Proposal - A Deep Learning approach to DDoS and malicious traffic detection

by Pascoli Massimiliano
AAU id: 12138922
e-mail: mapascoli@edu.aau.at
date: 5/11/2022

Motivation

Internet and its infrastructure, from its birth, has granted almost endless possibilities in the realm of computing. It is one of the most important inventions of humankind and a a powerful tool that potentially everyone can use at a fairly low price and effort. With the passing of time, many more devices were connected to the global network, in particular with the boom of IoT and decentralization. Many of them serve also in critical tasks and it is crucial that they are reliable and always accessible by their owners and legitimate users.

It is also true that, with so many machines connected, a consistent part is vulnerable to cyber attacks that are growing in frequency and intensity, because of valuable information/services stored on or provided by computers.

If for intrusion detection, exploits and misconfiguration of such devices there are easy and applicable solutions (stronger crypto algorithms and passwords, firewalls, keep systems up-to-date, utilizing only trusted software, ...), for DoS (denial of service) and DDoS (distributed DoS) attacks there are not.

The problem with DoS is that statistical-based methods rely on previous knowledge of network flow, this nowadays is too much variable. Hence, it is an almost impossible task to characterize the network traffic correctly. Most of the statistical DDoS detection methods are highly dependent on various user-defined thresholds and with a changing flow, these numbers must be adjusted continuously. There are some methods that measure entropy, this requires extensive network knowledge. To detect DDoS attacks Shannon entropy is used and this entropy detection uses only one feature like source IP addresses to create the detection model. Attackers can easily manipulate source IP address using tools like hping (*Catak and Mustacoglu 2019*) or HOIC (High Orbit Ion Cannon - that I know very well, specifically designed for distributed DoS, "spiritual" successor of LOIC - Low Orbit ...). Most of the statistical approaches like entropy, correlation, etc. ; take excessive computational time throughout DDoS attack detection. Therefore, they cannot be carried out in real time (*Hoque et al. 2017*).

These considerations leave out deep learning approaches: with their flexibility and adaptability they can be implemented real time and are not dependent on the network they are deployed in. Potentially they could also learn time patterns, in order to exploit time as another variable to better classify the traffic (especially helpful for low volume attacks or exploit attacks, see below).

I personally manage and use several services online, open to the public and I have been victim of small scale denial of service attacks in the past, so I find this topic very interesting. It is quite frustrating not being able to use or access remote machines to get things done when you most need them...

What is (Distributed) Denial of Service and enphasis on (D)DoS in all of its forms

This kind of attacks are the most common on the network infrastructure, but there is not a public-domain definitive solution that anyone could implement: this is mainly because they are diverse and because only an early detection and mitigation was possible before deep learning. Another reason is surely that a global defence against this attacks would be costly because that would mean to replace all current network infrastructure with software defined networking (some private big companies offer this in their semi-private secure network - Google, Amazon, Cloudflare).

(D)DoS attacks work either by:

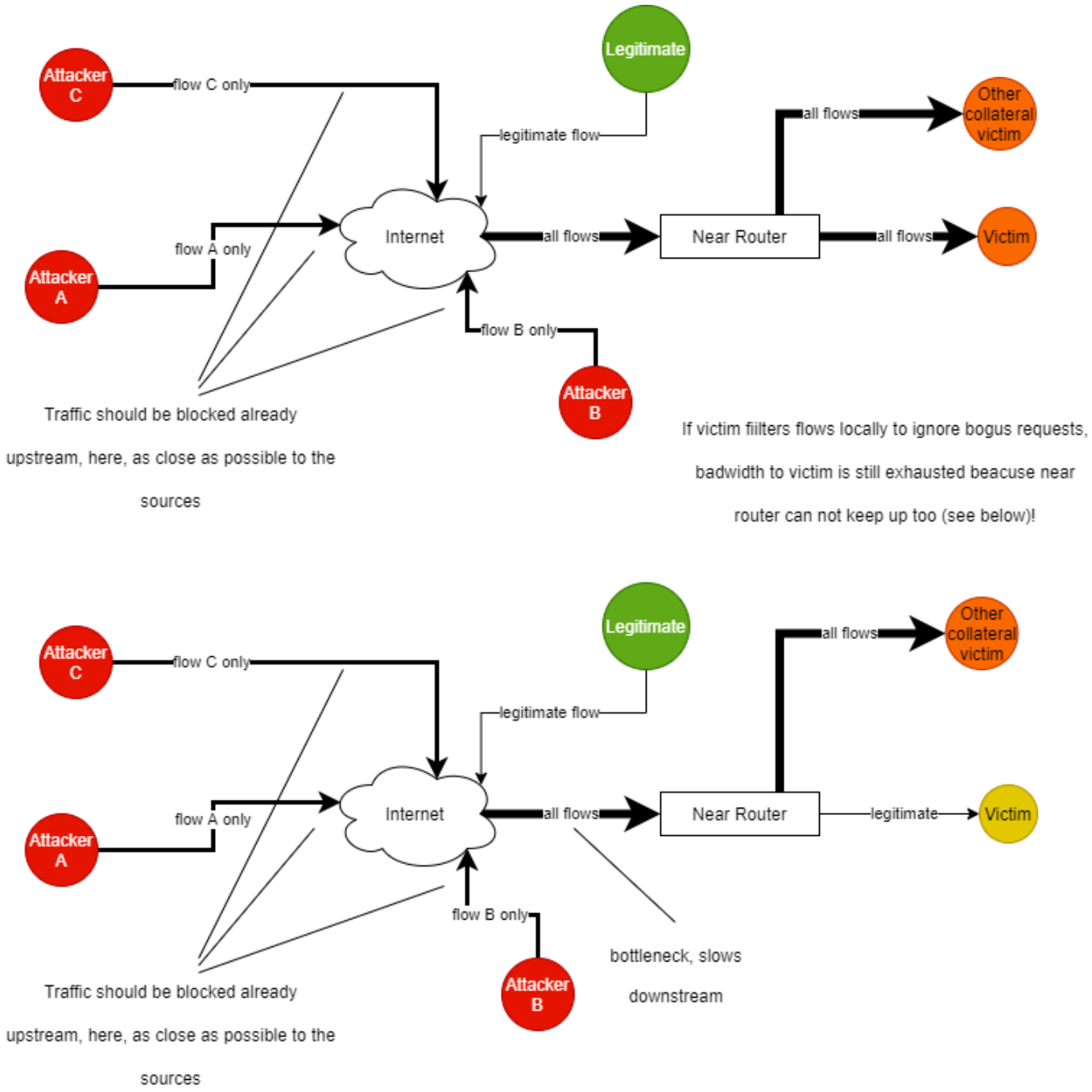
- flooding a victim of intense, bogus, ad hoc forged traffic, so that after some time it is unable to keep up with the requests degrading the performance also for legitimate requests due to bandwidth exhaustion OR
- they achieve the same by exploiting inherent vulnerabilities of protocols, for example traffic congestion management of TCP/IP protocol done with time-windows and time-outs (usually these attacks use very low traffic volume and are more difficult to detect)

Denial of service type attack only compromise accessibility to the data, not confidentiality nor integrity. Nevertheless, also e.g. old-fashioned brute force attacks that try to recover a passphrase (thus, try to compromise confidentiality), can lead to collateral denial of service due to the traffic generated: DoS, in this case is only a "byproduct" of another attack and stopping sources of the attack traffic is beneficial nonetheless.

What is my goal with this project

Every DDoS attack uses communication links between connected machines. It is possible, also manually for very small unorganized attacks, to list all active connections metadata and incoming internet packets; search for typical patterns in traffic flow (usually abnormal values in packets/sec and others) that resemble an attack and block, or limit rates of specific flows

through a firewall. This could be very well implemented on end computers, but to deal with high traffic volumes every device should do its part. If this is not the case, eventually bandwidth of last nodes near the victim would be saturated anyways. Filtering legitimate traffic upstream is necessary as shown in the example picture below.



My final goal here, is to create a software system to automatically detect which connection (flow) is part of an attack to facilitate the application of desired countermeasures, blocking malicious traffic and leaving legitimate one pass. This software could potentially be run on all nodes of the network/graph.

Since taxonomy of attacks is too large and the features to check are very diverse, the classification will be done in a deep learning fashion using a neural network.

Learning Task

I proceed here to define what the learning task will be for the system presented briefly before.

- Task T = the system should be able to distinguish if the connections flows belong to a (D)DoS style attack or are just benign traffic. If the flow is malign, possibly know what kind of attack it is (multi-class classification).
- Performance P = minimize the number of classification errors, in particular between benign traffic and all the other classes. If a flow is malign it should really be, it is less important that the attack technique (represented by a name e.g. "Slowloris", "UDPFlood", etc.) is recognized correctly (nevertheless a correct classification can be very informative to apply further, more precise, countermeasures). The system should not block legitimate requests! If that is the case, the system would not allow a legitimate user to connect to the machine causing an unwanted similar effect of a DoS attack...
- Experience E = datasets containing information of many generated flows, labeled as benign, specific attacks or general DoS. The data I will use was generated with this learning task in mind: it was recorded on a test network diverse enough to include several possible attack devices and victim devices; different types of attacks and the most recent is quite up to date in the DoS techniques (2019). I will combine several datasets, older and newer, with the same set of features that were generated in the same way to achieve a larger, more inclusive, more balanced dataset if needed. In particular, raw packets were captured by a OS independent library called PCAP that is available for use in many programming languages and then, using CICFlowMeter some metrics were computed (please refer to this page to know what metrics this tool can calculate given raw packets). Around 80 of these metrics were chosen to be present in the datasets. I will use, among others that can be found here (consider only DDoS and related attacks datasets):
  - DDoS Evaluation Dataset (CIC-DDoS2019)
  - CSE-CIC-IDS2018-AWS
  - CICIDS2017
  - CIC DoS dataset (2017)

Therefore, the learning task defined is the triple $\langle T, P, E \rangle$.

