ShieldNet Classic, Neural & Ada: An omni-comprehensive and customizable framework to DoS and malicious traffic detection

by Pascoli Massimiliano
AAU id: 12138922
e-mail: mapascoli@edu.aau.at
date: 10/03/2023

Outline

This project started as a proof of concept application and ended as a complete framework to deal with DoS and malicious traffic detection coming from external sources. The framework unifies two paradigms: a classical local way with *iptable* firewall management and a more modern deep learning approach. It is highly customizable; it was originally thought as a double layer DoS traffic filter, but can be easily extended to other problematics: this requires a new deep learning model, possibly with other metrics to compute; although the infrastructure in place remains the same. The utility can be applied, and is not limited to, end hosts or routers.

The source code, as well as complementary material, is publicly available on our GitHub repository: MiciomaXD/ShieldNetAdaptor.

Introduction

Internet is both a fascinating and truly dangerous place. It is an agglomerate of more or less valuable data and information, flaws, possibilities and conflicting interests. From its birth, it appeared clear that Internet is a tool that anyone can use at low costs and potentially high gains: this is especially true when the services offered online deal with highly sensible data.

Since anyone can connect his infrastructure with the outer world and given that the number of devices connected is always increasing at an astonishing rate, it is statistically very likely to find a vulnerable service or device. Many of those operate in extremely critical scenarios, in real-time or, perhaps, with confidential data.

Attacks are surely growing in frequency and intensity because the ever increasing value of online assets is very appetizing to malicious parties. But, not only these critical nodes are targets of attacks. In fact, to an attacker, every little bit of computing capability is useful to pursue his own goals: usually the latter are even more sophisticated attacks against stronger security measures of critical nodes in order to gain financial revenue in some ways (e.g. selling data).

There are several attacks to vulnerabilities that undermine confidentiality and integrity of the asset and there are many straight forward solutions, in the majority of cases easy to implement. To cite some NIST and NSA guidelines these countermeasures can be as easy as:

- adopting up-to-date crypto algorithms,

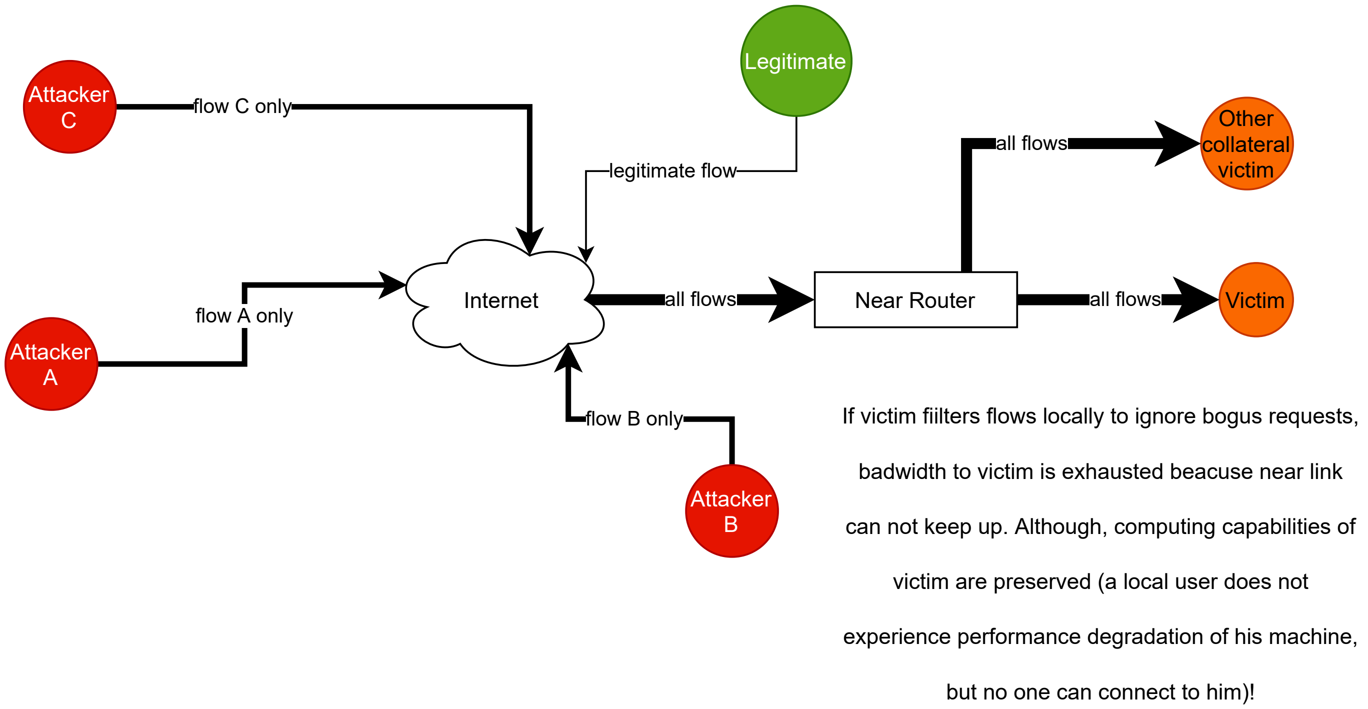
- stronger passwords with 2FA,
- configuration of firewalls,
- updating systems to latest releases and patches,
- segmentation of roles and responsibilities,
- utilizing only trusted and signed software,
- periodic cybersecurity assessment (IT auditing, penetration testing, cyberthreat risk management, ...)

