Machine Learning for Automatic Defence against Distributed Denial of Service Attacks

Stefan Seufert and Darragh O'Brien
School of Computing, Dublin City University, Ireland

Abstract— **Distributed Denial of Service attacks pose a serious threat to many businesses which rely on constant availability of their network services. Companies like Google, Yahoo and Amazon are completely reliant on the Internet for their business.**

It is very hard to defend against these attacks because of the many different ways in which hackers may strike. Distinguishing between legitimate and malicious traffic is a complex task. Setting up filtering by hand is often impossible due to the large number of hosts involved in the attack.

The goal of this paper is to explore the effectiveness of machine learning techniques in developing automatic defences against DDoS attacks. As a first step, a data collection and traffic filtering framework is developed. This foundation is then used to explore the potential of artificial neural networks in the defence against DDoS attacks.

I. Introduction

Computer Security rests on three columns: Confidentiality, integrity and availability [1]. Accordingly, the major threads in security research are breach of confidentiality, failure of authenticity and unauthorized denial of service [2].

According to [3] the term "Denial of Service" (DoS) was originally coined by Gligor in an operating system context [4] but since became widely adopted. Unlike random failures which tend to affect only a few nodes at a time, DoS attacks are designed to bring down all nodes providing a specific service. A DoS attack involving more than one computer to mount an attack on a target in a coordinated manner is called a Distributed Denial of Service (DDoS) attack. Combining the resources of multiple systems in order to attack a single victim makes it possible to outnumber the computing resources available to the target. Resource exhaustion is the most common type of attack seen today and this paper concentrates on such attacks.

As attack tools continue to advance, the attackers' focus is shifting from the Network Layer (Open Systems Interconnection (OSI) Reference Model Layer 3) towards the Application Layer (OSI Layer 7). Since applications tend to perform many more computations per packet than a network layer mechanism, less traffic is needed to cause the application to exhaust all CPU resources and to fail to handle valid requests [5].

In the remainder of this paper the authors first give an overview of existing approaches. Then the problems associated with defending against DDoS are summarized in Section III. A new defensive approach based on machine learning is proposed in Section IV and results are presented in Section V.

II. Related Work

Various methods have been suggested to counter DDoS attacks. Traceback schemes try to locate the real source of the attack while overlay networks attempt to guarantee the safe flow of information by authentication. Autonomous systems that do not carry transit traffic can also use ingress and egress traffic filtering to drop traffic not originating from or destined for their networks. Another approach is the idea to build traffic aggregates by classifying the traffic. Thus the traffic with the highest probability of being malicious can be filtered out. Two basic approaches based on this idea exist, signature detection and anomaly detection [6]:

A. Signature Detection

When performing signature detection the incoming traffic is scanned for known signs of DDoS attacks (e.g. the fixed TCP sequence numbers used by the DDoS tool Shaft). A database of signatures is built by hand a priori [7]. While the signatures generally are a very good indicator of malicious traffic, because human expert knowledge is used to extract them, such a solution cannot adapt to new attacks without human intervention. Creating these signatures takes a significant amount of technical skill. Thus this approach offers no quick defence against formerly unknown (Zero-Day) attacks.

B. Anomaly Detection

Anomaly detection does not use fixed signatures but compares the current traffic to a baseline profile [8]. Various methods have been proposed to accomplish this. They include statistical approaches like a Chi-Square-Test on the entropy values of the packet headers [9], covariance analysis [10], clustering and feature space modelling [11]. Different techniques taken from pattern analysis and machine learning such as Wavelets, Markov Models [5], Genetic Algorithms [12], Artificial Neural Networks (ANN) [13], [14] and Bayesian Learning [8] have also been applied.

