The use of machine learning (ML) in the detection of Distributed Denial of Service (DDoS) attacks.

David Abboud
Department of Electrical & Computer Engineering
American University of Beirut
Beirut, Lebanon
daa63@mail.aub.edu

***Abstract*—**In this paper we propose taking advantage of machine learning in detecting Distributed Denial of Service attacks. A simulation of a Distributed Denial of Service attack has been conducted on a Kali Linux virtual machine through hping3. The traffic was captured using Wireshark and exported as a comma-separated values file. The dataset was pre-processed and fed to a K-Nearest Neighbors classifier model from the sci-kit learn library of Python. The model performed well in accuracy score, and both confusion matrix and classification metrics.

Keywords*—*Artificial Intelligence (AI), Machine Learning (ML), Distributed denial of Service (DDoS), internet security, anomaly detection.

I. Introduction

Distributed Denial of Service (DDoS) attacks pose a severe problem to network, server, and end user system resources. An attacker would break into several hosts, different than the victim, that have weak security mechanisms and install slave programs onto them. These hosts are then instructed to execute the program simultaneously, targeting the victim's system and rendering their resources unusable. Not only are DDoS tools widely available, but other legitimate tools are also often abused to perform such attacks. In addition to the traffic coming from random infected hosts, packets' source IP addresses are spoofed as well, making the traceback of DDoS attacks even more challenging. A recent approach to such difficulties is the use of Artificial Intelligence (AI) in detecting these attacks [1]. Intrusion detection in internet security can be separated into three different categories: misuse detection, stateful protocol analysis, and anomaly detection. The last one, which is of interest in this paper, is statistical based: it detects an attack if the traffic's characteristics deviate from the expected normal behaviour, determined by the everyday use of network the system. Anomaly detection is a branch of Machine Learning (ML). Its premise is the recognition of abnormal patterns in learned data and the subsequent drawing of conclusions. Classification, as opposed to regression, is the main approach when it comes to anomaly detection. That is, predicting a class label (here: clean or suspicious activity) for a given sample. Training a ML model on a threat intelligence dataset which includes traffic behaviour and characteristics (time, source IP, destination IP, port numbers, etc.) allows for the detection of potential suspicious activity in a network.

This article explores a machine learning approach to improving intrusion detection by recording packets in a network and feeding them as input to a classifier model.

II. Related work

Intrusion Detection Systems (IDS) that make use of the capabilities of ML are extensively researched nowadays. In [2], authors compared different ML models as IDS backbones: Neural Networks, Bayesian Networks, Support Vector Machines, Genetic Algorithm, Fuzzy Logic, Decision Tree. Authors of [2] compared some of the aforementioned models, in addition to others. Although both papers provide no concrete conclusions since each model has its own advantages and disadvantages, [2] states that Markov models have been commonly used for packet inspection systems (system used in this work). Markov models are stochastic generative models based on the Markov property: future states depend only on the current state, and not on previous ones. Additionally, AI models have also been compared in the protection of consumer Internet of Things (IoT) devices against DDoS attacks. All algorithms tested in that environment had a test set accuracy of at least 99% [4], coherent with the results provided in this paper. Another approach was deep learning using Deep Belief Network (DBN), a class of neural networks used for unsupervised learning (trained on unlabeled datasets) based on learning the network's edges' weights and fine tuning them accordingly to achieve maximal performance. Such approach led to a test set accuracy of 92.84% [5]. A 2020 paper [6] conducted research very similar to this article's work: they implemented the same K-Nearest Neighbors (KNN) classifier to a DDoS dataset and achieved an accuracy of 98.3%.

III. Intrusion detection overview

Intrusion detection comes in three different types. The first one, misuse detection, is signature based: it analyses traffic based on a set of pre-defined criteria that are manually coded by a cybersecurity professional. In that case, traffic is only flagged if its behaviour is in violation of these regulations. One can see where this becomes an issue: the detection of unknown attacks is practically impossible [3]. Additionally, this requires immense human intervention, and is therefore costly both in time and money. The second is stateful protocol analysis. It is aware of the protocols' states [5], comparing the observed states to pre-established standard profiles provided by vendors [6]. Although misuse and stateful protocol analysis seem similar, misuse detection only brutely compares observed behaviour to a list of rules, unlike stateful systems. The latter have a deep understanding of how the protocols should work and how they should behave given their respective states [6].

