DDoS Detection Using Deep Neural Networks on Packet Flows

Jamie Weiss
Tufts University Department of Computer Science

December 2019

Abstract

Machine Learning algorithms have proven to be highly useful in industry by solving detection problems. Medical diseases and fraud credit card transactions are two examples of use cases for this technology. One of the hardest problems to solve, in cybersecurity, is combatting Distributed Denial of Services attacks (DDoS). The difficulty of mitigating these types of attacks, lie in the pure scale of the attacks themselves. DDoS attacks deal with flooding a victim server with millions of signals sent from a botnet; a network of thousands of infected computers all firing off their own signals. The victim server then has a difficult time deciphering which traffic is 'bad traffic' sent by the botnet, and which traffic is normal 'good traffic'. The ability to write rules to detect this type of bad traffic is difficult as DDoS traffic is made to mimic what real traffic looks like as best as possible. DDoS attack detection can benefit greatly from the implementation of machine learning. When trained on proper datasets, various neural network architectures have led to finding patterns within data, that humans would otherwise miss. The goal of this paper is to explore how deep learning can help detect DDoS attacks as early as possible. By doing this successfully, certain network traffic can be filtered and the victim server can thus remain alive.

Introduction

This paper will highlight the process of creating a distributed denial of service detection algorithm using deep learning. The paper will first highlight background information about DDoS attacks as well as a brief overview of machine learning, specifically the deep learning subset of it. The paper progresses into specific data preprocessing and model architecture techniques used on a large packet flow dataset from the University of New Brunswick. This study should not serve as a finished, industry-ready, product, but rather an overview of a possible methodology that can be improved upon, by others, to create an industry-ready product.

To the Community

This topic is extremely important in today's world. As the number of IoT devices is skyrocketing, the threat of distributed denial of service attacks increases as well. There has yet to be an industry standard for a process to detect these types of attack. I chose this topic because DDoS attacks are one of the biggest issues in cybersecurity right now. I am passionate about tackling these difficult problems and hope to inspire others to build upon my work to create great products for society.

Background Information

Distributed Denial of Service attacks (DDoS) are a huge problem for large corporations. Detecting these attacks early so that a cyber team can mitigate them is extremely important. As detection technologies progress, so do the technologies behind the attacks as well. A DDoS attack consists of a botnet of computers sending a huge amounts of packets to a single target system in a very short period of time. Because the attacker uses a botnet to send all these packets, this type of attack becomes especially hard to mitigate. The task of differentiating between 'normal' traffic and DDoS traffic is particularly difficult because actors will try to design DDoS traffic look as similar to normal traffic as possible to the victim system[1]. An example of how detrimental a DDoS attack can be for a company is best described in the scenario that a large bank gets attacked. During the attack, the bank will need to shut down their servers to prevent this malicious traffic from entering their private data centers. During this down-time, no client of the bank will be able to make transactions or use any of the bank's services. In general, banks can lose around \$10 million per hour that their services are down (including their reputational loss, which based on the size of the bank can be a lot worse)[2].

Different DDoS attack mitigation and detection techniques are being heavily researched due to the severity of these attacks. Artificial Intelligence (AI) is a popular, modern technology that has historically, been the foundation for highly accurate detection algorithms. AI, in general, is extremely good at finding patterns within datasets that would otherwise be overseen by the naked eye. Implementing AI alongside powerful computing power has unlocked the ability to find these abstract patterns in extremely large datasets. The specific subset of Artificial Intelligence, that performs well on big data, is called Deep Learning; especially Neural Networks. Neural Networks combine computer science with neuroscience by modeling data structures after the human brain[3]. While this may sound complicated, the idea is simple: design a data structure that will make decisions just as our brains do. To generalize the neuroscience behind decision making: the brain is composed of neurons that fire synapses to other neurons with different weights. Based on these weights - we make our final decision (to make a right turn or left turn based on what will get us closer to our destination). In neural networks, we start off by creating neurons and assigning random weights to the connections (they aren't often referred to as synapses in AI). By feeding in labeled data, the neural network learns what the optimal weights are as long as they are given the correct labels. As the model reads in more and more labeled data, it learns the correct weights better and better to the point where eventually, one can feed in data without labels and the neural network will be able to predict what the correct label is.

In this study, we use neural networks using DDoS data. At first, we tell the neural network whether the traffic it is reading is in fact malicious or benign. Then, after the model is trained, we can evaluate how well it predicts whether new traffic is malicious or benign. We hope to see a high accuracy because if there is any malicious traffic missed, the network will suffer immensely. In the next section, we will explore a dataset that contains information about incoming packets on a network where there was a DDoS attack.

DDoS Datasets

One of the beautiful things about cybersecurity as a whole is the data that is available. As mentioned before, Machine Learning is really good at finding abstract patterns in huge datasets. A lot of the data in the cybersecurity world is log data (the same set of metrics measured over large time intervals). Finding log data of DDoS attacks was an easy task. The University of New Brunswick has collected many datasets of various types of cyber attacks as well as normal traffic data on the same network[4]. The dataset analyzed in this study is a compilation of UNB's data into structured CSV files with various features extracted from another study[4].