III. Problems associated with defending against DDoS attacks

With DDoS attacks posing a very real threat to the operation of many businesses, countermeasures need to be developed. This proves to be a very difficult task because of the following factors:

A. Multiple attack vectors

Attack vectors are paths or means by which a hacker can gain access to a computer or network server in order to deliver a payload or cause malicious outcome[1]. Companies rely on a multitude of services to run their business. Web, mail and name servers are directly exposed to the Internet and thus are threatened most. But it is also possible to hit one of the back end services (e.g. database, authentication and billing) by using front end functions that impose a high load on them. Thus it is necessary to protect every service. Otherwise, overloading a single critical service can be sufficient to bring the victim's business to a halt. Especially for rarely used services, it is often too costly to keep adequate resources on standby to mitigate a coordinated DDoS attack.

Furthermore, attackers may use new methods or modify existing attacks to circumvent established defence mechanisms. Static defences do not work if a yet-unknown attack is used. Instead the system needs to adapt to new types of attack. Although adaptive filters provide a promising approach they are not always up to the task [15].

B. Multiple sources

It is difficult to block malicious traffic by hand, because in the case of a DDoS attack, requests are generated by many sources. The traffic from each source must be inspected and classified. Subsequently, appropriate filtering rules need be applied. Doing this manually for a large number of hosts is generally infeasible. The number of filtering rules which can be installed on a router/firewall is often limited. Therefore it can be impossible to set up a filter for each source when very large botnets are involved in the attack.

In the case of faked source addresses, filtering based on the source address is useless. Thus it is more desirable to identify the nature of the attack and to block packets based on their payload before they arrive at the victim's site.

C. Mix of benign and malicious traffic

Even during a DDoS attack there still is a proportion of bona fide service requests to use the service. This makes it harder to inspect the traffic and to work out a classification scheme for traffic filtering. Since not all incoming requests can be assumed to be part of the attack it is more complex to derive appropriate filtering rules. If the filters chosen are too specific they do not block the attack, if they are made too general they may block legitimate traffic [16].

D. Differentiation at various levels

Effective filtering is further complicated by the fact that differences between malicious and benign traffic may most easily be detected at different levels of the network stack. If large "ping" (ICMP/UDP) packets are used to exhaust the available bandwidth they are only visible on OSI level 3. If valid HTTP requests are used they can only be distinguished at OSI level 7 by looking inside the HTTP headers of the request [17]. Thus neither a detection mechanism located at OSI level 3 nor one located at OSI level 7 can detect both attacks.

E. Flash crowds

The root cause of an increased load situation may not always be an ongoing attack. If a service gains large public attention it may easily be overloaded by the unforeseen increase in valid requests. This has happened in the past when reports containing links to a web site were published on popular news portals like Slashdot. Thus this problem has been jokingly called the "Slashdot Effect". Some websites have even suffered overload situations due to unexpectedly successful marketing campaigns[2]. A DDoS protection system should not block increased traffic in flash crowd situations as long as the servers can still handle the load. It may however need to react when the system becomes overloaded [16].

F. Filter placement

Another problem that needs to be considered is the placement of the filters. Basically, there are three possible locations: Close to the source, close to the victim and within the intermediate network [6]. While placement close to the source or within the intermediate network is very desirable because the problem will be eliminated at its root, practical problems prevent such a solution. Although technical solutions like route-based distributed packet filtering [18] and traceback schemes [19] have been proposed, the necessary cooperation amongst the carriers is hard to achieve. Placing the filters close to the victim thus seems to be the easiest approach because a potential victim has a much stronger incentive to deploy defence mechanisms than network service providers [20].

G. Throughput

Today's Internet backbone operates at multi-Gigabit speeds. Most hosting sites are located close to the backbone and have at least Fast Ethernet connectivity to the network [21]. Thus it is important for any defence system to keep up with that load. Systems that keep per-flow or per-connection statistics often do not scale up to Gigabit speeds [22].