The ones mentioned above, are pretty standard solutions, applicable to all intrusion detection, exploits and misconfiguration problems, but there are other threats that are directed to block accessibility of an assets: it can be very frustrating having the data, but no one, even the legitimate parties, can access it.

This is the case of Denial of Service (DoS) style attacks that were the main reason for this project idea. In this paper, complemented by the Python files that can be found in the repository, we will explain how the framework created, code name ShieldNet (SN in the following), works and offers a multi-layered mitigation to DoS attacks that can be extended to other cyberthreats too. The framework was created having in mind Linux family OS and iptables, but can be easily adapted to work with Windows machines (mainly path representation changes and a different firewall interaction).

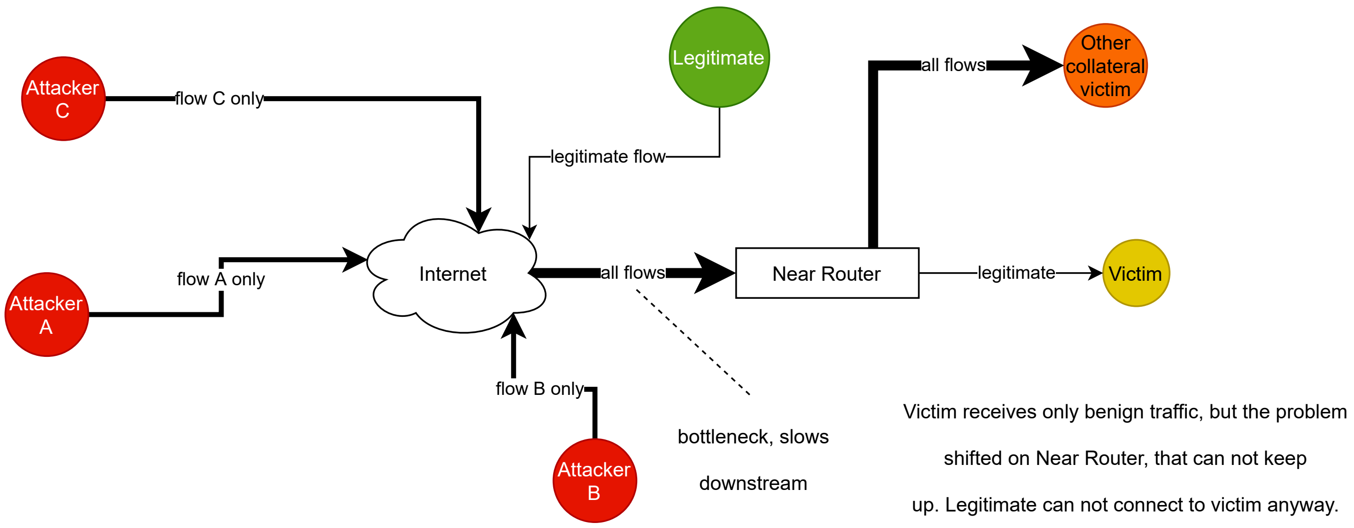
The double-layer defence by SN to be effective should be installed on routers, as upstream as possible to DoS traffic sources. Check the explanatory diagram below in this regard.

