Detecting DDoS attacks using a multilayer Perceptron classifier

Christos Siaterlis (csiat@netmode.ntua.gr)
Basil Maglaris (maglaris@netmode.ntua.gr)
Network Management and Optimal Design (NETMODE) Lab
Dept. of Elect. and Comp. Eng., National Technical University of Athens
Iroon Politechniou 9, Zographou, 157 80 Athens, Greece

March 2004

Abstract

Detection of Distributed Denial of Service attacks should ideally take place near their sources, at edge networks, where countermeasures are most effective. DDoS detection by monitoring an over-provisioned backbone link either near the source or the victim is challenging because congestion isn't the identifying anomaly signature. Most research efforts try to identify a single detection metric that can reliably detect DDoS attacks. On the contrary, we use multiple metrics to successfully detect flooding attacks and combine them with an Artificial Neural Network (ANN).
We explore the DDoS detection ability of Multi-Layer Perceptrons (MLP) as classifiers we can teach by example. The inputs of the MLP are metrics coming from different types of passive measurements that are available today to network administrators. We use these metrics to feed our MLP, train it and evaluate its performance in terms of 'false positive' and 'true positive' rates in the face of new data. Our analysis is based on data from several experiments that were conducted with the use of common DDoS tools in the production network of a university network. Beyond detecting plain flooding attacks we show that the MLP is capable of classifying the state of the monitored edge network as "DDoS source", "DDoS victim" or "normal". This way we show that an edge network can use a single mechanism to protect itself from incoming DDoS attacks and at the same time protect the rest of the network from outgoing attacks.

1 Introduction

It seems that we are witnessing a tremendous growth of Internet threats. The emergence of multiple worms[1] and viruses that are propagating by exploiting the numerous vulnerabilities that are discovered day by day, transforms poorly administered computers into a powerful army in the electronic battlefield. The

compromised hosts can be remotely controlled to perform various malicious activities like Spam forwarding [2], hosting illegal web sites or Distributed Denial of Service attacks [1]. Malicious users that have under their control a large number of compromised hosts are able to launch packet floods towards a victim host or a router with a single command. These packet floods may aim at bandwidth starvation, at overloading a system's IP stack or a router's flow based switching module and are able to make the victim devices unreachable - denying thus service to legitimate users.

The detection of Distributed Denial of Service attacks is vital for the security management of edge networks, especially of university networks and ADSL providers. Poorly administered workstations and servers of academic networks and non sophisticated users' home computers become the main sources of such attacks. Detection of DDoS attacks near their sources is the most effective approach that has been slightly explored [3],[4]. However, detection is hard even near the victim -destination network- , especially if we monitor non-congested links, which is the case in a overprovisioned ISP backbone. In this case link saturation can't provide us with an anomaly signature. In the same time it would be economically questionable to expect ISP's to perform DDoS detection on many, small and highly utilized customer links and not just at a few points of the over-provisioned backbone.

Given the current technology constraints, the research community has failed to offer to network administrators reliable and feasible detection methods. The problem is that our sensors have to cope with high data rates which impose constraints on the detection algorithm's complexity. This way, complex processing techniques like power spectral density estimation [5], clustering algorithms [6] or wavelet analysis [7] are promising but not readily available since they are based on hard to measure metrics (at least in real time) and they involve high processing overhead.

Moreover, most detection approaches focus on single detection metrics with the notable exceptions of [8],[9]. In contrast to the approaches mentioned before, we explore the detection ability of an ANN using several detection metrics that are based on passive measurements. Our detection metrics were measured in a production network using tools and methods available today to network operators in contrast with unrealistic approaches that for example keep detailed per flow statistics. We use a Multilayer Perceptron as a data fusion algorithm to combine these heuristics. We use a machine learning approach in order to avoid magic numbers and site-dependent thresholds that would demand great investment in terms of installation and customization. As a MLP can be taught by example and fuse together several detection metrics it is a suitable basis for a detection mechanism. Our analysis is based on UDP flooding experiments and traffic measurements on a link that can sustain packet floods without severe congestion. This made the detection of traffic anomalies challenging and in the same time allowed us to conduct DDoS attacks without causing any harm to

[1] more accurately packet flooding attacks in contrast to logical DoS attacks that exploit certain OS or application vulnerabilities

the legitimate network users. We test our detection system on real and non synthetic traffic unlike other approaches that when operating in a production network might generate an enormous size of false alarms due to inaccurate modeling of Internet traffic.

