A Novel Supervised Deep Learning Solution to detect Distributed Denial of Service (DDoS) attacks on Edge Systems using Convolutional Neural Networks (CNN)

SCIENTEER PROJECT ID: 182334

Note: Figures are created from cited sources or are my own

1: Background and Rationale

- The growing foray of Distributed Denial of Service (DDoS) attacks has caused the unavailability of millions of cloud services,
- Up to $660 Billion in loss of revenue (1),
- DDoS is a threat to basic internet standards and security.
- DDoS attacks critical services and flood the network with malicious traffic.
- In the first half of 2022 alone, there were 6,019,888 DDoS attack devices across the world. (2)

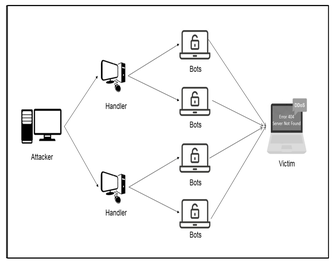

Fig. 1: Attacker to User network connectivity

- Therefore, models to defend and mitigate DDoS traffic need to have three features.
- Scalability, Flexibility, Reliability
- Current state of the art models, lack at least one of these components.
- The model **proposed will have to include these 3 features** and use machine learning to build an active intrusion detective system (IDS)

Rationale: Create a novel supervised model, that can handle any size of data and differentiate between malicious and benign traffic consistently. This model should be functional on private and public networks

2a: Engineering Question

Can a dynamic deep learning model effectively differentiate between malicious and benign traffic based on different characteristics of the dataset? Additionally, can CNN models be successfully applied to the field of cybersecurity?

2b. Engineering Goal

The engineering goal of this project is to **design and develop a dynamic deep learning model that can accurately identify malicious and benign network traffic** across a wide range of attack methods and situations, even when dealing with large amounts of real-time data in short time constraints.

3: Dataset and Materials

- Data was used from the Canadian Institute of Cybersecurity's (CIC) DDoS Evaluation Dataset released in 2019
  - CICDDoS2019 contains benign and the most up-to-date common DDoS attacks, which resembles the true real-world data (PCAPs).
  - Ideal for deep learning models because of variety of attacks and headers.
- Materials:
  - Computer
    - Linux Mint VM
  - Python Environment (using the Conda Framework and VSCode)
  - Keras API easily builds layers in the CNN Model.
  - Python wrapper for Pyshark, allowing python packet parsing using Wireshark dissectors.
  - Seaborn/Matplotlib functions to visualize data
  - TensorFlow to build the model in Python
  - Wireshark to visualize the packets in the dataset
  - CIC Dataset (8)

4a: Procedure

1. Data Preprocessing Algorithm

In order to analyze the dataset and implement our CNN model, we had to preprocess our data to ensure the fairness and equality of the spread of data to get the most accurate results.

- Algorithm 1 uses key feature extraction to parse packets into "flows" and time stamps (headers and columns) to easily feed into the CNN Model.
- Normalize and pad the resultant flow of data

Algorithm 1 DDoS Pre Processing

Input: PCAP Data ($pcap$), time flow(t), max packets per sample (m), abel of packet (l)
Output: Labelled samples for CNN Input ($sample$)

```
procedure PrePROCESSING(pcap, t, m, l)
    s ← ∅
    t_0 ← 0 ▷ set a local variable to the counter for the time for each packet
    id ← ∅ ▷ variable which looks at Source IP, Dest IP, etc. (for labeling)
    for each packet ∈ pcap do
        id ← packet.headers ▷ set "id" to the headers of the packet.
        if t_0 == 0 or packet.time ≥ t_0 + t then
            t_0 ← packet.time
        end if
        if sample[t_0, id] < m then
            sample[t_0, id].packet.append() ▷ add max num. of packets to sample
        end if
    end for
    sample ← normalization(s) ▷ normalize the sample for empty space
    for each flows ∈ sample do
        sample.label ← l[flows.id] ▷ Labeling the processed data
    end for
return sample
end procedure
```

Fig. 2: Algorithm with DDoS data preprocessing in Python

2. CNN Model Architecture

Next, we implemented our CNN model using TensorFlow and the Keras API for ease of coding and debugging. (See 4b. CNN Architecture)

3. Training

Finally, we will use the Adam optimizer for training the model and set the learning rate, batch size, and number of epochs as hyperparameters that the model tunes each iteration.