Each row in the dataset represents a flow of packets. Many other datasets are structured in a way where each row represents a single packet delivered to a network. Using flows is better for the problem at hand. If the rows in our dataset contained only information about a single packet at a time, it would be really hard for any model to detect an attack. This is because a DDoS attack deals with more the quantity of packets over a given time span as opposed to the content of the packets themselves. Flows contain multiple packets over a given time span. It is a good way to not only capture the content of the packets, but also metrics about the way packets are arriving at the source.

First, we download the dataset from Kaggle and look at the various features that were extracted as well as some other basic information: The flows in the dataset were generated using CICFlowMeter-V3[5]. There are 6 features about each flow itself, and more than 80 features about the network traffic in each flow. There are around 7 million rows in the dataset (it will be split up into chunks for training). The dataset is populated with around 20% of the flows being malicious and 80% being benign. In the dataset, the author highlights the most important columns as FlowID, Timestamp, Fwd Seg Size, Src IP, Dst IP, Flow IAT, Src Port, Tot Fwd Pkts, Bwd Win Bytes. To evaluate how good our model is, we can train and validate on different compilations of the features after we analyze them. There are several other features, however, that we have not yet described. For more information about these features the dataset's website[6] has more information. Otherwise we will continue onto our data preprocessing and highlight other important features as we analyze them.

Data Preprocessing and Feature Engineering

In order to build a highly accurate model it is important to analyze the features in the dataset and adjust them accordingly. By doing exploratory analyses, we can not only make our model more accurate, but learn more about the underlying profile of a DDoS attack (at least the one we are working with in our dataset). First, the dataset was parsed for any 'NaN' (not a number) values. Machine Learning models, for the most part, handle NaN values very poorly. There ended up being several NaN values in some columns dealing with the average amount of packets and bytes for a given flow. These columns were dropped from the dataframe as there were many other metrics describing the flow of bytes and packets as well as the amount of time the flow lasted for. Next, there are a number of categorical columns in the dataset that need to be encoded. A popular technique to cope with this is called one-hot-encoding. One-hot-encoding will convert rows containing categories to their own column with a 0 or 1 value meaning true or false[7]. While one-hot-encoding allows for better understanding of the data by the model, it often can lead to a large number of columns in the resulting dataset, especially if the columns to be encoded have a lot of categories. The IP address columns and the Port columns are two examples of columns with a lot of categories. As a result, a new type of customized one-hot-encoding was developed that also requires a threshold to be passed in. Any category that occurs a number of times in the column above said threshold, will be one-hot-encoded in their own column, while other categories will be encoded into an 'Other' category. I developed this function for the purposes of this project, however the range of uses of this custom one-hot-encoding are wide.

Index	Src18.219.193.20	Src172.31.69.28	Src172.31.69.25	SrcOther
0	1	0	0	0
1	0	1	0	0
2	0	0	0	1
3	0	0	0	1

Table 1: An example of custom one-hot-encoded Source IP feature

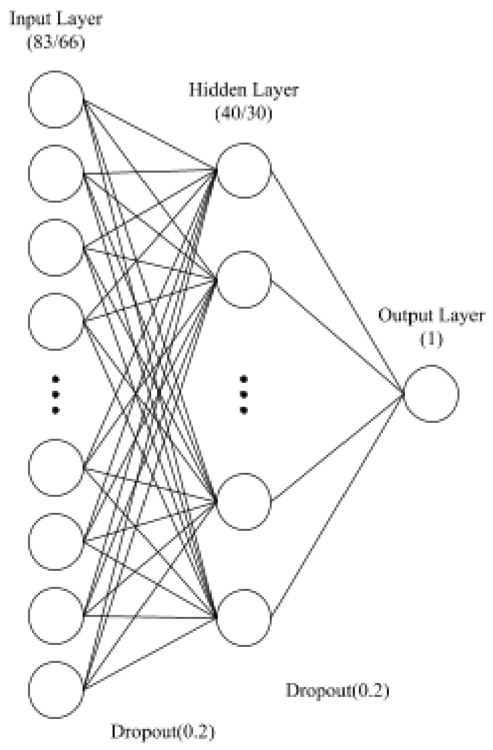
After all Source/Destination IP/Port features were encoded, we can examine the correlation each of our features have to the label. The highest/lowest correlation values represent the features that have the largest impact on the label, while the correlations closest to 0, represent those that have the least impact. Our highest correlations were the following:

Src IP_172.31.69.25	0.483251
Src IP_18.219.193.20	0.483449
Src IP_172.31.69.28	0.336337
Src Port_80	0.517200
Dst Port_80	0.432201
CWE Flag Count	0.380080

It isn't surprising that the IP addresses and the Source and Destination Ports had the highest correlation to the maliciousness of the traffic. In our models discussed later, we will build one with the IP/Port variables in the data, and one without to evaluate model performance. These features would be easy for an attacker to spoof so it is worth building two models. Features that fell below a threshold of 0.05 were parsed out as well and deemed unimportant. These unimportant features give us a good idea of different ways DDoS attackers model normal traffic. After the best features were selected, the dataset was normalized using Scikit-Learn's StandardScaler function making all values fall within a specific range. Normalizing data for deep learning will cause a speed up in the convergence of the model[8]. Now that the data is ready for the model we can randomly split off a portion of the dataset to be used for validation. The data was split such that 15% of the data was used for validation and 85% was used for training.