Scenario 1: victim is filtering locally

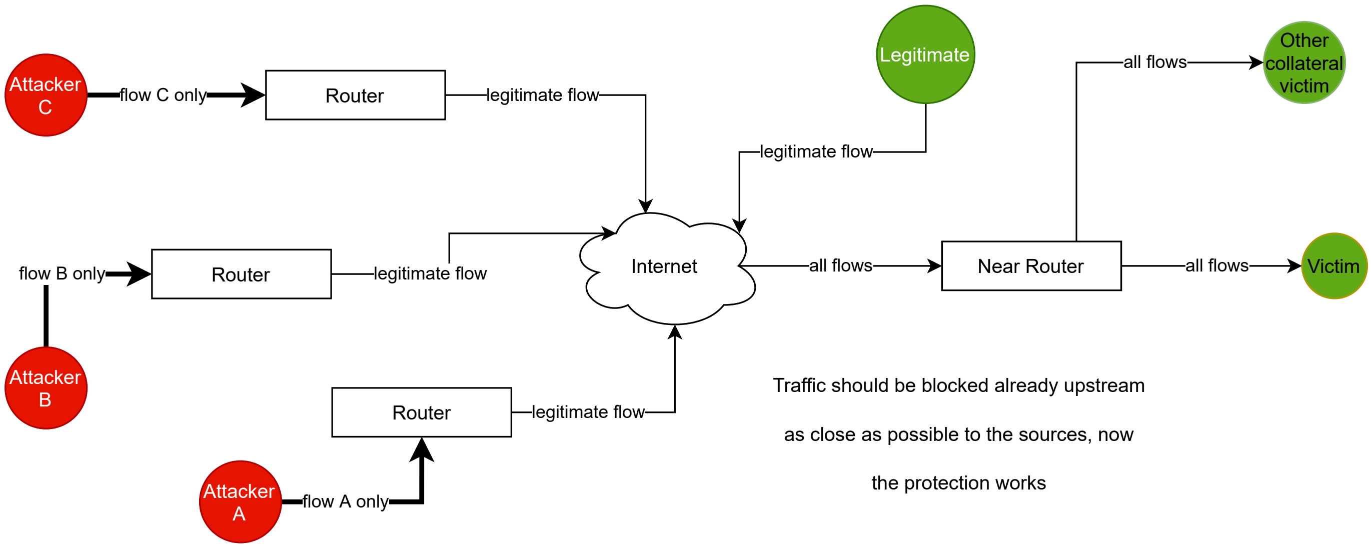


If victim fiilters flows locally to ignore bogus requests, badwidth to victim is exhausted beacuse near link can not keep up. Although, computing capabilities of victim are preserved (a local user does not experience performance degradation of his machine, but no one can connect to him)!

Scenario 2: only neighborhood is filtering


Attacker C
flow C only
Legitimate
legitimate flow
all flows
Other collateral victim
Attacker A
flow A only
Internet
all flows
Near Router
legitimate
Victim
flow B only
Attacker B
bottleneck, slows downstream
Victim receives only benign traffic, but the problem shifted on Near Router, that can not keep up. Legitimate can not connect to victim anyway.

Scenario 3: filtering upstream



In this paper we will cover briefly the realization of the deep learning model, since it is already presented in detail in our other work available at GitHub - MiciomaXD/MacLea_DeepLea_Project_Klagenfurt. Please, for a detailed reading regarding ShieldNet Neural (SNN) check proposal, presentation and final report produced in the scope of Alpen-Adria-Universität Klagenfurt course "Machine Learning and Deep Learning (650.025)" by professor Schekotihin Konstantin - semester 22W. This is because the core part of SNN is one of the many models produced in that setting.

We will explain in more details ShieldNet Classic (SNC - the part of ShieldNet that deals with Internet traffic without a neural network; this is done applying standard, field-tested filter rules) and ShieldNet Ada(ptor) (SNA - the part that aggregates and maintains up-to-date data for later use in SNN).

Related literature: limits, acknowledgements, techniques

First of all, what exactly is a Denial of Service (DoS) attack?