The paper is structured as follows: we begin in section 2 with a description of our detection system and continue on section 2.1 with a brief introduction to the available passive measurement techniques and the features that were selected as an input to the MLP. In section 3, we present the topology and the traffic characteristics of our experimentation platform, an academic-network ISP. In section 4 we continue with the evaluation of our detection system in terms of learning and generalizing and discuss some potential improvements. We conclude in section 5 summarizing the main results of our approach.

2 Detection System Architecture

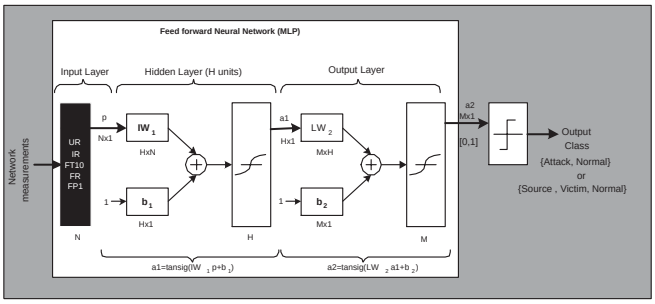


Figure 1: The detection system's architecture

One of the main points of our contribution is addressing the problem of UDP attack detection as a typical classification problem. Using the network state as input, a classifier can make an assessment whether it is a normal or an attack situation. For such classification problems neural networks are a promising solving technique. Neural networks were also used by the authors of [10] but in order to perform non-linear time series prediction and to detect UDP attacks as a large deviation from the predicted traffic value. We argue that the difficulty in detecting DDoS attacks rises from the fact that a reliable decision cannot be made based on a single metric but we should use a combination of several metrics' values. In the same time prediction of future values of network characteristics might be more difficult than our classification problem.
To address the classification problem we chose to follow a supervised learning approach. Using a carefully chosen set of network statistics as an input signal we train a multilayer (MLP) feed forward network to classify normal and attack states. The neural network configuration is very simple as it has only one hidden layer and will be analyzed in section 2.2. Figure 1 shows the architecture of the detection system that is presented in this section.

2.1 Feature set selection

To successfully detect the true network state we had to choose which features to feed into the classifier. We will present the metrics used, grouped by the measurement methodology. Before we start we would like to stress the fact that these methods are used today by network administrators to monitor their networks and show that our approach is realistic and feasible. In contrast to other approaches that keep per flow or per IP address state, our statistics are high-level (easy to measure and store) but can still indicate attack patterns.

2.1.1 Packet capturing

The most powerful passive measurement method is through direct packet capturing. The main problem of this method is that the monitor has to cope with very high link speeds that impose constraints on the complexity of the statistics kept. Specialized hardware like network processors [14] might help keeping up with increasing data transfer rates in the future.
Our packet capturing infrastructure consists of commodity hardware (Intel P4 2.4GHz with an Intel Gig. Ethernet card) and open-source software. We have developed a custom preprocessor plugin for the popular open source IDS Snort [15] that produces traffic statistics based on captured packet data (libpcap format). The statistics kept were chosen to be simple so that it would be feasible to run the plugin at high wire-speeds with minimum packet drops ($< 0.1\%$). Using these tools we are able to collect data of the incoming and outgoing UDP and ICMP packet rates and their corresponding share of the link utilization sampled in regular time intervals. The time granularity is set to 30sec so that we can detect even short-living anomalies. All the data produced by our plugin are stored in round robin databases with the use of the RRDtool [16]. The metrics that were used in the input vector are:

- UDP attacks are mainly bandwidth consumption attacks and as these traffic types generally utilize small amounts of bandwidth, sudden changes in the transferred UDP bytes/sec are good indications of attacks. An improvement over this simple approach is the UDP ratio : $UR = \frac{incoming\ bit/sec}{outgoing\ bit/sec}$ The intuition behind this metric is that although there isn't a clear symmetry in the UDP traffic as in the case of TCP, there is still a fairly stable site dependant behavior (ratio value) depending on the presence of DNS, NFS, streaming servers etc.

- The ICMP ratio $IR = \frac{outgoing\ bit/sec}{incoming\ bit/sec}$ serves also as an indication of an UDP flood because most of the times a reverse ICMP stream is generated during a UDP attack if the packets have a valid checksum.