The data will be evenly split into 3 distinct sets (train, test, val) in HDF5 format for readability

4. Testing

The model will be tested under common performance metrics such as a confusion matrix, accuracy, and F1 Score (See 5. Results for more information)

4b: CNN Architecture

1. The output of the preprocessing algorithm (Fig. 5) will then input the proposed CNN Architecture to be trained. (Figure 6)

	Flow 1	Flow 2	Flow 3	Flow 4
Time t_0	pkt. 1 pkt. 2	pkt. 1 pkt. 2 pkt. 3	pkt. 1	pkt. 1 pkt. 2
Time $t_0 + t$		pkt. 4 pkt. 5	pkt. 2	pkt 3.
Time $t_0 + 2t$	pkt. 3		pkt. 3 pkt. 4	
Label	0	1	1	0

Fig. 3. Proposed CNN architecture training flow

2. A 2D convolutional layer with 64 filters and a kernel size of 3 x 3.
- The convolutional layer will be responsible for extracting features from the input data by sliding the filters over the input and computing dot products (9).

3. A dropout layer with a recommended *dropout rate of 0.5 (11).*
- This layer will randomly set a certain percentage of input units to 0 at each update during training time, which helps prevent overfitting.
- Here we will use the ReLU (Rectified Linear Unit) activation function. This function calculates the output of a neuron as the maximum between 0 and the input value, mathematically represented as $f(x) = \max(0,1)$
- Turns off neurons that do not affect the prediction

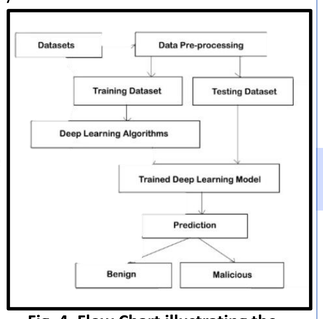

Fig. 4. Flow Chart illustrating the algorithm

4. A GlobalMaxPooling2D layer. This layer will perform max pooling on the input, which reduces the spatial dimensions of the input while retaining important features.
5. A flattened layer. This layer will flatten the output of the previous layer into a one-dimensional tensor.
6. The final fully connected layer will contain a **sigmoid activation function** (Figure 6) (12).

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

- This layer will perform the final computation to output a probability of the input being a DDoS attack.
- Outputs a number between 0 and 1 as labels to designate benign vs malicious traffic
- When $p > .5$, the attack will be classified as a DDoS attack, otherwise it is benign.

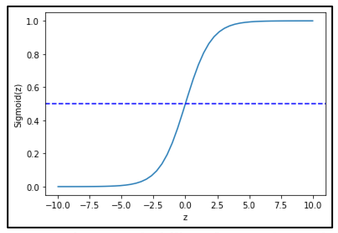

Fig. 5. Probability of a DDoS attack (in the fully connected layer as a function of the input data set)

5a: Training and Results

The final model resulted in a 64-kernel convolutional model trained over 20000 flows of data.(Figure 6)

- In the training phase, the model goes over a grid of hyperparameters, maximum 1000 epochs for each point in the grid. The training process can stop earlier if no progress towards the minimum loss is observed for PATIENCE=10 consecutive epochs.
- It achieved a training accuracy of **96.7%**
- Performance (F1 score) is defined as the harmonic mean between precision and recall. It is used as a statistical measure to rate performance: **96.4%**

