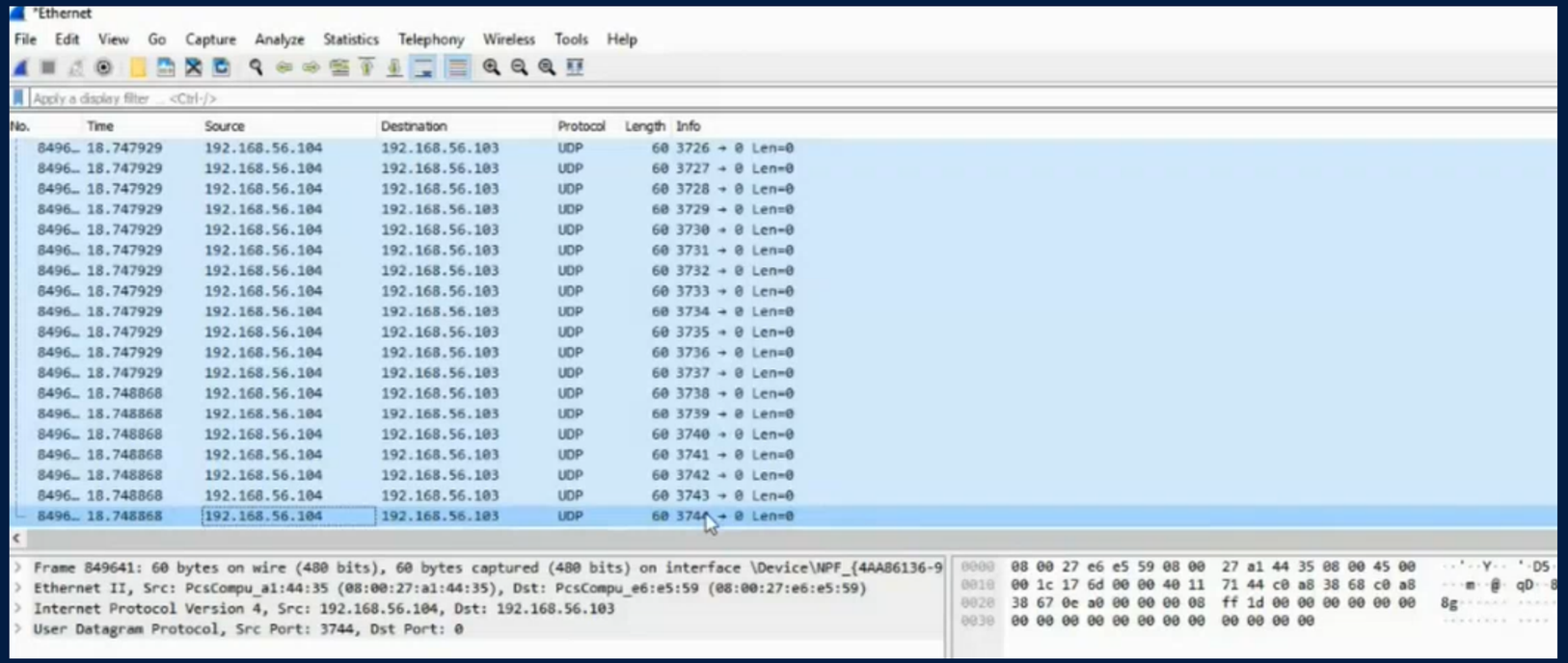
A Denial of Service (DoS) attack known as a "UDP flood" occurs when an attacker floods the targeted host's random ports with IP packets containing UDP datagrams. If there are no applications connected to these datagrams, the receiving host searches for them and sends a "Destination Unreachable" packet back. The system becomes overloaded and unresponsive to additional clients as more and more UDP packets are received and answered.