2.1.2 Netflow data analysis

A flow is defined as a unique set of the following 5 characteristics <protocol, source IP, source port, destination IP, destination port> and makes a higher

level description of a traffic stream. This information that is kept by the routers is much smaller in size than whole packets but in the same time loses some additional information like TCP header flags. Another characteristic of this measurement type is that it is near real-time, in the sense that a flow is exported to the monitoring station when it has expired [17]. Sampling is sometimes required because a router can't keep up with the high transfer rates. To get more detailed information about the flows seen by the router we developed a Netflow collector (based on 'Flow-tools' [18]) that gathers the flows that are exported by the router, process them and calculates our detection metrics. The sampling period used was 30sec.

- Flow length distribution. The distribution of the number of packets in a flow (per protocol) can provide a good sign of a spoofed DDoS attack. A high number of flows with few packets (1-3) is a hint for DoS because of the randomness in the source addresses and ports. More specifically we use the number of flows with one packet (FL1).

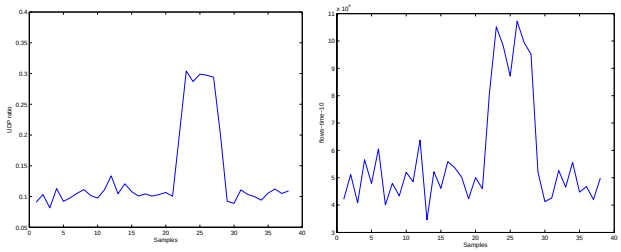
- Flow lifetime distribution. Similarly, the distribution of the lifetime of a flow in a routers cache is sensitive to DoS attacks but to port scans as well, producing thus a high number of false positives. Nevertheless, we chose to use as an input metric the number of flows with lifetime $< 10ms$ (FT10) according to the router's reports and the flow-tools utility implementation.

- Flow generation rate. In [19] was mentioned that the number of learning failures of a flow accounting algorithm was able to identify spoofed flooding attempts. The reason is that when a flooding attack occurs the amount of 'transports that are not completed'[17] is large and the entries are not removed gracefully but are filling up the cache causing flow learning failures. A similar metric that we used in this work is the flow generation rate (FR) measured in flows per second.

Based on these metrics we use the following input vector with N=5 elements: $i(t) = [UR(t)IR(t)FT10(t)FL1(t)FR(t)]$

2.2 Neural Network configuration

A Perceptron is an elementary network in which a neuron unit calculates the linear combination of its real-valued inputs and passes it through a threshold activation function. A multilayer Perceptron is a hierarchical structure of several Perceptrons. If we use non-linear activation functions (for example a sigmoid function) in the neural nodes, it can be trained to reproduce a nonlinear function mapping. To address our classification problem we chose to use feed forward network (MLP) with three layers: an input, a hidden and an output layer. As we use an input vector i ($|i| = N$) the input layer is constructed with N linear neurons. The second layer , which makes the neural network capable of recognizing M non-linearly separated classes(in our case attack and normal states), is constructed with $H = 2N + 1$ sigmoid neurons. As the theory indicates



(a) UR metric

(b) FT10 metric

Figure 2: Samples of the UR and FT10 metrics during a UDP attack.

([11] based on Kolmogorov's theorem [12]) by using at least $2N + 1$ neurons in the hidden layer the neural network can approximate any continuous function $F : [0, 1]^N \rightarrow R^M$. We conducted experiments with the number of neurons in the hidden layer and analyzed the performance of the MLP classifier in each case. Our choice is supported by the fact that even if we increase the number of hidden units $H > 2N+1$ the efficiency of our classifier isn't increased. With our choice we can recognize attacks and normal states while keeping our network small (Fig. 3). Although the exact relation between the number of neurons and the network response is not clear, we chose this value in order to avoid over-fitting.

The inputs of the network are the values of the detection metrics measured over

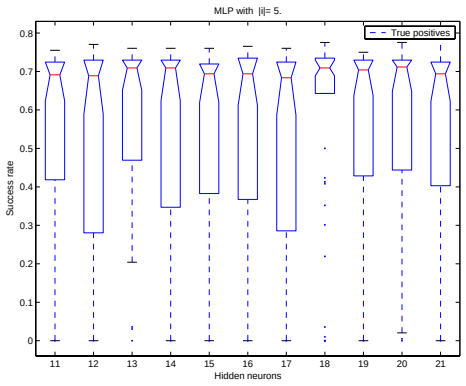


Figure 3: The effect of the number of neurons in the hidden layer on the True positive rate