However, such understanding comes with a high cost: lots of overhead. The last type of detection is anomaly detection, whose profile can either be dynamic or fixed. A fixed profile does not change once established. A dynamic profile learns while listening in to the traffic and updates itself accordingly, rendering the system prone to attacks that are spread over a long period of time since their behaviour will become part of the expected profile.

IV. Simulation & Implementation

The environment was set up on a single virtual machine using Oracle's VirtualBox. The chosen operating software was Kali Linux 2021.3 (64-bit), ran on 8GB of RAM, 2 processors and a PCIe M.2 NVMe SSD.

A. *The DDoS attack*

The tool used for the attack was hping3, which is part of the Kali Linux distribution. Hping3 is a network tool that allows sending custom TCP/UDP/ICMP packets. The difference between it and the original ping command is that the latter sends an ICMP echo request and simply waits for the echo reply and allows very minor customizations. The hping3 tool provides a wide range of custom flags: it allows extremely short intervals between packets (flooding), and source IP spoofing, among other features. For security and legal purposes, no real DDoS attacks were run. A simulated DDoS attack was implemented by a TCP SYN flood DoS attack where the source IP address was spoofed for every packet. A TCP SYN flood attack exploits the handshake process of TCP by repeatedly sending SYN connection request packets. The attack was ran using the following command:

sudo hping3 -S [target IP address] -d 120 -p 80 --flood --rand-sources

- sudo allows root access in order to open raw sockets using socket()
- -S sets the SYN flag
- -d followed by a number specifies the size of the data to be sent
- -p followed by a number specifies the destination port to be targeted, 80 here for HTTP (e.g., to be used on a website)
- --flood sends packets as quickly as possible without displaying any replies
- --rand-source spoofs IP source address by randomizing it for each packet

B. *Packet sniffing*

Upon starting the VM, Wireshark was initialized. Wireshark is a widely used network protocol analyzer that allows capturing both ongoing and outgoing packets of a network connection. Regular everyday network traffic was simulated at first by opening several different websites including streaming, video, learning, news, and gaming platforms. After about 180,000 packets recorded, the attack was executed. After the attack was run, an immense number of packets (33950 packets) was seen passing through the network to the target host, each having a randomized spoofed source IP address.

C. *Feature selection*

The features were selected through the Wireshark API. We select packet number (relative to the capture: will then be dropped, used just for reference purposes), time, source IP address, destination IP address, protocol, length, source port, destination port as our columns. Additionally, Wireshark allows the use of packet filtering. In order to properly fit our model, we will drop all packets containing IPv6 addresses, and all packets pertaining to the ARP protocol using "!ARP and !IPv6". If not filtered out, they would hinder the ability to convert all features to decimal for our model. We export the data as a comma-separated values (.csv) file and import it into Microsoft Excel. We manually add an additional column, corresponding to our target variable: Target. For each sample (packet), it is either "Clean" or "Attack". Anomaly detection generally requires substantially smaller amounts of abnormal samples than normal samples. Therefore, we cut the dataset, keeping 200,000 packets in total. We leave 2000 of 33950 abnormal, corresponding to a rough 1:100 ratio.

D. *Model training and testing*

The model was set up on Google Colaboratory (Colab), using Jupyter Notebooks. Colab provides an Intel Xeon CPU and a Tesla K80 GPU and runs Python 3.7.12. The model's architecture was implemented using the Scikit-Learn library. As for pre-processing, we firstly drop the packet number column, as it's of no relevance to the model since it's relative to the Wireshark capture. Since all features must be of a certain decimal type, we replace all dots ('.') in destination and source IP addresses by a blank character (''). Then, we assign each protocol a number, and replace the values of the Target column by 0 or 1, corresponding to Clean or Attack respectively. The dataset is then fed to a K-Nearest Neighbors (KNN) classifier model, a supervised machine learning model that can be used for both classification and regression problems, here used for classification. A supervised model corresponds to a model that deals with labeled data, where the target variable is available for each sample. A classifier outputs a discrete value corresponding to a certain class (here abnormal or normal, clean or attack) as output, as opposed to a continuous value. A KNN algorithm assumes that similar samples, datapoints, are situated close to each other and groups them together.