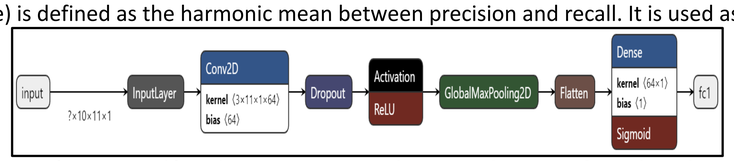

Fig. 6. Flow from the input (DDoS attack) to the various training layers, and output of ML algorithm predicting an attack

5b: Results

- **The proposed model was tested on over 5000 samples of unseen DDoS flows.**
- The Confusion Matrix below shows that our model was very successful. We can use these values to analyze our results in the form of other common performance metrics
  - True Negative (TNR), True Positive (TPR)
  - False Negative (FNR), False Positive (FPR)

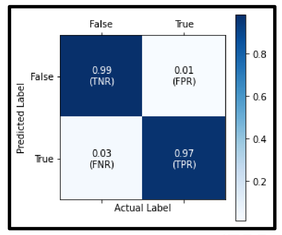

Fig. 7. Confusion Matrix of Testing accuracy of CNN Model

Accuracy: $\frac{(TPR+TNR)}{(TPR+TNR+FPR+FNR)} = 97.7\%$

F1 Score: $\frac{(TP+TN)}{TP+TN+FP+FN} = 97.2\%$

6: Discussion /Conclusions

Strengths of this study:

- Automated hyperparameter setup to fine tune and optimized to not keep lower accuracy model
- The model's use of validation-test split optimizes the model for different features in PCAP files
- The ReLU Activation and kernel technique was able to successfully identify the importance of specific features with respect to others.

Further Research

- The model is based on a lab- based environment and dataset
- However, the algorithm can be trained for more complicated circumstances
- Can be developed into an active IDS platform to implement in the real world

6: Bibliography

1. AO Kaspersky Lab. (2023, January 10). Distributed denial of service: Anatomy and impact of DDoS attacks. www.kaspersky.com. Retrieved January 22, 2023, from https://www.kaspersky.com/resource-center/preemptive-safety/how-does-ddos-attack-work
2. NetScout. (2022, September 26). NETSCOUT DDoS Threat Intelligence Report - Latest Cyber Threat Intelligence Report. Netscout. Retrieved January 18, 2023, from https://www.netscout.com/threatreport/
3. What is DDoS mitigation? [Internet]. Cloudflare. Cloudflare Inc.; 2019 [cited 2022Dec22]. Available from: https://www.cloudflare.com/learning/ddos/ddos-mitigation/
4. Kawtar Bouzoubba, Youssef Taher, and Benayad Nsiri, "Predicting DOS-DDOS Attacks: Review and Evaluation Study of Feature Selection Methods based on Wrapper Process".International Journal of Advanced Computer Science and Applications(IJACSA), 12(5), 2021. http://dx.doi.org/10.14569/IJACSA.2021.0120517
5. Mahjabin T, Xiao Y, Sun G, Jiang W. "A survey of distributed denial-of-service attack, prevention, and mitigation techniques." International Journal of Distributed Sensor Networks. 2017;13(12). doi:10.1177/1550147717741463
6. S. T. Zargar, J. Joshi and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," in IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2046-2069, Fourth Quarter 2013, doi: 10.1109/SURV.2013.031413.00127.
7. T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 773–788, March 2019.
8. Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", IEEE 53rd International Carnahan Conference on Security Technology, Chennai, India, 2019.
9. R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincón and D. Siracusa, "Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection," in IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 876-889, June 2020, doi: 10.1109/TNSM.2020.2971776.
10. Madhavan, S. (2021, July 13). Introduction to convolutional neural networks. IBM developer. Retrieved January 5, 2023, from ttps://developer.ibm.com/articles/introduction-to-convolutional-neural-networks/
11. Brownlee, J. (2020, August 20). A gentle introduction to the rectified linear unit (ReLU). MachineLearningMastery.com. Retrieved December 18, 2022, from https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
12. M. Roopak, G. Y. Tian and J. Chambers, "An Intrusion Detection System Against DDoS Attacks in IoT Networks," 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0562-0567, doi: 10.1109/CCWC47524.2020.9031206.
13. Sharma, S. (2022, November 20). Activation functions in neural networks. Medium. Retrieved January 22, 2023, from https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Supplementary Information