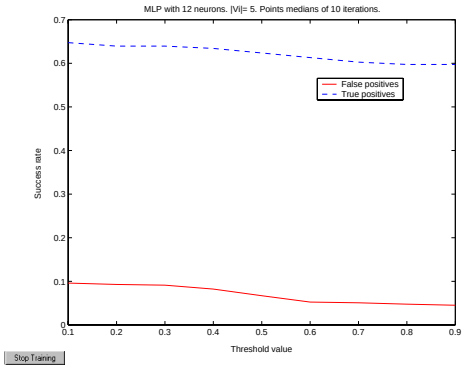


Figure 4: The effect of the threshold setting on the average detection rate in the training phase

a specific time period and were discussed in section 2.1. The size of the output layer depends on the number of classes M we are using to describe the network's

state. In the first case of simple attack detection the output layer is made of just one sigmoid neuron that generates output values between 0 and 1. We represent a normal network state with the value 0 and an attack state with the value 1. If we differentiate between "DDoS source", "DDoS victim/destination" in the case of an attack, we use two sigmoid output neurons that generate "01","10" respectively and "00" for "Normal". Because of this binary representation the output of the MLP is filtered by a hard limit function which classifies values below 0.5 as normal states and values over 0.5 as attacks. The exact value of the threshold is not affecting the detection performance as we have a good classifier with almost binary outputs(figure 4).
The equations that describe our system are: $a_1 = tansig(IW_1\ p + b_1)$, $a_2 = tansig(IW_2\ a_1 + b_2)$, $o = hardlim(a_2)$, where p is our input vector i(t) and o is the output of the classifier. The matrices IW_1,IW_2 (weights) and the vectors b_1,b_2 (biases) are parameters that are determined during the training phase. By repeatedly presenting to the network a set of inputs and desired outputs, a training algorithm (for example a backprobagation algorithm) calculates the desired values for IW_1,IW_2,b_1,b_2. The MLP simulation and the analysis of the experimental data was performed with the use of the Matlab Neural Network Toolbox. The initial values of weights and biases were randomly chosen by Matlab and this introduced a random factor in each training attempt. That's why in order to obtain a representative and consistent view of the MLP's behavior we repeated each test several times. The sigmoid transfer function of the hidden and output layer neurons was the hyperbolic tangent function that can be efficiently computed. The backprobagation algorithm that was used for training and convergence of weights and biases to their final values, is the Levenberg-Marquardt algorithm [13]. This algorithm is known in the bibliography for its fast training ability.
At this point we should discuss the feasibility of our approach as an on-line detection system. An already trained ANN can easily keep up with high input data rates. The algorithms associated with the training phase might have high computational complexity but this doesn't apply to the operation phase. This way network administrators could collect representative input data and train the detection system offline. Experience with the NTUA campus network that for a period of at least 6 months the normal values of the metrics that were used in our analysis remained in a constant interval but we can re-train the network periodically to adapt to long-term network changes.

3 Experimentation platform

To evaluate the effectiveness and usability of our detection engine we have performed a series of experiments on an academic research network. As we argued in the introduction, DDoS detection on an over-provisioned high-bandwidth link where traffic is aggregated but stays in low utilization levels holds great interest. We monitored the Gigabit Ethernet link between an academic ISP and a

large university that is a single hosted network with a fast but under-utilized upstream link. Some information of interest is that this link keeps a sustained rate of 250Mbps with peaks higher than 450Mbps and contains a rich network traffic mix carrying both standard network services like web traffic, but also peer-to-peer application traffic, online games, as well as streaming audio and video traffic. This fact is significant because some detection algorithms might work fine in simulation or lab-testbed experiments, but their high false alarm rate when facing real traffic renders them useless. We conducted more than 36 experiments over several days during business hours and with background traffic generated from the more than 4000 hosts of the university campus. Our total sample size was 2159 points that corresponds to approximately 18 hours. In the first experiment scenario the victim was located inside the campus with a 10Mbps link whereas the attacker was outside the campus coming directly from its ISP. The attacker was connected to a 100Mbps interface to simulate the aggregation of traffic from several attacking hosts (Fig. 5). Using a modified version of the well known DDoS tool TFN2K [20] we performed a series of UDP flooding attacks with spoofed IP's, 3 minutes duration and controlled packet and bytes generation rates. In the second experiment set the roles were reversed and the university network became the attack source. In table 1 we show the combinations of attack characteristics. Note here that the attack traffic rates is only a fraction of the normal UDP traffic. The attacks used the common method of selecting source addresses from the attacker's real subnet in order to bypass any e-gress or RPF filtering.
In figures 7-6 we see the range of values each metric takes in the case of attacks and in normal states. We divided our samples into two classes : attack and normal states and plotted them without chronological order. On the left side we see normal states represented by circles and on the right side attack states represented by crosses. With this diagram we highlight that the attack and normal classes are not linearly separated. Although there is a definite trend, for example high values of the FT10 metric is in most cases an attack indication, there is still an uncertainty. In section 4.4 we see the gain of using an MLP in comparison to simple thresholds on single metrics.