IV. Our Approach

To further extend the effectiveness of existing DDoS defences the authors investigated how machine learning algorithms can be used to counter a DDoS attack in an automated fashion. To this end we built an extensible framework that can be used to test machine learning algorithms for defending against DDoS attacks. Although machine learning techniques have been used before we introduce some novel ideas on how they can be applied to make the defence more effective. Our key features are:

1) Fully automated learning approach, no human interaction necessary. Can deal with Zero-Day attacks.

[1]`http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci1005812,00.html`

[2]`http://www.cbsnews.com/stories/2003/01/27/scitech/pcanswer/main538171.shtml`

2) Monitoring of resource utilisation instead of the incoming traffic for attack detection.
3) Using Artificial Neural Networks for anomaly detection (other machine learning algorithms can easily be integrated).
4) Extensible, distributed data collection and filtering system.
5) No expert knowledge required except basic information about protocol structures.

A. Overview

The system is composed of several parts that are combined to effectively counter DDoS attacks:

1) Attack Detection: All systems we know of monitor the network traffic in order to establish whether the system is under attack or not. Instead, in this approach distributed agents are used. They report resource utilisation levels to the master at regular intervals. These values are compared against threshold values configured by the administrator. If the limits are exceeded the system is considered to be under attack.

This approach is similar to the modus operandi used in medicine. A doctor does not decide on the status of a patient based on the food the patient has recently consumed. Instead symptoms like an increased temperature are used to detect an infection. Where required further action will be taken. If no symptoms are present, no treatment is applied. This is important because drugs can have severe side effects. Ideally treatment begins when the patient shows the first symptoms but before he/she gets seriously ill.

Transferring this approach to the area of DDoS defence means that action is only taken if the system becomes overloaded due to an attack. We do not care about ineffective attacks nor do we take any action when the system is considered to be operating normally. This reduces the chance that legitimate requests are dropped because of a false positive as might occur with systems looking at traffic metrics only. As soon as the system is found to be under attack countermeasures are immediately deployed, accepting that some benign requests might be dropped. However, as the system already is under attack this is much more acceptable than being unable to answer any requests at all.

2) Feature Extraction: Although there is no consensus on what kind of traffic should be viewed as ”abnormal”, the conventional wisdom is that the traffic generated by DDoS attacks usually exhibits some unique characteristics [7]. Similar to the load measurements described above, a distributed set of probes is used for feature extraction. Crucially we do not focus on a single set of features but use a set of probes to extract features from various protocol layers. We use metrics from the headers of IP packets (OSI Level 3), TCP headers (OSI Level 4) and data from the application layer such as HTTP requests (OSI level 7). This provides a much wider spectrum of data to analyse and thus a higher probability of finding a feature which can be used for successful classification of the traffic. It previously has been shown that the fusion of multi-sensor data is useful for attack detection [23] but to our knowledge has never been incorporated into a DDoS defence system.

3) Training Phase: Once an attack has been detected it is necessary to classify future incoming requests as benign or malicious. During this stage, machine learning techniques are used to dynamically train an algorithm to learn the differences between the feature sets collected during normal operation and those received while under attack. Only one half of the data is used to train the algorithm. The other half is used to evaluate the result of the training. If the training is successful (i.e. the algorithm can discriminate between the two sets) then the algorithm can be used to classify incoming requests and to filter malicious traffic in the next step. Otherwise it is assumed that the algorithm is not able to detect the attack using this specific feature set and it is not used for filtering.

4) Traffic filtering: Using the fully trained algorithms from the previous step malicious packets can now be eliminated early and before they can cause any harm to the infrastructure. The classifier can either be directly used to filter the traffic or it can be used to generate rules for an existing filtering framework to block traffic from sources that have recently sent malicious requests.

B. System Architecture

As shown in Figure 1 the system is built around a central server. The load and traffic probes are distributed on the victim site and send usage data and traffic samples to the server. The management interface can be used to retrieve status information and to modify the configuration of the system. All components are installed on the victim's premises.