Note: Figures are created from cited sources or are my own

Code of Preprocessing Data

pre-processing.py

```
import sys
import pyshark
import socket
import argparse
import ipaddress
from sklearn.feature_extraction.text import CountVectorizer
from multiprocessing import Process, Manager, Value, Queue
import util_functions

dataset = {'attackers': ['172.16.0.5'], 'victims': ['192.168.50.1', '192.168.50.4']}
#identify IP's of victims/attackers as given in the documentation

#main function to parse packets for input --> label
def process_traffic(capture, dataset_type, labels, max_flow_duration, traffic_kind='all',
window_duration=TIME_WINDOW):
    start = time.time()
    flow_dict = OrderedDict()
    labeled_flows = []

    window_start = start
    window_end = start + window_duration

    if isinstance(capture, pyshark.LiveCapture):
        for packet in capture.sniff_continuously():
            if time.time() >= window_end:
                break
            parsed_packet = parse_packet(packet)
            flow_dict = store_packet(parsed_packet, flow_dict, window_start,
max_flow_duration)
    elif isinstance(capture, pyshark.FileCapture):
        while time.time() < window_end:
            try:
                packet = capture.next()
                parsed_packet = parse_packet(packet)
                flow_dict = store_packet(parsed_packet, flow_dict, window_start,
max_flow_duration)
            except StopIteration:
                break

    apply_labels(flow_dict, labeled_flows, labels, traffic_kind)
    return labeled_flows
```

Code of CNN model

cnn.py

```
import tensorflow as tf
import numpy as np
import random as rn
import os
import csv

#layers and hyperparameters of model
PATIENCE = 10
DEFAULT_EPOCHS = 1000
hyperparamters = {
    "learning_rate": [0.1,0.01,0.001],
    "batch_size": [1024,2048],
    "kernels": [1,2,4,8,16,32,64],
    "regularization" : ['l1','l2'],
    "dropout" : [0.5,0.7,0.9]
}

#build the model and it's layers
def Conv2DModel(model_name,input_shape,kernel_col,
kernels=64,kernel_rows=3,learning_rate=0.01,regularization=None,dropout=None):
    K.clear_session()

    model = Sequential(name=model_name)
    regularizer = regularization

    model.add(Conv2D(kernels, (kernel_rows,kernel_col), strides=(1, 1),
input_shape=input_shape, kernel_regularizer=regularizer, name='conv0'))
    if dropout != None and type(dropout) == float:
        model.add(Dropout(dropout))
    model.add(Activation('relu'))

    model.add(GlobalMaxPooling2D())
    model.add(Flatten())
    model.add(Dense(1, activation='sigmoid', name='fc1'))

    print(model.summary())
    compileModel(model, learning_rate)
    return model

def compileModel(model,lr):
    optimizer = Adam(learning_rate=lr, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0,
amsgrad=False)
    model.compile(loss='binary_crossentropy', optimizer=optimizer,metrics=['accuracy'])
```