Table 1: The combinations of attack characteristics (UDP traffic only).

Edge network state	# Attacks	Bandwidth range (Mbps)	Packet rate range(pps)
DDoS source	16	1-6	1000-6000
DDoS victim	32	1-6	1000-6000
Normal	-	12 - 30	2500-4500

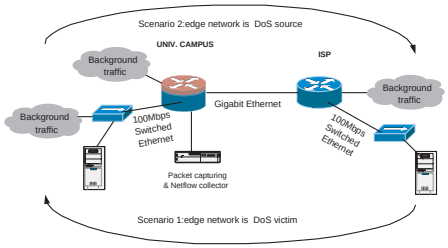


Figure 5: The topology of the experiment setup

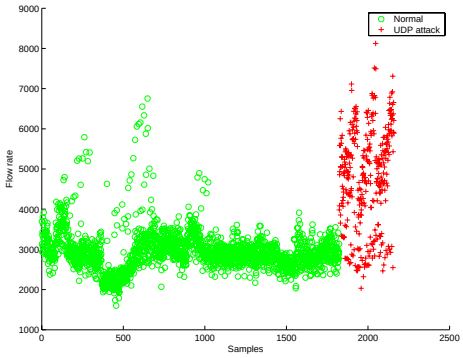


Figure 6: The FR metric

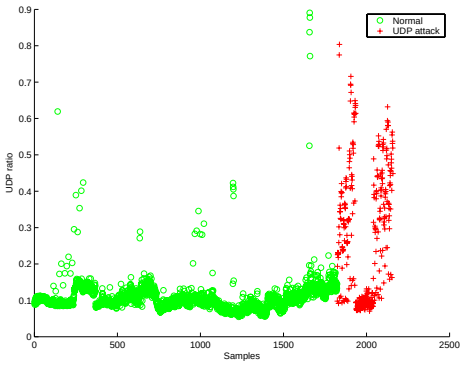


Figure 7: The UDP ratio metric

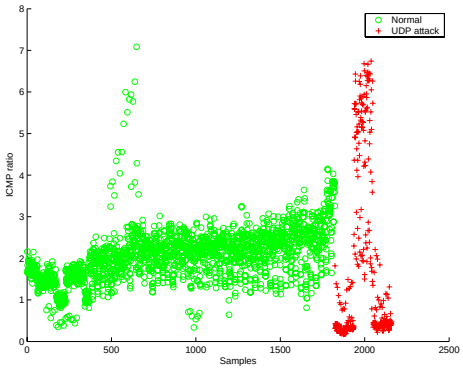


Figure 8: The ICMP ratio metric

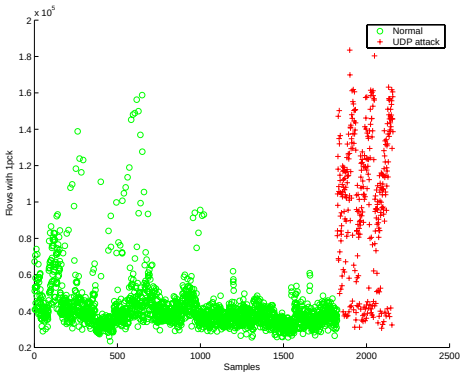


Figure 9: The FL1 metric

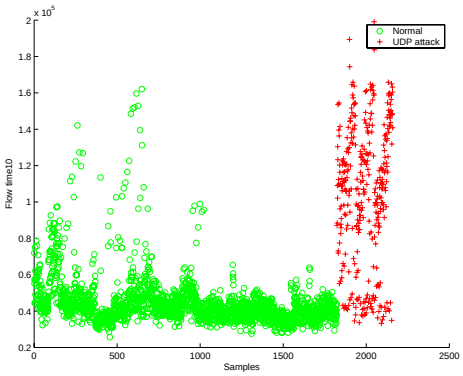


Figure 10: The FT10 metric

4 MLP detection performance

4.1 Performance Evaluation Criteria

We will borrow the following definitions from traditional detection theory to evaluate the performance of our detection system: TP (true positives) = # alerts (test positive) and attack is present , FN (false negatives) = # no alerts (test negative) and attack is present, FP (false positives) = # alerts (test positive) and attack is not present and TN (true negatives) = # no alerts (test negative) and attack is not present.
Our study of the MLP detector's performance focuses on the following two metrics: the rate of true positive alarms (sensitivity), which is the number of alerts when there is an on going attack to the total number of attacks: $TPR = Sn = \frac{TP}{(TP+FN)}$ and the rate of false positive alarms (1-specificity), which is the number of alerts when there is not an on going attack to the total number of normal states: $FPR = 1 - Sp = \frac{FP}{(TN+FP)}$. These values are used to draw Receiver Operating Characteristic (ROC) curves that show the sensitivity tradeoff of our detector.

4.2 Learning and Generalization Ability

Our data set was divided in a training set and a validation set in approximately 7%-93% proportion. The training phase had a maximum of 400 epochs and the performance goal was set to 10^{-9}. In figure 11 we can see the detection performance of the corresponding ANN in the face of new data, ie the validation set. We visualize the detection ability of the MLP in figure 11 where real