Deep Neural Network

Two similar neural networks were built using 2 datasets. The first dataset included all IP and Port information while the second one didn't. The reasons for this are discussed in the previous section. The deep learning package used was Keras with a Tensorflow backend. Keras provides a high level wrapper in python to construct powerful deep learning models[9]. The architecture of the models have one hidden layer, with an input layer and an output layer. After tuning the model's activation functions and optimizers, the final models were compiled and trained over 100 epochs on an NVIDIA GeForce 2080 GPU. At the end of trimming the data, the final training set had a size of 300,000 rows and the testing set had 60,000. A diagram of the model shape is shown below:



The results of the model when tested on various validation tests were unbelievable. The model with the IP addresses and ports converged at a training accuracy of 0.9990 with a binary cross-entropy loss of 0.0026 and a validation accuracy of 0.9993 and a loss of 0.0021. The model without IP addresses and ports converged at a training accuracy of 0.9926 and loss of 0.0201 and a validation accuracy of 0.9936 and a loss of 0.0182. The high accuracy of *both* models confirms the use of deep learning in this context. The models performed within 0.006 of each other when one didn't have some of the highest correlated features.

The confusion matrix for the later model is as follows:

	Predicted Benign	**Predicted DDoS**
True Benign	47150	285
True DDoS	78	9611

The fact that the model predicted a lower number of false negatives than false positives is good because it falls under the 'better safe than sorry' mentality a lot of companies have. As long as the accuracy is good, we would rather detect extra malicious packets and investigate them rather than malicious packets go undetected.

Future Works

There are many different ways to build upon the work from this study. A hypothetical use case would be to complete the same process described in this paper, for a specific network, then save the model weights and set up a network sniffer where flows can be read in real time and sent into the model for prediction. The time complexity of a prediction for a single flow is miniscule enough where it would be worth using as an alarm detection system. Some other future works can include doing a full data exploratory analysis on features and using unsupervised learning to extract further information about the profiles of various attacks.

Conclusion

In many cases, studies like this one fall under the radar of larger corporations. While it's true that it is very hard (maybe impossible) to build a model that is general purpose enough, it will work on any network, the process described in this study is one that can be recreated for any use case. All the tools used to collect features about flows of packets are readily available and most big businesses even have the ability to use more refined tools. It is also important to understand that the high accuracy of the model, should not be the only takeaway from this study. The results of the data exploration performed in the preprocessing steps, gives us insight on the different tactics attackers have been using to conduct these attacks. The preprocessing methodology in this study is also not extensive and can be improved upon drastically. In conclusion, deep learning can be used as a powerful tool to detect DDoS attacks early as well as gather more information about the attacks themselves. This study was conducted as a way to demonstrate a basic approach to leveraging Artificial Intelligence in the cybersecurity field. This paper should inspire others to build powerful real-time detection tools for various attacks.

Supporting Material

The source code for this study is available on GitHub. All information about the repository is documented in the README. The link to the repository is below:

https://github.com/jamie-weiss/DNN-DDoS-Detection

As noted in the documentation for the repository, the code in source.py is not meant to be compiled and ran immediately. It is simply code copied from a jupyter notebook and will be reformatted in future commits.

References

[1] CloudFlare. “What Is a DDoS Attack.” *CloudFlare*, https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/. Accessed 12 Dec. 2019

[2] Crosman, Penny. “Banks Lose Up to $100K/Hour to Shorter, More Intense DDoS Attacks.” *American Banker*, 2 Aug. 2018, americanbanker.com/news/banks-lose-up-to-100k-hour-to-shorter-more-intense-ddos-attacks.

[3] Hardesty, Larry. “Explained: Neural Networks.” *MIT News*, 14 Apr. 2017, news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[4] Prasad, M Devendra, et al. “Machine Learning DDoS Detection Using Stochastic Gradient Boosting.” *International Journal of Computer Sciences and Engineering*, vol. 7, no. 4, 2019, pp. 157–166., doi:10.26438/ijcse/v7i4.157166.

[5] Lashkari, Arash Habibi, et al. “Characterization of Tor Traffic Using Time Based Features.” *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, Jan. 2017, doi:10.5220/0006105602530262.

[6] “CSE-CIC-IDS2018 on AWS.” *University of New Brunswick Est.1785*, www.unb.ca/cic/datasets/ids-2018.html.

[7] Vasudev. “What Is One Hot Encoding? Why And When Do You Have to Use It?” *Hackernoon*, 25 Oct. 2019, hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f.

[8] Stottner, Timo. “Why Data Should Be Normalized before Training a Neural Network.” *Medium*, Towards Data Science, 16 May 2019, towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d.

[9] “Keras: The Python Deep Learning Library.” *Home - Keras Documentation*, keras.io/.