Code of CNN model

cnn.py

```
import tensorflow as tf
import numpy as np
import random as rn
import os
import csv

#scoring our model

def report_results(true_labels, predicted_labels, num_packets, model_name, data_source,
prediction_duration, result_writer):
  ddos_rate = '{:04.3f}'.format(sum(predicted_labels) / predicted_labels.shape[0])
  if true_labels is not None and len(true_labels.shape) > 0:  # if we have the labels, we can
compute the classification accuracy
    true_labels = true_labels.reshape((true_labels.shape[0], 1))
    accuracy = accuracy_score(true_labels, predicted_labels)

    f1 = f1_score(true_labels, predicted_labels)
    tn, fp, fn, tp = confusion_matrix(true_labels, predicted_labels, labels=[0, 1]).ravel()
    tnr = tn / (tn + fp)
    fpr = fp / (fp + tn)
    fnr = fn / (fn + tp)
    tpr = tp / (tp + fn)
    cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix,
display_labels = [False, True])
    cm_display.plot()
    plt.savefig('output.png')

    row = {'Model': model_name, 'Time': '{:04.3f}'.format(prediction_duration), 'Packets':
num_packets,
           'Samples': predicted_labels.shape[0], 'DDOS%': ddos_rate, 'Accuracy':
'{:05.4f}'.format(accuracy), 'F1Score': '{:05.4f}'.format(f1),
           'TPR': '{:05.4f}'.format(tpr), 'FPR': '{:05.4f}'.format(fpr), 'TNR':
'{:05.4f}'.format(tnr), 'FNR': '{:05.4f}'.format(fnr), 'Source': data_source}
else:
    row = {'Model': model_name, 'Time': '{:04.3f}'.format(prediction_duration), 'Packets':
num_packets,
           'Samples': predicted_labels.shape[0], 'DDOS%': ddos_rate, 'Accuracy': "N/A",
'F1Score': "N/A",
           'TPR': "N/A", 'FPR': "N/A", 'TNR': "N/A", 'FNR': "N/A", 'Source': data_source}
pprint.pprint(row, sort_dicts=False)
result_writer.writerow(row)

def main(argv):
  #command line
  ...
  #applies pre-processing methods on model
  samples = process_live_traffic(cap, args.dataset_type, labels, max_flow_len,
traffic_type="all", time_window=time_window)
            if len(samples) > 0:
                X,Y_true,keys = dataset_to_list_of_fragments(samples)
                X = np.array(normalize_and_padding(X, mins, maxs, max_flow_len))
                if labels is not None:
                    Y_true = np.array(Y_true)
                else:
                    Y_true = None

                X = np.expand_dims(X, axis=3)
                pt0 = time.time()
                Y_pred = np.squeeze(model.predict(X, batch_size=2048) > 0.5,axis=1)
                pt1 = time.time()
                prediction_time = pt1 - pt0

                [packets] = count_packets_in_dataset([X])
                report_results(np.squeeze(Y_true), Y_pred, packets, model_name_string,
data_source, prediction_time,predict_writer)
                predict_file.flush()
```

Current Day State of the Art Drawbacks.

Current detection approaches mostly use filtering (4) or rate limiting, which can help in reducing some types of attacks such as spoofing IP addresses as used by attackers to hide their identity but are not flexible when new attacks are made and can slow down website performance speeds. Reactive techniques are often required, and detection is needed to alert about the attack to perform some automatic action. (Figure 6)

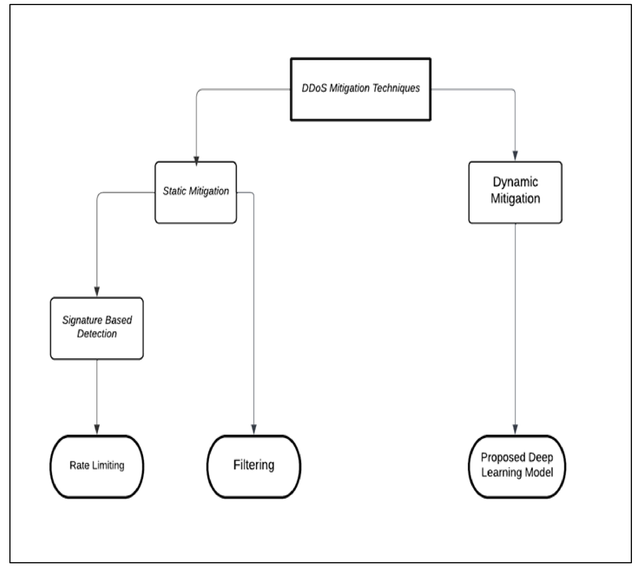

Fig. 8: Flow Diagram of the current state of DDoS mitigation techniques

Additionally, a problem in many solutions is that it is always challenging to differentiate malicious flows from legitimate flows (5). In commonplace networks, existing defense mechanisms against DDoS attacks have limited success because they cannot meet the considerable challenge of achieving simultaneously efficient detection, effective response, acceptable rate of false alarm, and the real-time transfer of all packets (6).