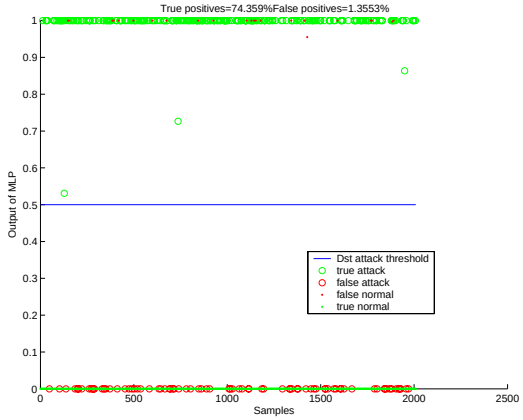


Figure 11: Output of the detection system with random order of samples.

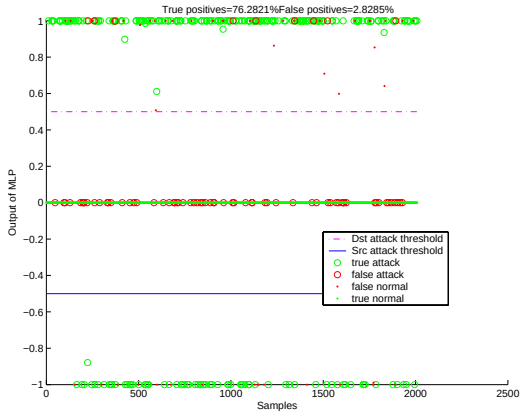


Figure 12: Output of the detection system with random order of samples and "DDoS source","DDoS victim" classification.

attacks are represented by circles and normal states by dots. The horizontal

line represents the threshold of 0.5 above which results are classified as attacks. Using the "DDoS source", "DDoS victim", "Normal" classification scheme we see the results in figure 12. The results shown are the output of the neural network before being filtered by the binary threshold function. These results obviously show that using a neural network for DDoS attack detection is a promising approach. Without further improvements a successfully trained MLP can classify attacks and normal states with $TPR > 74\%$ and $FPR < 3\%$. Taking into account the relatively small size of training set, the large size of the validation set, along with the fact that our data stems from several hours of real production-network traffic, the generalization ability of the neural network is impressive.

4.3 Preprocessing

In this section we will discuss potential improvements of our approach with the use of preprocessing techniques. In the beginning we will explore the use of normalization. In figure 13 we can see that we have a higher concentration of points in the upper left area when we normalize our input data $i(t)$ in the $[-1, +1]$ interval. This is a standard practice in the field of Neural Networks that helps our network to be trained easier. We define $i_{norm}(t) = \frac{2*(i(t)-min(i))}{max(i)-min(i)} - 1$. The vectors min(i) and max(i) contain the minimum and maximum values of the input signals and are specific to the metrics we use. After the network has been trained, these vectors should be used to transform any future inputs that are applied to the network. They effectively become a part of the network, just like the network weights and biases.