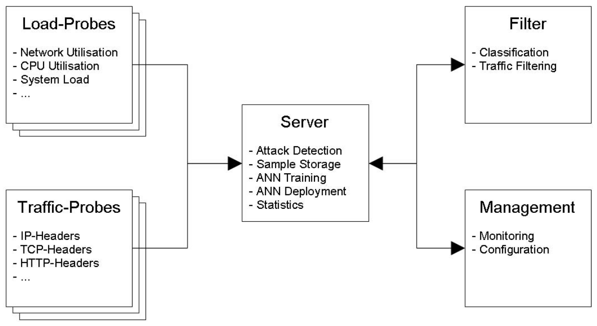

Fig. 1. The Architecture of the System

1) Load Probes: The load probes monitor resources crucial to the delivery of the service. The system can easily be extended to support new probes. Each probe monitors a specific part of the system and regularly sends updates to the central server. We implemented load probes to monitor the utilisation of a network link, the system load average and CPU utilisation.

2) Traffic Probes: The traffic probes collect traffic samples at various points in the system. Using a hook into the Linux Netfilter subsystem we can process incoming packets in user space. We extract salient values from the headers of IP, ICMP, UDP and TCP packets. At the application level we generate statistics about the user's behaviour on a web server and extract additional information from the HTTP-headers. Table I gives an overview over the data collected in our implementation.

3) Server: The server is the central component in the system. It performs two basic tasks.

Its first job is to handle the communication with the other components of the system. It receives information from the traffic and load probes and stores them in its internal data structures for further processing. It also handles requests from the management interface.

Its second task is to periodically check the state of the system. If the system is operating normally then the traffic samples received since the last check are merged into the baseline profile. When the system is found to be under attack the training component is activated. The recently received packets and the previously established baseline are used as input to the machine learning algorithm's training system. This is carried out for all features extracted in the previous step. If the algorithm succeeds in distinguishing adequately between the baseline profile and the traffic collected during the attack we can use this result to defend against the attack.

4) Traffic Filters: Lastly, the traffic filters drop malicious requests using the algorithms produced during the training session. Above a certain threshold the packets are dropped according to their probability of being malicious. So even if a benign request is wrongly classified as malicious and is consequently dropped, there still remains a possibility that it will pass the filter if resent. Since most protocols are designed to resend packets in case of data loss, this will only result in a delay. Thus probabilistic packet dropping will drop all traffic which has a high probability of being malicious and slow down dubious traffic.

5) Management Interface: The management interface is implemented as a web application. It allows the user to view the state of the system and to alter the systems configuration (e.g. set the threshold values). It also allows the system to be switched on or off.

C. Artificial Neural Networks

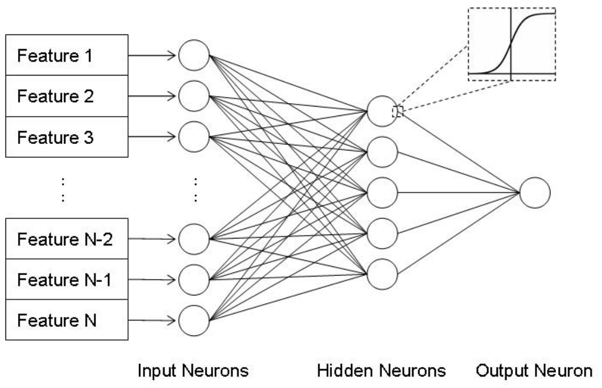

Fig. 2. A three-layer feed-forward neural network with a sigmoid transfer function

We implemented a classifier based on Artificial Neural Networks for the framework described above. ANNs provide a general, practical method for learning real-valued, discrete-valued and vector-valued functions from examples. ANN learning is robust to errors in training data [24]. This makes them a good choice to classify the network traffic while under attack. The robustness to errors in the training data is especially important in our case because the samples collected during the attack phase will contain legitimate requests, too. Similarly baseline traffic can contain malicious requests from previous, unnoticed attacks.