Plan

In this section I would like to describe the general approach to solve this learning task, naturally some changes will occur if something fails or is found to be unfeasible.

The realization of the system will go as follows:

1. Download several datasets to combine. These datasets will include older and more recent traffic flows generated in a test network.
2. Understand what each feature represents and tidy up the data if it is not formatted correctly (e.g. remove NaNs, reformat some columns, ...)
3. Depending on how much observations I have for every class, try to balance the classes, drop some data, collapse certain classes entirely in label "general attack" or set up a weighted oversampler of some kind
4. Split the dataset randomly into training, validation and testing observations
5. Train the neural network in batches of training data to recognize several kinds of attacks, validating on the validation set thanks to performance metrics calculation
6. Test the network on the test data and evaluate the post-training performance of the neural network
7. Hopefully I have a system that is accurate enough (if the final dataset was balanced, performance will be measured with simple accuracy, F1-score otherwise).

The model has to be very accurate on classifying attack traffic and benign traffic, also because an error of only 1% on large number of connections (recent routers manage hundred thousands, if not millions per second) results in large number of flows potentially blocked or allowed when they should not.

Related Work

I am not the first to deal with this topic, others asked themselves if a deep learning way is possible, especially now that attacks are growing in danger (registered up to 46 million req/s on Google Armor for a 2,54 Tbps traffic flow on 1st June 2022) and frequency. In this chapter I will provide a summary of works that I have found.

First of all, I would like to mention this work that makes a good job in analyzing the data. If I find out the feature are too much to handle, I could remove some of them (the less informative) by looking at the correlation: in fact, if two feature are too correlated, having both of them is useless, it is less informative, so I potentially could collapse them in a single feature that combine the two in some way or leave only one. That said, neural networks have been created also to remove (partially) the need of feature engineering, it will all depend on if my model manages to converge to something useful with more or less human "help".

In my search for materials that could be useful for this project I found, again on Kaggle,this. This work is the first that I found that resolves a similar learning task (simplified to binary classification) on CICIDS2017, applying a variety of techniques. It is interesting to notice that a LSTM (put simply, a recurrent NN with extra customization and hyperparameters to manage "persistence" and influence of past experience) is used: this NN is capable to learn time relations between the data. It is applied on a quite old dataset so I could start from this work to create a better model trained on more up to date data. The performance of the NN presented on Kaggle is quite impressive, nonetheless.

The literature is rich of other examples, as found in this very interesting and recent (27th January 2022) paper analyzing the current state of attack detection using machine learning techniques. It is worth mentioning that in several cases a small amount of epochs were necessary to obtain extremely good performance. It also provide a list of well studied and crafted datasets in which figure also the ones mentioned above: from that I can expand, if needed, the variance of samples and see how other researchers, even very recently, have built their models on this data.

Finally, I would like to spend some words for this paper. In this article there is a comparison between different approaches in the deep learning realm utilizing autoencoders, to master the overwhelming number of features that the dataset has. In the paper three are described: convolutional, bidirectional and classic. Again, simple models without many layers reach more than acceptable performance on the simplified, binary classification problem. The analysis goes further by combining the bidirectional autoencoder with different activation functions and, then, measuring performance metrics, stating that this kind of approach is applicable for DDoS detection in a software defined networking (SDN) scenario.

Risk management

In this section I present what is going to be an alternative in case the construction of the original neural network fails, or it is not sufficient.

If the multi-class classification problem fails I fallback to building a, possibly, simpler model that only has to distinguish between benign/attack traffic flows. This now should be easier, being a binary classification problem with respect to the original objective. As seen in the papers, this is definitely possible and could work very well, since many others have tackled with the challenge.

If everything else fails, the situation remains as is, and a user should implement already available DoS mitigation and early detection techniques (mainly statistical ones) with all the difficulties that arise to set up a more "traditional" defence system.

As a practical DoS defence, I bring Cloudflare network that applies sophisticated, yet classic, approaches: it provide a network capable of 172 Tbps so that none can (easily) exhaust its

bandwidth in any point (no bottlenecks, see picture) and statistical methods on load balancer servers to shield e.g. a website (website IP is hidden behind Cloudflare servers that expose their IP, so no attacker can DDoS the website IP directly because it is not known). The problem with Cloudflare is that it is not completely free...