Another potential improvement is based on the common belief that we can

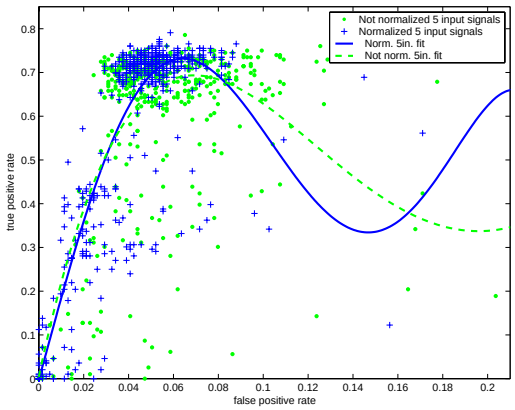


Figure 13: Impact of normalization on performance

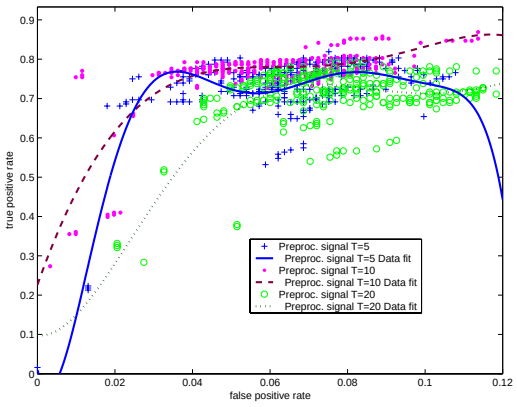


Figure 14: Output of the Detection System using window size T=5,10,20.

decide if a DDoS attack occurs not only based on instantaneous metrics but also judging from the metric's past values. We follow this approach and we

augment our original input vector for example $i(t) =< UR(t), FR(t) >$ with each metric's difference from an average over a window $W = (t - T, t)$ of past T values [2]: $i(t) =< UR(t), FR(t), diff(UR(t)), diff(FR(t)) >$ where $diff(x(t)) = \frac{x(t)-avg(x(W))}{avg(x(W))}$.
We train and evaluate the performance of the MLP using a window size T=5,T=10 and T=20 and show the results in figure 14. On the one hand we would expect history to aid the detection performance since attacks often appear as abnormal bursts in network characteristics that follow a diurnal circle. On the other hand, if the history window we take into account consists of several successive abnormal measurements (which depends on the duration of the attack) we might get a false impression of stability. In any way our results indicate that further research is needed to determine whether to include past values in the input vector and what the optimum window size is. A comparison of the MLP's performance using the original or augmented input is shown in figure 15.

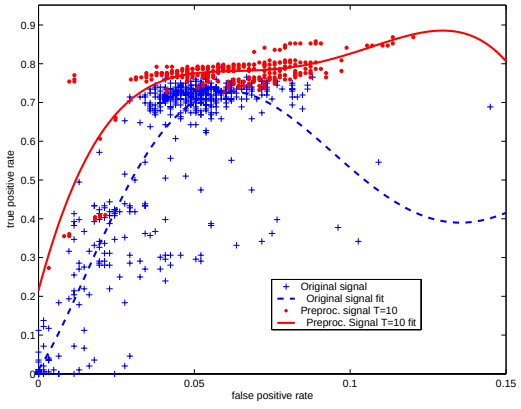


Figure 15: Comparison of original and preprocessed input vector

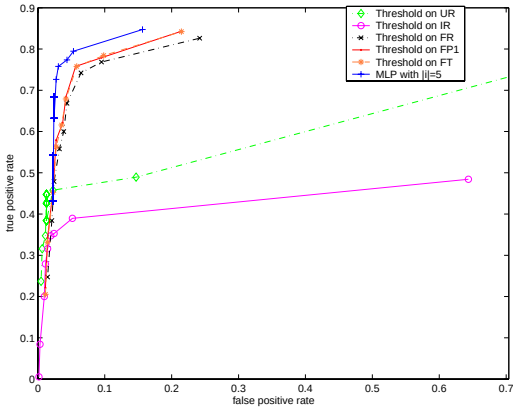


Figure 16: Comparison with simple thesholds

4.4 Comparison with simple thresholds

In order to have a comparative estimate of the performance of our detection mechanism we have used sigmoid thresholds on single metrics as simple detection sensors. The results that are shown in figure 16 are indicative of the superiority of a data-fusion technique like our MLP. By combining several metrics and by training the ANN we are able to achieve much higher detection rates with the same number of false positives. These results are promising for the development of truly efficient DoS detection systems that take advantage of several detection heuristics.