As the reader could guess from the name, in its simplest form a DoS attack is a procedure aimed to make a victim unreachable, hence unable to offer its services to legitimate parties or at least degrading performances greatly.

The way this is achieved is by exhausting victim network bandwidth or connection capabilities through bogus requests in a plethora of ways. The taxonomy of DoS and DDoS style attacks (the most common and simple, "Distributed" variant of DoS, with several attack machines and high traffic volumes e.g. SYN flood technique) is vast and classic statistical methods struggle to keep up. This is happening in a settings where organized attacks reach the overwhelming magnitude of up to 46 millions requests/s on Google Armor for a 2,54 Tbps traffic flow on 1st June 2022, just to cite one example.

To make the situation even more problematic, not all attacks use high traffic volumes and thus are easy to identify, but some of them exploit intrinsic flaws in Internet protocol / applications specifications and implementations (e.g. Slowloris technique with low traffic volume or using TCP shrinking window and time-out-based retransmission for congestion management as explained here).

Finally, unlike other attacks, DoS is technically always possible from the very first moment a machine is publicly available.

Statistical solutions

So how can someone defend or mitigate his subnetwork from these devastating DoS attacks?

Literature presents several statistical approaches with the same background idea. First, an average network behaviour model is computed, dependent on the setting we would like to defend. Then, we focus on abnormal behaviour that diverges a lot from the "average model".

The problem with this, is that the aforementioned average model is constructed on general and user-defined thresholds. With a very variable flow, it is an almost impossible task to characterize a proper model without continuously adjusting those thresholds.

Many (if not all to our knowledge) classical approaches are based around Shannon entropy and Pearson's correlation to generate a mathematically accurate representation of the network using features like IP addresses. This requires a lot of information and extensive network knowledge on its own and it is difficult to carry out in real time (Hoque et al. 2017).

Attackers can easily manipulate and forge ad-hoc traffic to bypass these countermeasures with tools such as *hping* or *HOIC* (popular successor of *LOIC* with DDoS capabilities).

Most successful methods used for DoS detection are recent machine learning methods: Support Vector Machines (SVM), k-Nearest Neighbour (kNN) and Naive Bayes Classifier (NB). As stated in Deep learning approaches for detecting DDoS attacks: a systematic review the mentioned methods are not efficient.

DL approaches

Neural networks can be the solution to DoS detection and this project wants to be yet another very practical confirmation.

Their adaptability and flexibility can be implemented real time under some constraints, especially if we account for GPU accelerated computing. In addition to that, very simple models, proposed also by us in the previous cited work, are expressive enough to capture complex attack patterns with ease.

To make a quick roundup of related work proposed by other researchers (treated in more detail in our complementary work), it is possible to distinguish two ways of proceeding:

- the first, involves the evaluation of a model for every single incoming packet. As examples we bring: CNN-Based Network Intrusion Detection against Denial-of-Service Attacks and DDoS attack detection and classification via Convolutional Neural Network (CNN). Both apply a clever packet to image conversion (bit to BW pixel or byte to greyscale pixel) and than perform classification using CNNs.

- the second, requires the a priori aggregation of Internet flow metrics and then the classification of flow statistics. As examples:

    - GLD-Net: Deep Learning to Detect DDoS Attack via Topological and Traffic Feature Fusion - PMC uses GNNs with attention. The model representation of the network is a connected undirected graph, similar to classic methods, where every node is a host. Every edge of the graph contains flow metadata (e.g. bytes/s, number of flags, up and down rate, ...) that can be checked and taken under consideration by the GNN layer to classify every flow between two nodes $\langle n_1, n_2 \rangle$. This approach is particularly interesting because GNN use a representation that is very natural and conceptually easy to read. Furthermore, to classify a flow between $\langle n_1, n_2 \rangle$ with GNNs, also adjacent pair $\langle n_2, n_3 \rangle$ data can be used to obtain a perhaps more accurate classification even if node n_1 is not even aware of the existence of n_3: this is characteristic of GNNs, context is extrapolated and summarized following a recursive criteria from the neighbourhood of a node up to certain extents and collapsed together with the local data, then a classification is made on the combined metrics.