A simple three-layer feed-forward network with sigmoid transfer functions as shown in Figure 2 has been used for all experiments. The data extracted by the traffic probes is fed into the input layer. The number of input neurons was always chosen equal to the number of features available from the actual probe. We used a fixed number of hidden neurons which was determined using a separate experiment. A single output neuron was used as an indication of the likelihood of the packet being malicious in nature. The sigmoid function is used to map the output of each computation back into the range $[0, 1]$.

Training was implemented using an extended backpropagation algorithm as described in [24]. The ANN was trained to output a value of 1.0 for samples collected during an attack and a value of 0.0 for benign samples. The progress of the training was observed by calculating the mean square error between the actual and the expected output of the ANN.

V. Results

A series of experiments was carried out to examine the capabilities of the presented approach. First the classification performance of the ANN system was tested in isolation. Then the system as a whole was put to the test using a network emulator.

A. Classification performance

Because the performance of the system as a whole heavily depends on the accuracy of the classification of the machine learning algorithm used, we first tested our ANN implementation in isolation. We captured samples of legitimate and attack traffic using our traffic probes. For the legitimate traffic we observed users browsing a web site using different browsers. As examples of attack traffic we recorded a Flood Ping (massive ICMP echo request/replies), UDP flooding (storm of UDP packets with faked source addresses) and excessive web page requests generated using Apache Bench (benchmark tool supplied with the Apache web server). Using these samples various tests have been performed.

1) Classification ability: Although ANNs have been used in traffic classification before, we had to check if our implementation had the basic ability to classify the traffic samples. Networks with ten hidden neurons were used in this test. We used a maximum of 100 training rounds to limit the time needed for training. In each round 100 benign and 100

TABLE I
TRAFFIC PROBES

Type	OSI Level	Stateful	Features used	Goal
IP packet header	3	No	Header fields from raw IP packets	Detecting anomalies at IP level
ICMP packet header	3	No	Header fields from ICMP packets	Detecting anomalies at ICMP level
UDP packet header	3	No	Header fields from UDP packets	Detecting anomalies at UDP level
TCP packet header	4	No	Header fields from TCP packets	Detecting anomalies at TCP level
URL session histogram	7	Yes	Histogram of the hashes of the URL visited in a session	Detecting strange user behaviour
Selected HTTP headers	7	No	Hashes of selected HTTP headers specific to Browsers	Isolating malicious clients
Present HTTP headers	7	No	Headers present in the HTTP request	Isolating malicious clients

TABLE II
CLASSIFICATION ABILITY OF A ANN WITH 10 HIDDEN NODES

Malicious Sample	Traffic Probe	Error
Flood Ping	IP Header	0.00018324
UDP Flood	IP Header	0.00000502
Apache Bench	Spec HTTP hdrs	0.00000001

malicious samples where presented to the learning algorithm. The results can be seen in Table II.

The table lists the type of malicious traffic used in each experiment together with the feature set that yielded the best classification results. For each experiment the mean squared error is given. We observed no false accepts and no false rejects during our experiments. Only a negligible squared error remained after the training run. This shows that the ANN not only classified the samples correctly but also that it did so with a high degree of confidence. The system proved that it is very well able to learn the difference between the data sets.

2) Number of hidden neurons: After establishing that our ANN implementation can classify the samples we examined the influence the number of hidden neurons has on the classification results. Figure 3 depicts the results.

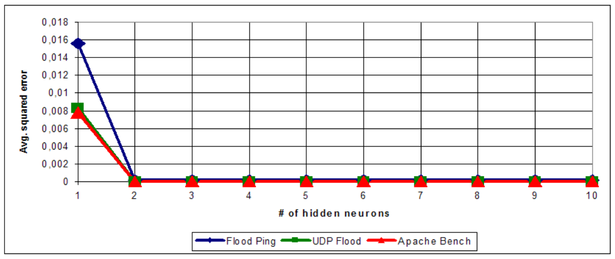

Fig. 3. Influence of the number of hidden neurons on the classification results

When only one hidden neuron is used then the ANN is effectively reduced to a two-layer network. This is obviously not sufficient to classify the traffic. Starting with two neurons the ANNs are able to classify the data with a high degree of certainty. This allows us to work with less than our initial ten hidden neurons speeding up training and processing.