Result

UDP FLOOD ATTACK





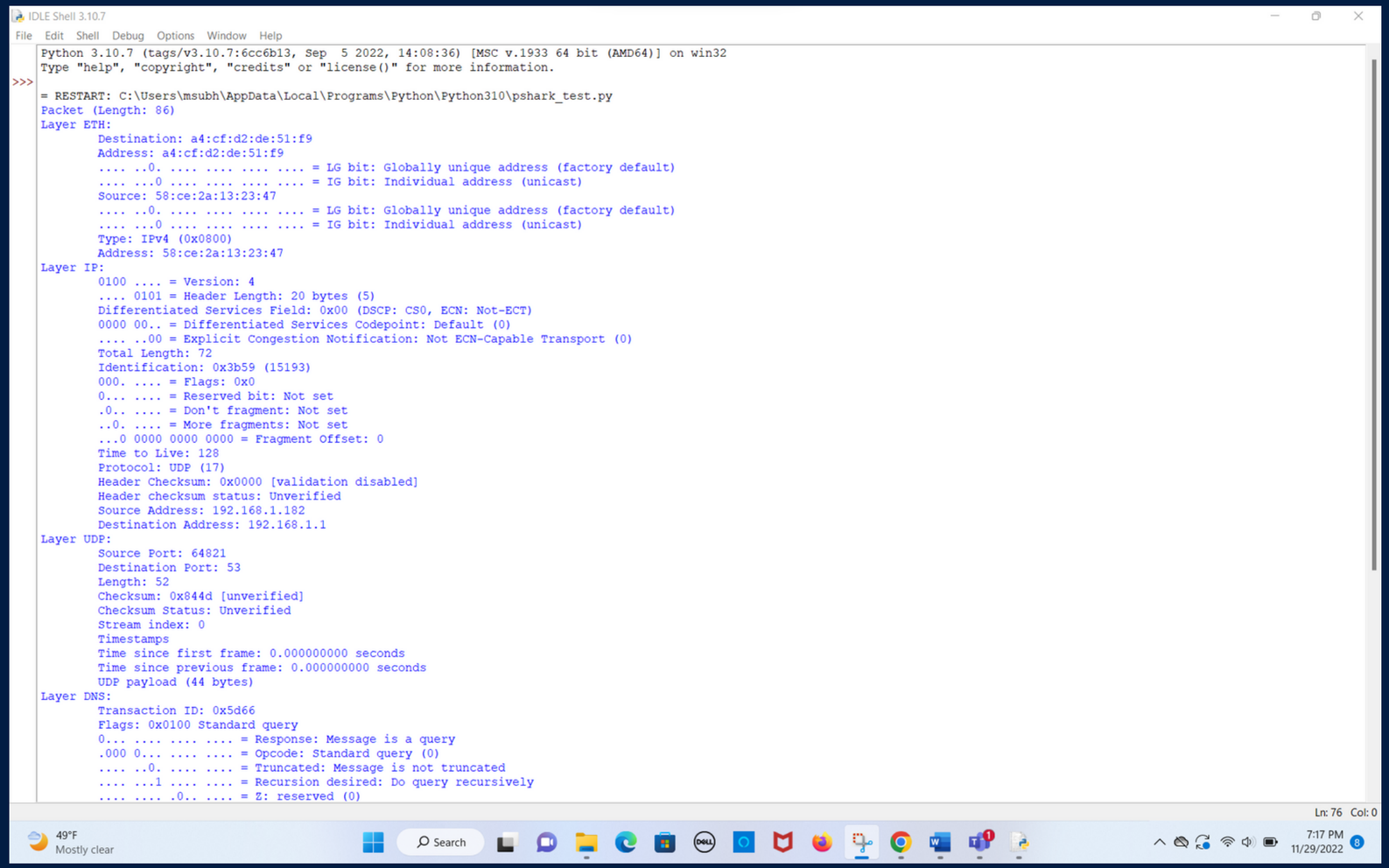
Result

Packets Captured using Pyhon

```
def get_packet_details(packet):
    """
    This function is designed to parse specific details from an individual packet.
    :param packet: raw packet from either a pcap file or via live capture using TShark
    :return: specific packet details
    """
    protocol = packet.transport_layer
    source_address = packet.ip.src
    source_port = packet[packet.transport_layer].srcport
    destination_address = packet.ip.dst
    destination_port = packet[packet.transport_layer].dstport
    packet_time = packet.sniff_time
    packet_length=packet.length
    # Reading input Intrusion data from network
    data_headers=['duration', 'protocol_type', 'flag', 'src_bytes', 'dst_bytes', 'land',
                  'root_shell', 'su_attempted', 'num_file_creations', 'num_shells', 'num_
                  'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count
                  'dst_host_srv_diff_host_rate']
    data=[0,protocol,0,packet_length,packet_length,0,source_address,source_port,0,0,dest
    #[0,2,0,221,222,0,0,0,0,0,0,0,0,0,0,0,16,15,0.94,0,0.94,0.12,0,15,16,0,0.07,0.1

    f = open('C:\\Users\\msubh\\OneDrive\\Desktop\\Project-IICS\\ip1.csv', 'w')
    row=data
    writer = csv.writer(f)
    writer.writerow(data_headers)
    writer.writerow(row)
    f.close()

    return f'Packet Timestamp: {packet_time}' \
           f'\nProtocol type: {protocol}' \