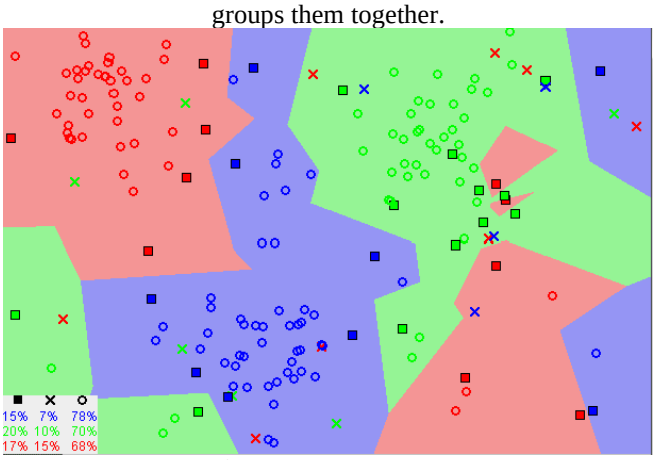

Fig. 1. Datapoints of a dataset being grouped together by a KNN algorithm.[8]

70% of the dataset is then randomly split for training, and the other 30% for testing, using Scikit-Learn's train_test_split.

E. Model performance

The test set accuracy, which is the accuracy of a model on samples it hasn't seen, using Scikit-Learn's accuracy_score, was shown to be 99.8%. The confusion matrix reported 58572 true negatives, 3 false negatives, 94 true positives, and 534 false positives. The classification scores are found below:

	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	58575
1	0.99	0.85	0.92	628
Accuracy			1.00	59203
Macro average	1.00	0.93	0.96	59203
Weighted average	1.00	1.00	1.00	59203

Fig. 2. Scikit-Learn's classification scores.

With:

$$\text{Precision} = \frac{True\ positive}{True\ positive + False\ positive} \quad (1)$$

$$\text{Recall} = \frac{True\ positive}{True\ positive + False\ negative} \quad (2)$$

$$\text{F1} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3)$$

V. CONCLUSION

In this work, we proved that machine learning can indeed be used for intrusion detection systems for DDoS attacks as anomaly detection models, with high accuracy and precision. We simulated a DDoS attack by spoofing the packets with a hping3 DoS attack, captured the packets using Wireshark, and selected relevant features for the IDS. We then exported the capture as a dataset and applied necessary pre-processing and added our target variable. With an accuracy of over 99%, ML has once again proven its importance and efficiency, which encourages more studies in merging the fields of internet security and artificial intelligence.

REFERENCES

[1] S. Haider *et al*., "A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks," in *IEEE Access*, vol. 8, pp. 53972-53983, 2020, doi: 10.1109/ACCESS.2020.2976908. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] D. Gold, ''Is signature-and rule-based intrusion detection sufficient?'' Tech. Rep., Mar. 2017. [Online]. Available: https://www.csoonline.com/article/3181279/security/is-signatureandrule- based-intrusion-detection-sucient.html.

[4] R. Doshi, N. Apthorpe and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," 2018 IEEE Security and Privacy Workshops (SPW), 2018, pp. 29-35, doi: 10.1109/SPW.2018.00013.

[5] N. Shone, T. N. Ngoc, V. D. Phai and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," in IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41-50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.

[6] Liao, Hung-Jen, et al. "Intrusion detection system: A comprehensive review." *Journal of Network and Computer Applications* 36.1 (2013): 16-24.

[7] D. Mudzingwa and R. Agrawal, "A study of methodologies used in intrusion detection and prevention systems (IDPS)," 2012 Proceedings of IEEE Southeastcon, 2012, pp. 1-6, doi: 10.1109/SECon.2012.6197080.

[8] File:Map1NNReducedDataSet.png. (2021, November 16). *Wikimedia Commons, the free media repository*. Retrieved 07:10, December 2, 2021 from https://commons.wikimedia.org/w/index.php?title=File:Map1NNReducedDataSet.png&oldid=607523785.

YouTube link: https://youtu.be/TXfVEbY7jiQ