3) Impact of data quality: As mentioned, we have to deal with dirty data in a real-world scenario. That means that the traffic recorded during an attack will contain legitimate requests and that the benign samples will contain malicious traffic. It is important to understand the effects of this mix on the classification results. We deliberately added impurities to our data sets by mixing our input data sets. Figure 4 shows our findings. With the pure data sets we have no false accepts and no false rejects. When we deliberately add some noise a small percentage of samples is wrongly classified (e.g. about 6% of the malicious packets are wrongly accepted if the baseline contains 10% attack traffic and the attack traffic includes 10% legitimate requests).

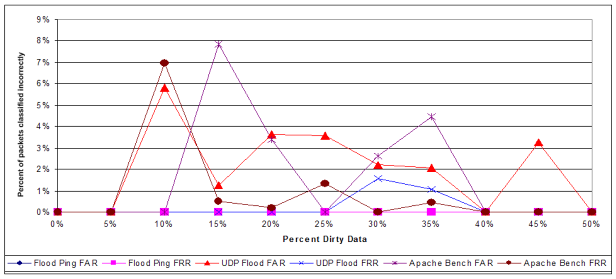

Fig. 4. False acceptance (FAR) and false rejection rates (FRR) for input data of different quality

The results show that our classification algorithm can function effectively in the presence of dirty data. As expected the classification ability decreases slightly and some malicious packets are accepted while some benign packets are dropped. With the errors staying well below 10% the system remains effective.

B. Simulated attack results

After performing some experiments under laboratory conditions we tested the system using the NCTUNS network simulator[3] in emulation mode. We built a small Internet-like network containing twenty nodes. The victim system was running inside a virtual machine. The host computer was also part of the simulation and used as a test client. Half of the nodes were used to launch different kinds of attacks against

[3]`http://nsl10.csie.nctu.edu.tw/`

our victim system. The other computers were used to measure access times to the web server running on the victim system. The results can be found in Figure 5.

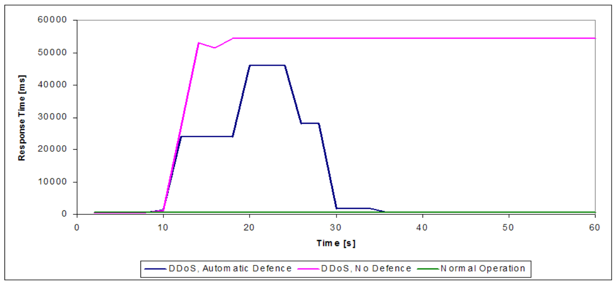

Fig. 5. Response times for a CPU intensive website. An application level DDoS attack requesting one single page is started at 10 seconds

The unprotected system shows high response times. The protected system returns to normal response times after the attack has been detected and countermeasures have been installed.

VI. Conclusion

Defending against DDoS attacks is a complex and complicated topic. Unless traceback schemes become universally established, a defence close to the victim is the most practical solution.

We were able to present a proposal for DDoS defence based on Artificial Neural Networks which achieved excellent detection results. Unlike other proposed solutions we do not try to detect the presence of an attack from the traffic. Instead we monitor the resource usage within the system. We do not try to identify a single metric that can be used to classify all possible types of attacks. Further differentiating ourselves from previous approaches, data is collected at different levels of the network stack and fed into the training algorithm. We then filter the requests at the level which proved itself to be most discriminative during training. Although we use ANNs as the machine learning algorithm of our choice we have built a framework that is easily extensible with different algorithms.

To date one machine learning algorithm has been tested. We intend to extend this solution to use multiple algorithms in the future.