           f'\nSource address: {source_address}' \
           f'\nSource port: {source_port}' \
           f'\nDestination address: {destination_address}' \
           f'\nDestination port: {destination_port}\n'


capture = pyshark.LiveCapture(bpf_filter='udp')     #(bpf_filter='tcp port 80')
capture.sniff(packet_count=10)
#print(capture)
for packet in capture:
    #print(packet.highest_layer)
    print(get_packet_details(packet))
```

Result

Packets Captured using Pyhon



Result

Created a test data and verified the accuracy (66%)

```
testingdatapath="C:\\Users\\msubh\\OneDrive\\Desktop\\Project-IICS\\inputdata1.csv"
td=pd.read_csv(testingdatapath)
#td = pd.DataFrame(pd.read_excel(testingdatapath))
td.dropna()
Y_data = td[['typeofattack']]
X_data = td.drop(['typeofattack',], axis=1)
td.head()
givendatapredict1=rfe.predict(X_data)
print(givendatapredict1)
print("Test score is:",rfe.score(X_data,Y_data))
#-----
```

```
['dos' 'dos' 'normal']

Warning (from warnings module):
  File "C:\Users\msubh\AppData\Local\Programs\Python\Python310\lib\site-packages
\sklearn\base.py", line 493
    warnings.warn(message, FutureWarning)
FutureWarning: The feature names should match those that were passed during fit.
 Starting version 1.2, an error will be raised.
Feature names unseen at fit time:
- dst_host_srv_diff_host_rate0.0
Feature names seen at fit time, yet now missing:
- dst_host_srv_diff_host_rate

Test score is: 0.6666666666666666
```