    - Improved DDoS Detection Utilizing Deep Neural Networks and Feedforward Neural Networks as Autoencoder make use of powerful autoencoder paradigm to shrink down model size, hence optimizing performance. Starting from a high number of features (a high-dimensional feature space), autoencoders are a great way to shrink down input dimensionality maintaining and selecting useful information to reproduce the input itself starting from the summary. This gives to autoencoders the classic and recognizable hourglass shape. If it is possible to start with simplified input, training of a classifier will be easier and the classifier itself will be smaller in size, boosting efficiency during deployment.

  - Binary classification using SMOTE+LSTM | Kaggle. This is an approach task that has potential, followed by a user on Kaggle facing DoS detection. Another dimension that can be considered is, in fact, time. Metadata of a flow changes throughout time; how the change happens between two "snapshots" of a flow can be a distinguishing factor between a malicious or a benign connection. LSTMs are used in this setting to account for the time dimension, consuming the dataset as a series of sequences. In addition, this implementation uses Synthetic Minority Oversampling Technique to generate artificial samples where is needed to rebalance the dataset.

Every approach listed above is capable of 99+% accuracy, setting the state-of-the-art standard, reached and in some cases, exceeded, by our, even smaller, model.

The approach we followed in SNN, was the "aggregation first than classify" one. The reasons for that are mainly for time and memory efficiency. An evaluation for every packet would kill connection throughput, so to classify every packet is not feasible. Secondly, GNNs and LSTMs requires extended network knowledge (a "master router" that knows everything is its neighbourhood) or to keep in memory more data per flow (every flow has multiple "reincarnations" that describe the trend through time).

Autoencoders, on the other hand, were used in our model production.

Installation of Shield Net

SN is a multi-threaded interactive application, that uses Linux firewalling utility "iptables" to achieve the desired filtering of the internet traffic. Furthermore, the caption of the packets (sniffing) is a very low level process that has components running in kernel space. Hence, it is necessary that the user running the scripts has root permissions when starting the application.

In addition to that, we advise to use a virtual environment created with tools like Miniconda in order to manage the required libraries installation. In addition to Python 3.10.X, SN needs:

- Pytorch for all that concerns the deep learning layer of SN (i.e. SNN);
- Pandas and correlated NumPy / SciPy stacks for internal data metrics computation and memorization;
- Scapy for packet management, decoding, sniffing, analysis.

Once that is done, it is sufficient to run

```
python3 daemon.py
```

to start background packet processing. When needed, a human readable internal representation of the current state of SN can be obtained with

```
python3 command_interface.py situation
```

or SN can be safely stopped in a controlled manner with

```
python3 command_interface.py stop
```

Please, note that upon stop command execution, iptables will be reset to the situation previous to the start of SN, so any preconfigured set of rules will remain untouched (SN works on custom iptables chains created specifically for it at start-up, that flank seamlessly already specified rules - more on that in the following).

ShieldNet: a panoramic of the whole infrastructure

SN is composed by three major components

ShieldNet Classic: classic vanilla defence and firewall connections management

We were attacked udpflood managed no downtime around 1Gbps

ShieldNet Neural and ShieldNet Ada: deep learning with built-in metrics computation and firewall connections management

Results and future steps

In this project we explored the possibilities of DoS detection and mitigation, realizing a working framework that would allow, not only big corporations, but also common network users like *we* are in the first place, to defend our online projects against this and other cyberthreats. All of this, at a low price.

The project is far from complete! Our future steps include:

- the realization of a simple to use GUI to monitor the situation internal to SN;
- polishing of python code, making it extendible, more comprehensible and portable, so that the utility is easily installed and simply works well;
- porting of SN to other, more efficient, programming languages (maybe C language): this is not only to gain from the performance point of view, but also to make SN installable in a capillary manner on every network node in order to make the protection effective (see the 3 scenario diagrams at the beginning);
- testing in a real world environment, with more diverse data and attacks;

- extensions of SNN core to detection of other non primarily DoS attacks (e.g. brute forcing, infiltration attemps, ...).

Traditional methods remains, and services as Cloudflare offers already a great protection against DoS, but we hope this project is a starting point or at least inspiring to those who want to expand on the topics presented.

Finally, we would like to thank professor Schekotihin Konstantin for the useful advices provided during the creation of ShieldNet Neural.