References

[1] M. Bishop, *Introduction to Computer Security*. Addison-Wesley Professional, 2004.

[2] R. M. Needham, "Denial of service," in *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 1993, pp. 151–153.

[3] K. Lakshminarayanan, D. Adkins, A. Perrig, and I. Stoica, "Taming ip packet flooding attacks," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 45–50, 2004.

[4] V. D. Gligor, "A note on denial-of-service in operating systems," *IEEE Trans. Softw. Eng.*, vol. 10, no. 3, pp. 320–324, 1984.

[5] Y. Xie and S.-Z. Yu, "A novel model for detecting application layer ddos attacks," in *Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on*. IEEE Press, 2006, pp. 56–63.

[6] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: a classification," in *Signal Processing and Information Technology, 2003. ISSPIT 2003. Proceedings of the 3rd IEEE International Symposium on*. IEEE Press, 2003, pp. 190–193.

[7] Y. Xu and R. Guerin, "On the robustness of router-based denial-of-service (dos) defense systems," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 47–60, 2005.

[8] M. V. Mahoney and P. K. Chan, "Learning nonstationary models of normal network traffic for detecting novel attacks," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2002, pp. 376–385.

[9] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, vol. 1. IEEE Press, 2003, pp. 303–314.

[10] S. Jin and D. Yeung, "A covariance analysis model for ddos attack detection," in *Communications, 2004 IEEE International Conference on*, vol. 4. IEEE Press, 2004, pp. 1882–1886.

[11] S.-Y. Jin and D. Yeung, "Ddos detection based on feature space modeling," in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 7. IEEE Press, 2004, pp. 4210–4215.

[12] T. Shon, Y. Kim, C. Lee, and J. Moon, "A machine learning framework for network anomaly detection using svm and ga," in *Systems, Man and Cybernetics (SMC) Information Assurance Workshop, 2005. Proceedings from the Sixth Annual IEEE*. IEEE Press, 2005, pp. 176–183.

[13] D. Gavrilis and E. Dermatas, "Real-time detection of distributed denial-of-service attacks using rbf networks and statistical features," *Comput. Netw. ISDN Syst.*, vol. 48, no. 2, pp. 235–245, 205.

[14] Y. Xiang and W. Zhou, "Mark-aided distributed filtering by using neural network for ddos defense," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 3. IEEE Press, 2005, pp. 1701–1705.

[15] Q. Li, E.-C. Chang, and M. Chan, "On the effectiveness of ddos attacks on statistical filtering," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 2. IEEE Press, 2005, pp. 1373–1383.

[16] R. Thomas, B. Mark, T. Johnson, and J. Croall, "Netbouncer: client-legitimacy-based high-performance ddos filtering," in *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, vol. 1. IEEE Press, 2003, pp. 14–25.

[17] F. Kargl, J. Maier, and M. Weber, "Protecting web servers from distributed denial of service attacks," in *WWW '01: Proceedings of the 10th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2001, pp. 514–524.

[18] K. Park and H. Lee, "On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2001, pp. 15–26.

[19] C. Huang, M. Li, J. Yang, and C. Gao, "A real-time traceback scheme for ddos attacks," in *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, vol. 2. IEEE Press, 2005, pp. 1175–1179.

[20] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: an effective defense against spoofed ddos traffic," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2003, pp. 30–41.

[21] D. K. Y. Yau, J. C. S. Lui, F. Liang, and Y. Yam, "Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 29–42, 2005.

[22] R. R. Kompella, S. Singh, and G. Varghese, "On scalable attack detection in the network," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM Press, 2004, pp. 187–200.

[23] C. Siaterlis and B. Maglaris, "Towards multisensor data fusion for dos detection," in *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*. New York, NY, USA: ACM Press, 2004, pp. 439–446.

[24] T. Mitchell, *Machine Learning*. McGraw-Hill Education (ISE Editions), 1997.