Result

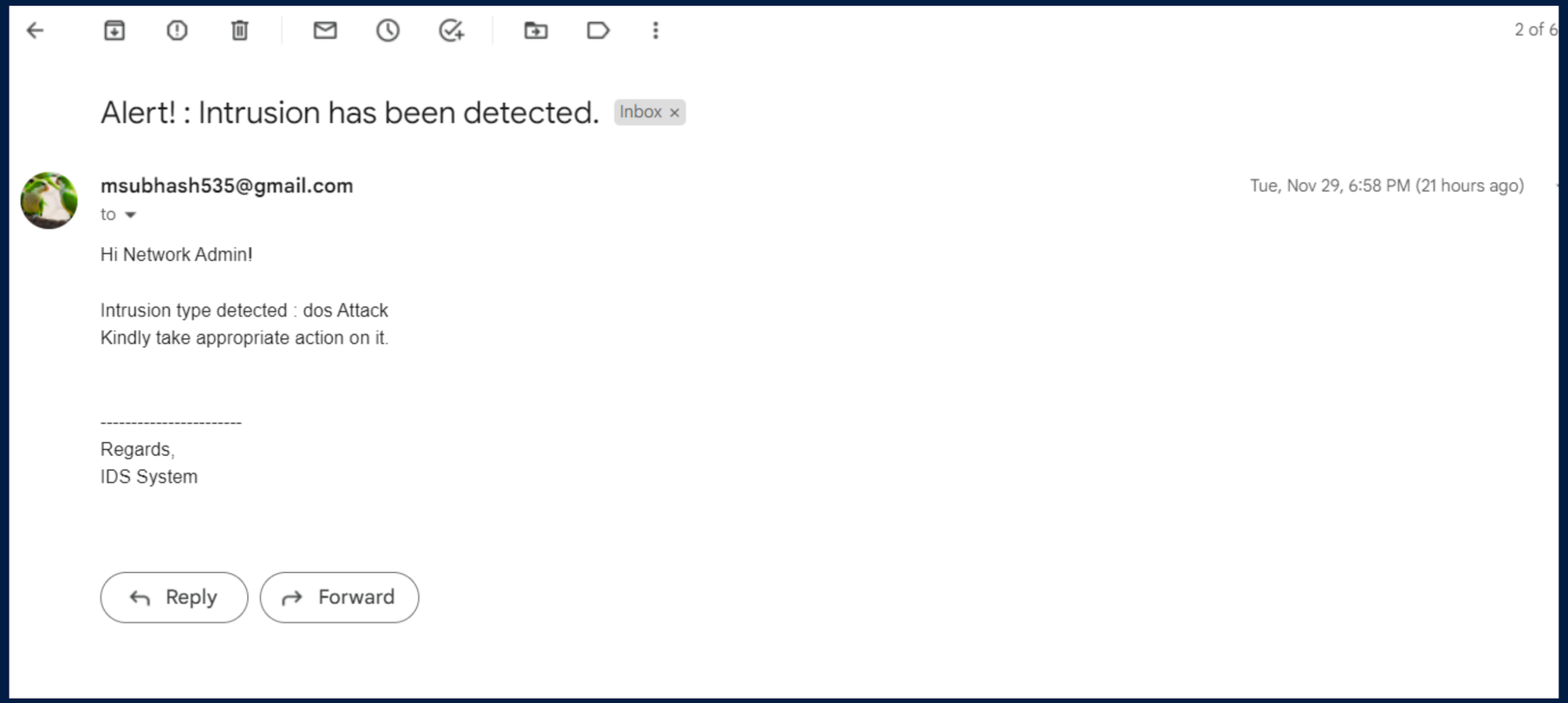
Alert email is sent, if attack is detected

```
def Send_EMail_Method(Intrusion_Type):
    # SET EMAIL LOGIN REQUIREMENTS
    gmail_user = 'msubhash535@gmail.com'
    gmail_app_password = 'ozapvsxjqlxbbilg' #'149162***'


    # SET THE INFO ABOUT THE SAID EMAIL

    sent_from = gmail_user
    sent_to = ['adapa.usharaman@gmail.com','msubhash535@gmail.com','sm@ttu.edu']    # ['msubhash535@gmail.com', 'sm@ttu.edu']
    sent_subject = "Alert! : Intrusion has been detected."

    sent_body = ("Hi Network Admin!\n\n"
                 "Intrusion type detected : "+Intrusion_Type+"\n"
                 "Kindly take appropriate action on it.\n"
                 "\n"
                 "\n-----------------------\n"
                 "Regards,\n"
                 "IDS System\n")


    message = 'Subject: {}\n\n{}'.format(sent_subject, sent_body)
    # SEND MAIL
    try:
        server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
        server.ehlo()
        server.login(gmail_user, gmail_app_password)
        #server.sendmail(sent_from, sent_to, email_text)
        server.sendmail(sent_from, sent_to, message)
        server.close()
```

```
if(givendatapredict[0]!='normal'):
    print("Intrusion of type -",givendatapredict[0]," Attack is detected!")
    Send_EMail_Method(givendatapredict[0]+" Attack")
```

Result

Email Sent



Conclusion:

- IDS is designed to provide strategies to protect existing systems on networks directly or indirectly online. But finally at the end of the day to the Network Administrator to make sure his network is out of danger.
- Many different methods have been used in the screening process. Among them machine learning plays a vital role. This analysis works with machine learning algorithms.
- This does not completely protect the network from attackers, but IDS helps the Network Administrator track down the bad guys on the internet whose purpose is to bring your network into a hotspot and make it vulnerable to attack.

References:

- 2021 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS 2021), 26 June 2021, Shah Alam, Malaysia -Intrusion Detection Systems Based on Machine Learning Algorithms
- Network Intrusion Detection using Machine Learning Techniques - IEEE 2020 by SCSVComputer & Internet Security A Hands-on Approach MV University, Kanchipuram, India
- Network Security A Hands-on Approach - 3rd edition , Wenliang Du