[2] we abuse the formal mathematical notation and we represent avg(UR(t-T),UR(t-T+1),...,UR(t)) as avg(UR(W))

5 Conclusion

In this paper we view DDoS detection as a classification problem and propose a set of detection metrics that are based on complementary passive measurement methods like packet capturing and Netflow based traffic monitoring. Our metrics are fused with a multilayer Perceptron (MLP) that classifies the network state into 2 (attack and normal) or 3 (DDoS source, DDoS victim, normal) classes. The evaluation of our detection engine was based on experiments that used a customized version of the TFN2K tool, capable of launching UDP floods with adjustable bandwidth and packet rate, and took place in a national academic ISP network. Detailed performance measures of various configurations were presented in the form of ROC curves. To sum up, we consider our detection approach successful as it is able to classify attacks with high true positive rates while keeping false positives low. We demonstrated that we don't need complicated per IP address or per flow metrics to detect DDoS attacks in a real operational network. Based on multiple but simple detection metrics we can create simple classifiers such as our MLP that can detect attacks originating or destined to an edge network. We hope that it opens a new direction in DDoS detection research where multiple and rather high level detection metrics and simpler analysis methods (with smaller computational complexity) are used.

References

[1] CERT/CC, "Cert advisory ca-2003-20 w32/blaster worm," August 2003, `http://www.cert.org/advisories/CA-2003-20.html`.

[2] Brian McWilliams, "Cloaking device made for spammers," .

[3] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proceedings of ICNP 2002*, Paris, France, Nov. 2002, pp. 312–321.

[4] Thomer M. Gil and Massimiliano Poletto, "MULTOPS: A data-structure for bandwidth attack detection," in *Proceedings of the 10th USENIX Security Symposium*, August 13–17, 2001, Washington, DC.

[5] Chen-Mou Cheng, H.T. Kung, and Koan-Sin Tan, "Use of spectral analysis in defense against DoS attacks," in *Proc. of IEEE GLOBECOM*, 2002,

[6] Cristian Estan, Stefan Savage, and George Varghese, "Automatically inferring patterns of resource consumption in network traffic," in *Proceedings of the ACM SIGCOMM Conference*, Germany, Aug. 2003.

[7] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Internet Measurement Workshop*, 2002.

[8] Aditya Akella, Ashwin Bharambe, Mike Reiter, and Srinivasan Seshan, "Detecting ddos attacks on isp networks," in *ACM SIGMOD/PODS Workshop on Management and Processing of Data Streams*, 2003.

[9] C. Siaterlis and B. Maglaris, "Towards multisensor data fusion for DoS detection," in *Proceedings of ACM SAC'04*, Nicosia, Cyprus, 2004.

[10] Jun Jiang and S. Papavassiliou, "Detecting network attacks in the internet via statistical network traffic normality prediction," in *Journal of Network and Systems Management*, March 2004, vol. 12, pp. 51–72, Special Issue: Security and Management.

[11] R. J. P. De Figueiredo, "Implications and applications of kolmogorovs superposition theorem," in *IEEE Transactions on Autom. Control*, 1980, p. 12271230.

[12] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition," 1957, (In Russian).

[13] M. T. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," in *IEEE Transactions on Neural Networks*, 1994, vol. 5, pp. 989–993.

[14] I. Charitakis, D. Pnevmatikatos, E. Markatos, and K. Anagnostakis, "S2I: a tool for automatic rule match compilation for the ixp network processor," in *Proceedings of the 7th International Workshop on Software and Compilers for Embedded Systems*, Vienna, Sep. 2003.

[15] Brian Caswell and Marty Roesch., "Snort: The open source network intrusion detection system," `http://www.snort.org`.

[16] Tobi Oetiker, "About RRDtool," `http://people.ee.ethz.ch/~oetiker/webtools/rrdtool`.

[17] CISCO, "Netflow," `http://www.cisco.com/go/netflow`.

[18] Steve Romig, Mark Fullmer, and Ron Luman, "The OSU Flow-tools package and CISCO NetFlow logs," in *Proceedings of 14th Systems Administration Conference (LISA 2000)*, 2000, pp. 291–303.

[19] C. Siaterlis and B. Maglaris, "Detecting ddos attacks with passive measurement based heuristics," in *Proceedings of the 9th IEEE Symposium On Computers and Communications* , Alexandria, Egypt, 2004.

[20] Jason Barlow and Woody Thrower, "TFN2K - an analysis," March 2000, `http://packetstormsecurity.com/distributed/TFN2k_Analysis-1.3.txt`.