Future Work

1.Implement the model in a real-world environment: The high accuracy of the model suggests that it would be effective in a real-world setting, so implementing the model in a network security system would be a crucial next step.
2.Incorporate more data sources: Currently, the model is based on a single public dataset, but incorporating data from additional sources, such as private datasets or real-time network data, could further improve the model's accuracy and generalizability.
3.Evaluate performance under different types of attacks: The current model has been tested on a variety of attack types, but evaluating the model's performance on a larger and more diverse set of attack types would provide a more comprehensive understanding of its capabilities.
4.Evaluate the impact of network characteristics on performance: The network characteristics, such as network size and architecture, can impact the effectiveness of the model. Evaluating the model's performance under different network conditions would help understand the role that these factors play.
5.Consider the impact of data preprocessing techniques: The data preprocessing techniques used in this project, such as normalization and padding, are crucial to the model's performance. Evaluating alternative preprocessing techniques and optimizing these steps could lead to further improvements in the model's accuracy.
6.Address interpretability and transparency concerns: As the model is used in security-critical applications, understanding why the model makes certain predictions is important. Research into methods for increasing the interpretability and transparency of the model's predictions would be valuable.

Abstract

OFFICIAL ABSTRACT and CERTIFICATION

A Novel Supervised Deep Learning Solution to detect Distributed Denial of Service (DDoS) attacks on Edge Systems using Convolutional Neural Networks (CNN)

Vedanth Ramanathan

LASA H S, Austin, Texas, US

Category

Systems Software

Cybersecurity attacks are becoming increasingly sophisticated and pose a growing threat to individuals, and private and public sectors. DDoS attacks are one of the most harmful of these threats in today's Internet, disrupting the availability of essential services. This project presents a novel deep learning-based approach for detecting DDoS attacks in network traffic using the industry-recognized CICDDoS2019 dataset, which contains packet captures from real-time DDoS attacks, creating a broader and more applicable model for the real world. The algorithm employed in this study exploits the properties of Convolutional Neural Networks (CNN) and common deep learning algorithms to build a novel mitigation technique that classifies benign and malicious traffic. The proposed model preprocesses the data by extracting packet flows and normalizing them to a fixed length. The data is then fed into a CNN architecture consisting of a 2D convolutional layer with 64 filters, kernel size of 3x3, and kernel regularization, followed by layers regulating node dropout, normalization, and a sigmoid activation function to out a binary classification. This will allow for the model to process the pcaps effectively and look for the nodes that contribute to DDoS attacks while dropping the "noise" or the distractors. The results of this study demonstrate the effectiveness of the proposed algorithm in detecting DDOS attacks in network traffic as well as being scalable for any network environment.

1. As a part of this research project, the student directly handled, manipulated, or interacted with (check ALL that apply):
   - ☐ human Participants ☐ potentially hazardous biological agents
   - ☐ vertebrate animals ☐ microorganisms ☐ rDNA ☐ tissue
2. I/we worked or used equipment in a regulated research institution or industrial setting:
   - ☐ Yes ☑ No
3. This project is a continuation of previous research:
   - ☐ Yes ☑ No
4. My display board includes non-published photographs/visual depictions of humans (other than myself):
   - ☐ Yes ☑ No
5. This abstract describes only procedures performed by me/us, reflects my/our own independent research, and represents one year's work only:
   - ☑ Yes ☐ No
6. I/we herby certify that the abstract and responses to the above statements are correct and properly reflect my/our own work:

This stamp or embossed seal attests that his project is in compliance with all federal and state laws and regulations and that all appropriate reviews and approvals have been obtained including the final clearance by the Scientific Review Committee.